Pairwise Approach to Premier League Ranking

Mohammed Zaid Ally
School of Computer Science and Applied Mathematics, Wits University
1828282@students.wits.ac.za
Supervised by Dr Hairong Bau and Prof Terence L Van Zyl

***Abstract*—The Premier League is the most competitive football league in the world. This is due to the constant inflow of capital which allows any team to win the title every year. It is an exciting spectacle that thrills people all across the world because of its inherent unpredictability. Fans' intense involvement and the readily available statistical data present a great opportunity for bookmakers and sports analysts to predict the club standings. Pairwise ranking will be used in this research to forecast the final league standings. The goal of this research is to predict and compare the effectiveness of five different models: a transformer-based model, a RankNet baseline model, a hybrid SNN+XGBoost approach, a Siamese Neural Network (SNN), and a standalone XGBoost model. Conventional models frequently fail to account for the relationships between team features.**

Our models are based on data from the 2012–13 Premier League season through the 2022–23 season. As our main model, we propose the SNN+XGBoost model, which is improved by adding SNN activation values. Our models produced accurate rankings; the transformer-based and XGBoost models performed best, with a mean Average Precision (mAP) of 0.86 and a Normalized Discounted Cumulative Gain (NDCG) of 0.99 for the 2021/22 season. In addition, the SNN+XGBoost model continuously produced strong results, as seen by its 0.98 NDCG score and 0.85 mAP for the same season.

This research is significant because it can be used to make predictions about the English Premier League. It offers insights into the benefits and drawbacks of both conventional and cutting-edge machine learning models in the sports industry by contrasting the various models. These models can be used to improve strategies for specific teams and offer betting companies useful information.

Keywords: Premier League, pairwise ranking, XGBoost, SNN, transformer-based models, NDCG, mAP, Ranknet

I. Introduction

The English Premier League, with its illustrious history and worldwide audience, has become an incredibly fascinating avenue for data scientists and betting companies. The varying performance of teams, driven by money, players, and tactics, allows for an exacting task to any machine learning model. This paper aims to successfully rank these football teams based on their performance in previous seasons using varying machine learning techniques.

We first reviewed relevant literature, paying special attention to the application of transfer learning—which involves adapting a pre-trained model to a new problem [29]. The main purpose of this approach's investigation was to evaluate its effect on English Premier League ranking [16]. In order to determine similarity and provide ranks, an SNN is a neural network design in which the similarity between inputs is determined. They are made up of two identical subnetworks with the same weights that are fed as an input pair. SNN is not particularly effective at ranking but excels at learning features [6]. Furthermore, a RankNet model [3] was utilized because of its capacity to forecast pairwise rankings, producing a score ranging from 0 to 1. A higher score denotes a better probability of one team surpassing another.

Predictive modeling was expanded to other sports, like basketball and rugby, by key literature [32], where team similarities were evaluated by an SNN. It was possible to calculate similarity scores between team pairs using this methodology even though the statistical grounds were different from football. List-wise rankings were then generated by feeding these scores into an XGBoost [5] and LightGBM [12] models. By gradually merging weak models into a powerful predictive model, XGBoost improves predictions. On the other hand, LightGBM employs a tree-based learning algorithm to rank NBA clubs. Data was run through the SNN for each season, and LightGBM and XGBoost used the similarity scores that were produced for each game. Ultimately, these scores were used by a TallyRank method to establish the season's final standings.

The research questions that this paper aims to answer are:

- If SNN activation values are added to paired team data and then fed into an XGBoost model, will this improve Premier League rankings compared to only using SNN or XGBoost separately?
- When it comes to the pairwise ranking of Premier League teams, does the use of a transformer-based model perform better than the outcomes documented in earlier research that used alternative machine learning techniques?

The methodology used creates pairwise data for all participating teams by evaluating 380 unique matches that are played in the English Premier League each season. The first model we use as a baseline is the RankNet model [16], which computes the Euclidean distance between teams to produce pairwise rankings. The neural network ends with a linear layer as the output. Then, using a similar process, an SNN model is used, with activation values obtained from a Rectified Linear Unit (ReLU) in the last layer. Every season,

the XGBoost model also determines pairwise rankings. The paired data and the ReLU activation values of the SNN model are then combined, utilizing transfer learning to create an improved SNN+XGBoost model. Lastly, a Transformer-based model is presented [19], which explicitly models the links between paired statistics throughout the dataset by means of a self-attention mechanism. Metrics like the mAP and NDCG are used to evaluate each model's effectiveness, guaranteeing a thorough analysis of the final league standings.

Among the novel applications of transfer learning in this paper's contributions is the use of activation values to forecast soccer team rankings. Furthermore, this work introduces an approach not before investigated: the integration of these activation values from an SNN with an XGBoost framework for Premier League ranking prediction. In addition, using a transformer-based model for sports prediction is a new technique that has typically only been used in recommender systems [19]. The use of transformers in this situation is especially encouraging since they may identify intricate relationships in the data and provide insight into the dynamics that exist between teams.

The structure of this research comprises four sections. The first section explores previous work in predicting Premier League standings (Section II). The second section addresses data pre-processing, implementation processes, metrics for evaluation, and the experimental setup (Section III). The third section discusses the results and analysis, as well as future improvements and limitations (Section IV). The final section concludes the research, summarizing the aspects discussed (Section V).

II. BACKGROUND AND RELATED WORK

Using a variety of machine learning algorithms, it is possible to forecast final league rankings while learning to rank teams in sports. This issue is complex and has been researched using various methodologies in many sports. For instance, [32] used deep learning to rank NBA teams, while [16] used similarity learning to rank English Premier League soccer teams, and [26] used artificial neural networks to predict the Turkish Super League. [31] and [24] both used PageRank models to rank NBA teams and [14] used a Linear Support Vector Classifier to predict team rankings.

When using a graph-based model [13], a game graph that depicts the relationship between each team is created. The NBA characteristics are not as extensive here; rather, it considers the results of two teams against each other as well as the marginal effects of the results. The PageRank model that was used to generate the TeamRank model takes into account not just wins/losses/draws but also how this team beat another team and created a weighted graph to predict results. The weighted TeamRank approach was also employed, [13], but the difference here was that the score by which one team beat another team was taken into account, as well as how many times the two teams played each other. In this situation, the point difference is taken into account when generating the network and the adjacency matrix.

Football games are considered to be the most exciting and unexpected sport. There are an unlimited amount of factors to take into account. Two large feature areas were selected to address this challenge. The first is the game's real statistics, including goals scored, goals conceded, clean sheets, etc. The second is that the Premier League is the most competitive and unpredictable league in the world because of the outside elements that affect a team's success [16]. One of the most important variables taken into account was external data [16]. The Transfermakt [25] provided external data, including the number of arrivals, departures, total managers, and other statistics. Due to the inclusion of external data, the RankNet model's match-wise ranking fell but achieved a better list-wise ranking.

Transfer learning is the application of a previously trained model to a new problem. The primary motivation for developing these models was to study the impact of transfer learning on the ranking of the English Premier League [29]. An SNN, like similarity learning, is employed by being provided an input pair that describes the match. Each input is analyzed using a distance metric to determine how similar the two inputs are. To forecast the pairwise ranking of teams, a RankNet model [3] was developed. The RankNet model was demonstrated because of its capacity to generate a score between 0 and 1, with a larger score indicating a better possibility that the first team will be ranked higher.

The physical prowess of a player is a crucial factor that is occasionally overlooked while creating these models. All models typically include seasonal statistics. However, when using an Artificial Neural Network(ANN) [33], it was discovered that including a player's physical and technical characteristics, such as the distance traveled at high speeds and the amount of ground covered during a game, had a significant impact on prediction [26]. As a result, league standing forecasts became more accurate.

Similarity learning is a technique for determining how similar two inputs are. The author employed an SNN to determine the league positions for NBA teams [2]. SNNs are a neural network design in which the similarity between inputs is computed. They are made up of two identical subnetworks with the same weights that are fed an input pair. SNNs are not particularly effective at ranking but excel at learning features. Because SNNs are not good at ranking, the author added the XGBoost model after learning the characteristics to rank the NBA teams. This similarity score was fed into a TallyRank, which calculated the season's final standings [32].

Transformer-based models are normally used for recommender systems [19]. It uses a list of Items using

a Personalized Re-ranking Model (PRM) to give you a list of items from most important to least important. The model consists of three parts: the input layer, the encoding layer, and the output layer. It takes in an initial list of ranked items and tries to re-rank them.

The input layer creates a representation of all items in the initial list and feeds them to the encoding layer [27]. It contains the raw features of items and a personalized vector (PV) that uses user-specific preferences. The PV is then multiplied by the feature matrix to create an intermediate embedding matrix. Position Embeddings (PE) are used to account for the sequential information of items.

The personalized vector is defined as:

$$PV = \text{pre-trained model output} \tag{1}$$

The position embedding is calculated as:

$$PE = \text{learnable position embedding} \tag{2}$$

The second layer is the encoding layer which uses a transformer to encode the mutual influences between pairs of items, user preference, and the ranking order of the initial list. It has a self-attention mechanism which is used to capture interactions between items [23]. The self-attention mechanism is described by the equation:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d}}\right)V \tag{3}$$

The matrices Q, K, V represent queries, keys, and values respectively. d represents the dimensionality of the matrices. Softmax$(\cdot)$ is used to convert the values of the matrices into an appropriate value to be multiplied by vector V.

The output layer generates a score for each item which is then used to rank it. It calculates it using a softmax layer and calculates the probability of a click for each item [11].

$$\text{Score}(i) = \text{softmax}(F(N_x)W_F + b_F) \tag{4}$$

Score(i) is the score for item i, which will be used for re-ranking. Softmax$(\cdot)$ is the softmax function that normalizes an input vector into a probability distribution. $F(N_x)$ is the output of the N_x blocks of the Transformer encoder for the input item i. W_F is a learnable weight matrix in the final scoring layer. b_F is the bias term added to the weighted sum before applying the softmax function.

Despite the fact that sports are very unpredictable, learning to rank sports teams may be accomplished by utilizing a variety of machine learning approaches. The deep similarity model, which enabled the most data to be taken into consideration to make a fair forecast, was the key model that showed the most promise. It was also discovered that there is a trade-off when computing match scores vs league rankings. The obvious nature of our findings demonstrates that a "one size fits all" approach is incorrect. Each sport must be addressed separately in terms of data, model, and measure. Employing transfer learning by augmenting the activation values of an SNN with team statistics and passing it through an XGBoost model for predicting Premier League rankings is an approach that addresses the limitations of previous research. Furthermore, the absence of transformer-based models in sports prediction as they identify intricate relationships in data and provide insight into the dynamics that exist between teams highlights a gap that this research aims to fill.

III. Research Methodology

The overview of the research methodology is divided into four sub-sections. Section III-A details the data processing methods, Section III-B describes the implementation of models, Section III-C discusses the metrics used for evaluating the final standings, and Section III-D outlines the experimental procedure.

A. Data

First, we gathered seasonal data for the seasons 2012–13 through 2022–23. WhoScored.com provided the source of these figures [30] and Table I contains a list of the features used. The 'wins' feature in Premier League standings prediction can only be between 0 and 38, whereas the 'fouls' feature can be between 0 and 1000 for a certain season. This is problematic in two ways. First of all, just because a feature has a greater value does not necessarily mean that it is more significant than another. Second, convergence might be impacted because gradient descent may take longer to converge when features are of different sizes. Here's where the pre-processing phase becomes useful. We first standardize all our numerical data [17]. Standardization helps to ensure that the predictive model is fair and converges quickly resulting in accurate standings. Standardization is given as $Z = \frac{(X-\mu)}{\sigma}$. Here, X represents the original value of the statistic, μ is the mean (average) value of the feature across the dataset, and σ is the standard deviation of the feature.

There were also certain statistics that weren't available. To fill in missing values, we take the mean of the column and replace the missing value with the mean. This ensures consistency across all features.

Every Premier League season, the bottom 3 teams are relegated, which poses a challenge since we do not have the same 20 teams each season. Our solution was to assign a unique team identifier to each team across all 11 seasons. This allows for consistent identification of teams each season.

The goal of this data is to use past season statistics to predict the ranking of teams in the current season. Our data uses a semi-time series approach [1] where training cannot be done in an arbitrary order. Seasons must be trained and tested in chronological order to prevent data leakage, hence we employ

time series cross-validation [4]. The train-test-validation splits are as follows:

- Train on seasons 2012/13-2016/17, validate on 2017/18, and test on 2018/19.
- Train on seasons 2012/13-2017/18, validate on 2018/19, and test on 2019/20.
- Train on seasons 2012/13-2018/19, validate on 2019/20, and test on 2020/21.
- Train on seasons 2012/13-2020/21, validate on 2020/21, and test on 2021/22.
- Train on seasons 2012/13-2020/21, validate on 2021/22, and test on 2022/23.

TABLE I
SOCCER STATISTICS OVERVIEW

	Statistics		
1	Total shots	17	Red Cards
2	OutOfBox shots	18	Caught Offside
3	SixYardBox Shots	19	Clearances
4	PenaltyArea Shots	20	Shots Blocked
5	Total Goals	21	Crosses Blocked
6	SixYardBox Goals	22	Passes Blocked
7	PenaltyArea Goals	23	Total Saves
8	OutOfBox Goals	24	SixYardBox Saves
9	Total Dribbles	25	PenaltyArea Saves
10	Dispossessed	26	OutOfBox Saves
11	Total Tackles	27	Total Passes
12	Dribbled Past	28	Key Passes
13	Interceptions	29	Assists
14	Fouled	30	Wins
15	Fouls	31	Losses
16	Yellow Cards		

To ensure our algorithms function correctly, we require both pairs and labels. The procedure was as follows: We generated 380 unique pairs per season, which included features for both teams, necessary for the pairwise ranking algorithm to operate effectively. Since some teams have had longer tenures in the Premier League, they might have played more matches and, consequently, faced certain opponents more frequently. To address this, we concatenated the pairs from each season within our training data. This concatenation included the 190 pairs from every season, wherein each team competes against all others, thereby accommodating teams that have been relegated. When a team appears for the first time in a test season and their prior statistics are unavailable, they are presumed to have lost all their matches. However, once that season is designated as a training season, their statistics are included. Labels were assigned as 0 or 1, indicating whether team A finished the season above or below team B, respectively. These labels served as our target variables.

B. Algorithms

To identify the optimal hyperparameters for each model, we employed Keras Tuner in TensorFlow [22], which utilizes a random search algorithm. This algorithm systematically explores a wide range of hyperparameter combinations, evaluating each one based on its performance relative to the loss function. A Random search does not test all possible values; instead, it selects random combinations within predefined hyperparameter spaces. The random search algorithm implemented by Keras Tuner performs a specified number of trials—in our case, 20—each trial being a complete execution of the model with a particular set of hyperparameters drawn from the defined search space. Through these iterative trials, the algorithm aims to approximate the best-performing hyperparameters that yield the lowest loss function value, thereby enhancing model convergence. The most effective hyperparameters identified at the conclusion of these trials are documented in Table II, Table III, and Table IV.

TABLE II
ARCHITECTURE AND PARAMETERS OF RANKNET AND SNN MODELS AFTER HYPERPARAMETER TUNING.

Parameter	**RankNet**	**SNN only**
Input layer	51	51
Hidden layer 1	84	67
Hidden layer 2	50	67
Output layer	1	5
Learning rate	0.01	0.001
Dropout rate	0.2	0.085
Epochs	50	50

TABLE III
XGBOOST MODEL PARAMETERS

Parameter	**Value**
max depth	7
Learning rate	0.47
subsample	0.6221
colsample bytree	0.9053
minimum child weight	9
gamma	4.09
lambda (L2)	0.314
alpha (L1)	0.278
Epochs	50

TABLE IV
TRANSFORMER-BASED MODEL PARAMETERS

Parameter	**Value**
number of heads	2
feedforward dimension	256
dropout rate	0.2
number of transformer blocks	1
epochs	20

1) RankNet model: The first model we developed was the RankNet model [3]. RankNet is a type of neural network that includes two identical subnetworks with shared weights. Each subnetwork takes the statistics of a home and an away team as inputs. What makes the RankNet model distinctive is its specialized loss function and the fact that it uses a linear activation function with a single output node. For optimization, we employ the Adam optimizer over 50 epochs with a learning rate of 0.01 [16] seen in Table II. Each subnetwork outputs a score, which we represent as $g(x_a)$ and $g(x_b)$.

Previous work done [16] made use of a probability ranking between two teams.

$$P_{a;b}(g) = \frac{\exp(g(x_a) - g(x_b))}{1 + \exp(g(x_b) - g(x_b))} \tag{5}$$

In the context of RankNet, $P_{a;b}(g)$ represents the probability that item a is ranked higher than item b given the function g, which is the neural network model in this scenario. The scores $g(x_a)$ and $g(x_b)$ are the outputs of the neural network for items a and b respectively. The equation applies the logistic sigmoid function to the difference in scores, which mAPs the raw score difference into the $(0, 1)$ interval, thus interpreting it as a probability.

Rather than computing a probability that one team is better than the other, our approach involves using the raw scores $g(x_a)$ and $g(x_b)$ and their difference. RankNet is tailored to determine the ordering of teams, not their exact scores. By focusing on raw scores, our model prioritizes the correct ranking sequence. This is expressed as:

$$\text{Final score} = g(x_a) - g(x_b) \tag{6}$$

We then rank the teams by aggregating these final scores. If team A outperforms team B, the score difference is added to team A's tally. Conversely, if team B is stronger, we add the score difference to team B's tally and deduct it from team A's. This approach accentuates the gap between disparate pairs and narrows it for similar ones.

The training of the model relies on the RankNet loss function, which is formulated as:

$$L(y_{\text{true}}, y_{\text{pred}}) = \log\left(1 + \exp(-y_{\text{true}} \cdot y_{\text{pred}})\right) \tag{7}$$

Here, y_{pred} is the predicted score difference from the model, and y_{true} is the true label that indicates which team is better. This discrepancy is not problematic because the logistic function, integral to the loss computation, naturally mAPs any real number to the range (0,1).

2) Siamese Neural Network: The second model developed was a Siamese Neural Network (SNN) [6]. While the SNN operates similarly to the RankNet model, it differs in terms of the loss function and the architecture of the final layer. Unlike RankNet, which outputs a direct ranking, the SNN provides a measure of similarity between team pairs. The final layer of the SNN employs a rectified linear unit (ReLU) activation function and has five output nodes rather than one. We feed the SNN with pairs of team statistics; it then assesses similarity using a distance metric, specifically the Euclidean distance in our implementation. Training was performed using the Adam optimizer for 50 epochs at a learning rate of 0.001, as detailed in Table II.

The SNN utilizes a contrastive loss function [9] that evaluates the outputs from both subnetworks. It operates on pairs of values, aiming to increase the distance between dissimilar team performances while decreasing it for similar ones. This approach maintains proximity among comparable teams. The contrastive loss function is defined as:

$$J(\theta) = \frac{1-Y}{2}(D_{a,b})^2 + \frac{Y}{2}(\max(0, m - D_{a,b}))^2 \tag{8}$$

where $J(\theta)$ is the cost function, θ is the vector of model parameters, Y the true label indicating $Y = 0$ if a team wins and $Y = 1$ when a team loses, $D_{a,b}$ is euclidean distance for the output vectors between team A (a) and team B (b).

The Euclidean distance is calculated by taking each feature pair and calculating how similar the two features are [7].

$$D_{a,b}(x_1, x_2) = \sqrt{\sum_{k=0}^{K} \left(b(x_2)_k - a(x_1)_k\right)^2} \tag{9}$$

Here, $a(x_1)$ represents the output vector of subnetwork one, and $b(x_2)$ represents the output vector of subnetwork two. The sum is taken over all n components of the output vectors, in our case it will be 5 as that is our output nodes. This distance metric quantifies the dissimilarity between x_1 and x_2 after being processed by the Siamese network. A smaller distance implies higher similarity, and conversely, a larger distance indicates greater dissimilarity.

The third model built was an XGBoost model [5]. XGBoost starts with a weak model and iteratively adds weaker models to overcome prediction flaws to generate a strong model. The pairwise ranking XGBoost model was used with the logistic loss function. After hyperparameter tuning the maximum depth of the tree calculated was 7 nodes, the learning rate was 0.47, and the fraction of training samples used to train each pair was 0.6221, the fraction of features used per tree was 0.9053 and the minimum child weight(hessian) was 9 as seen in Table III.

The XGBoost training objective combines a loss function l with a regularization term Ω, where the loss function, in this case, is the binary logistic loss (log loss) [18]:

$$\text{Objective} = \sum_{(i,j)} l(y_{ij}, \hat{y}_{ij}) + \Omega(\theta) \tag{10}$$

$$\Omega(\theta) = \gamma T + \frac{1}{2}\lambda\|\theta\|^2 \tag{11}$$

where T is the number of leaves in the tree, θ are the parameters of the model, γ which in our case is the complexity control on the number of leaves which was found to be 4.09, and λ is the L2 regularization term on the weights which was found to be 0.314.

Due to the use of binary logistic loss, the true labels and predicted labels are consistent. The predicted scores range from 0-1 indicating that one team has a higher chance of ranking above its opposition. The log loss for each pair of teams $\hat{y}_{ij}$ with the true label y_{ij} is given by:

$$l(y_{ij}, \hat{y}_{ij}) = -y_{ij}\log(\sigma(\hat{y}_{ij})) - (1-y_{ij})\log(1-\sigma(\hat{y}_{ij})) \tag{12}$$

XGBoost uses second-order optimization, so it calculates both the gradient (first-order derivative) and the Hessian (second-order derivative) which was found to be 9 of the loss function [20]:

$$g_{ij} = \frac{\partial l(y_{ij}, \hat{y}_{ij})}{\partial \hat{y}_{ij}} \tag{13}$$

$$h_{ij} = \frac{\partial^2 l(y_{ij}, \hat{y}_{ij})}{\partial \hat{y}_{ij}^2} \tag{14}$$

In each iteration, XGBoost updates the model using the gradient and Hessian. For a given tree structure q, the optimal weight for leaf t is:

$$w_t^* = -\frac{\sum_{i \in I_t} g_i}{\sum_{i \in I_t} h_i + \lambda} \tag{15}$$

where I_t is the set of instances in leaf t.

This allows us to get a prediction of whether one team is better than another. The same concept is used in the RankNet model where we add the score to Team A if it is predicted to be better and subtract the score from Team B.

3) SNN+XGBoost: The fourth model developed is the SNN+XGBoost model [32], which employs a transfer learning approach. This involves preserving the activation values from the rectified linear output nodes of the SNN. Subsequently, these values are supplemented with team pairs to serve as new features. Both the RankNet and SNN models used an input of 51 features per team, comprising 31 features plus a one-hot encoded representation of team names for each subnetwork. The XGBoost model then takes these 51 features for each team and enhances them with the activation values giving us a feature space of 102, facilitating feature embedding. This approach has seen to have good results in the field of NBA as seen in Section II where an SNN excels at learning feature embeddings and the XGBoost model excels at taking pairs of teams and giving a list-wise ranking directly. The operational mechanism of this model mirrors that of the aforementioned XGBoost model. It adopts the same principle as the RankNet model, where a victory for Team A increases its score while decreasing Team B's score accordingly.

The final model employed is a Transformer-based approach [10]. The structure of the Transformer model consists of three key layers, which are elaborated in Section II. Through the process of hyperparameter tuning, the model was configured to train with two heads, enabling the attention mechanism to separately focus on the statistics for each of the two teams involved. Each Transformer block includes a feedforward neural network, which features two linear transformations with a ReLU activation function in between. The hidden layer within this network has a dimension size of 256. We utilized a singular Transformer block that integrates the self-attention mechanism and the feedforward network. Convergence of the model was observed after 20 epochs, a result that is documented in Table IV.

The input layer receives the same data as seen in our previous models. It receives pairs of teams statistics. This is then passed through the transformer block which contains the Multi-Head Self attention and Feed Forward Network (FFN). The self-attention mechanism computes the scores for the different heads of inputs. Essentially it computes the scores for each team in the pair. From equation 3, this is calculated internally by TensorFlow 'MultiHeadAttention' built in function [8].

The Feed Forward Network is represented by:

$$\text{FFN}(x) = \max(0, xW_1 + b_1)W_2 + b_2 \tag{16}$$

Where W_1 and W_2 are the weights, and b_1 and b_2 are the biases of the respective layers.

The output layer uses a softmax function [15] to get the scores for each team. These scores are then aggregated to get a final score for each team and ranked in descending order using its probability of winning.

After this phase, we begin the re-ranking process [19]. As the Premier League model is trained across multiple seasons, it's necessary to account for the varying relevance of each season, with more recent seasons being deemed more significant than older ones. To address this, a formula was developed that not only incorporates historical data but also reflects the actual rankings of teams within each season. The weight assigned to each season's rankings decreases according to the number of years that have elapsed since then. For instance, when training our model on 6 seasons, the weightings are designed to increase linearly from 1 for the oldest season to 6 for the most recent one. This approach signifies that the latest season's data is considered six times more influential than that of the first season, resulting in what we refer to as normalized standings (ns). Once this step is completed, we proceed to calculate the adjusted scores (as):

$$\text{a}s_{\text{team\_id}} = w_1 \cdot \text{a}p_{\text{team\_id}} + w_2 \cdot \text{n}s_{\text{team\_id}} \tag{17}$$

The adjusted score (as) for a given team, indexed by *team_id*, is computed as a weighted sum where $w_1 = 0.7$ represents the weight for the average probabilities (ap), which captures the expected performance based on historical data, and $w_2 = 0.3$ is the weight for normalized standings (ns), indicating the team's ranking adjusted for the passage of time. The as thus represents a combination of each team's trend over time (ap) and its adjusted recent performance (ns), with these weights signifying the relative significance of each component in the determination of the final score.

C. Metrics

The predicted ranks of each test set were compared to each other using the metrics Mean Average Precision(mAP) as well as Normalized Discounted Cumulative Gain(NDCG). The metrics compared each test season's predicted rankings to the actual rankings of that season. The reason why these metrics were chosen was because they reflect the models' ability to not only predict the correct teams in the top positions but also prioritize to them correctly.

1) Normalized Discounted Cumulative Gain(NDCG): NDCG (Normalized Discounted Cumulative Gain) assesses the quality of rankings by considering varying degrees of relevance [28]. In the context of the Premier League, this relevance takes on particular significance since the top 4 teams qualify for the Champions League, while the bottom 3 face relegation. As a result, the top 4 teams can be seen as the most relevant but the entire standings also need to be evaluated, hence we test the degree of relevance at k = 4 and k = 20. We determine the relevance of a ranking by assigning the highest relevance score (rel) of 1 when a team's predicted rank perfectly aligns with its actual rank, and this relevance score diminishes as the discrepancy between the predicted and actual ranks grows. The Discounted Cumulative Gain (DCG) is given as:

$$DCG_k = \sum_{i=1}^{k} \frac{rel_i}{\log_2(i+1)} \tag{18}$$

k is evaluating the quality up until a particular ranking. The term $\log_2(i+1)$ acts as a discounting factor to reduce the weight of relevance scores at lower ranks, embodying the principle that correct predictions at higher ranks are more valuable than those at lower ranks. Here, rel_i is the relevance score assigned to the item at the i^{th} position.

$$\text{rel}_i = \frac{1}{|\text{predicted\_rank}_i - \text{actual\_rank}_i| + 1} \tag{19}$$

rel_i is the relevance score for the item at position i. Predicted_rank$_i$ is the rank predicted by the model for the team at position i. Actual_rank$_i$ is the actual rank of the team at position i. The absolute difference |predicted_rank$_i$actual_rank$_i$| is taken to account for the magnitude of the ranking error irrespective of the direction. Adding 1 to the denominator is important because teams are indexed from 0-19 and adding the 1 allows the teams to be ranked in their correct position.

Finally, the NDCG score is calculated.

$$NDCG_k = \frac{DCG_k}{IDCG_k} \tag{20}$$

Here, DCG_k represents the Discounted Cumulative Gain at rank k. The denominator, $IDCG_k$, is the DCG value for an ideal ranking where teams are perfectly ordered by their relevance, acting as a benchmark. By comparing DCG_k to $IDCG_k$, NDCG conveys how closely a given ranking approximates the ideal case, with a higher NDCG score indicating a better ranking quality.

2) Mean Average Precision (mAP): Mean average precision [21] is a performance metric used to evaluate the performance of ranking algorithms to see if a teams predicted position is close to its actual ranking. When measuring the importance of a team in a certain position, two important metrics are *Mean Average Precision (mAP)* and *Average Precision (AP)*. The formula for mAP is given by:

$$\text{mAP} = \frac{1}{k} \sum_{k=1}^{K} AP_k \tag{21}$$

where K represents the degree of relevance we are taking into account and the same concept to NDCG. On the other hand, the formula for AP for a specific instance i is:

$$\text{AP}_i = \frac{1}{K} \sum_{k=1}^{K} \frac{\text{True positives}_k}{\text{True positives}_k + \text{False positives}} \tag{22}$$

Average Precision (AP) combines precision and recall. It is the true positives in the top k ranked teams. True positives (TP) and false positives (FP) are calculated as:

$$TP_k = \sum_{i=1}^{k} \mathbf{1}\{rank(actual_i) < k\} \tag{23}$$

$$FP_k = k - TP_k \tag{24}$$

The indicator function is denoted as $\mathbf{1}\{\text{condition}\}$, which returns 1 if the condition is true, and 0 otherwise. Rank$(actual_i)$ denotes the actual rank of the i-th team, and k is a specified rank threshold.

D. Experimental Setup

Our initial setup involves establishing a baseline model, specifically a RankNet model [3], to serve as a point of reference for evaluating subsequent models. This baseline consists of a dual-subnetwork architecture that processes input statistics for home and away teams. The uniqueness of the RankNet model lies in its specialized loss function and the employment of a linear activation function, culminating in a single output node. This output quantitatively assesses the comparative strengths of teams, with score differentials affecting the cumulative tallies and resulting in a ranking order. The final rankings derived from this model are then assessed against actual outcomes using metrics such as NDCG and mAP.

Building on this, we then engineered a Siamese Neural Network (SNN) model [6]. Structurally akin to RankNet, the SNN differentiates itself by utilizing a Rectified Linear Activation (ReLU) function, having 5 output nodes and making use of a contrastive loss function. It generates similarity scores, which influence the ranking process by a defined threshold, thus clustering or dispersing team scores accordingly.

Subsequently, we employed an XGBoost algorithm [5], functioning as a pairwise ranker that makes use of a logistic loss function [18] in a manner that leverages the advanced learning mechanisms intrinsic to XGBoost substantiated in Section III-B to produce the final league standings.

An approach was realized by combining the SNN and XGBoost models. This combination integrates the activation values from the SNN as supplementary features within the XGBoost model, enhancing the input data and potentially enhancing predictive accuracy.

Finally, a transformer-based model was developed [19], utilizing a self-attention mechanism [27] and categorical Cross-Entropy [34] for the computation of team scores, which are aggregated to form preliminary rankings. These rankings undergo further refinement based on historical data, as elaborated in Section III-B.

Each model's efficacy was gauged by its ability to compute mAP and NDCG for the top 4 and top 20 ranks. The SNN+XGBoost hybrid model aimed to ascertain whether the inclusion of SNN activation values could outperform the standalone performances of both SNN and XGBoost models. Furthermore, we investigated whether the transformer-based model could outperform the established ranking algorithms specifically in football.

IV. Results and Analysis

The overview of this section includes the results found from evaluating our team standings (Section IV-A), comparing the results of the different approaches (Section IV-B), explaining future improvements (Section IV-C) and finally the limitations (Section IV-D) of the methodology.

A. Results

The results of our analysis are presented in Tables V and VI, where each model's performance was averaged over 20 runs. An average score was calculated for each season and subsequently averaged across all seasons. Table V displays the NDCG and mAP scores for our five test seasons from 2018/19 to 2022/23 at relevance levels k=4 and k=20. Table VI summarizes the average scores across these five seasons for the same relevance levels. Trends in mAP and NDCG scores over the five test seasons at both relevance levels are depicted in Figures 1 and 2.

B. Analysis

The comparison of different models shows that while no single model consistently outperformed the others, all models exceeded the baseline RankNet's performance. According to Table VI, RankNet's mAP and NDCG scores were 0.58 and 0.63, respectively. The SNN model, operating solely on a contrastive loss function, closely matched RankNet's performance, due to the influence of structural similarities and the divergence in loss function and output layer. The SNN model achieved an NDCG score of 0.66 and a mAP score of 0.72.

Across the test seasons, the XGBoost model stood out with the highest average scores at both relevance levels, indicating the predictive accuracy for teams likely to qualify for the Champions League where it achieved an average mAP score of 0.71 and an NDCG score of 0.88 at k=4. Notably, its scores of 0.91 and 0.79 at k=20 suggest a robust ability to predict the complete league standings accurately.

mAP scores were generally lower than NDCG scores, which can be attributed to mAP being a more stringent metric, as shown in equation 21. We used a more relaxed metric to calculate true positives and negatives as seen in equation 23. The higher NDCG scores indicate that our models were more successful in accurately predicting the top teams' positions, as this metric penalizes errors more severely lower in the league standings.

The 2018/19 season was the third-highest-performing season across all models. This was attributed to 3 of the teams from the previous season finishing in the top 4 in the 2018/19 season. Manchester City maintained the top position in both seasons. The SNN+XGBoost model outperformed others at both levels of relevance, achieving an NDCG score of 0.95 and an mAP score of 0.79. It accurately predicted Manchester City and Liverpool in the first and second positions, respectively, mirroring the actual standings. However, it incorrectly forecasted Manchester United for the third position due to their second-place finish in the previous season, whereas they actually fell to sixth in this season.

The 2019/20 season exhibited a noticeable decline in performance for all models, as illustrated in Figures 1 and 2. This was primarily because the models did not predict the season's actual champion correctly, having been trained on data where Manchester City emerged as the winner in the preceding four seasons. Conversely, Liverpool claimed the title that year, impacting the models' metrics.

The 2020/21 and 2021/22 seasons were our most successful, with 2021/22 being the top-performing season across all models. It was during this season that the transformer-based model achieved the highest NDCG scores of 0.98 and 0.99 at k=4 and k=20, respectively. This model precisely predicted the positions of Manchester City, Liverpool, and Chelsea, only misplacing Tottenham Hotspur and Arsenal in fourth and fifth positions. The XGBoost model was a close contender, securing higher mAP scores of 0.85 and 0.86 at k=4 and k=20, respectively. Despite correctly identifying the top team, it inaccurately predicted the second position. However, the teams were predicted closer to their actual positions, which is why their scores were marginally higher than those of the transformer-based model, at 0.84 and 0.86.

The 2022/23 season also showed a decrease in performance following the peak season for all models. This was due to unexpected outcomes, such as Arsenal securing a top-four finish for the first time since the 2015/16 season, Newcastle's rise following a change in ownership, and Liverpool's surprising fall from the top four.

TABLE V
MAP AND NDCG SCORES ACROSS ALL TEST SEASONS

		2018/19		2019/20		2020/21		2021/22		2022/23	
		4	20	4	20	4	20	4	20	4	20
Rank-net (baseline)	NDCG	**0.64**	**0.73**	0.30	0.52	0.35	0.50	0.46	0.58	0.40	0.55
	mAP	**0.54**	**0.73**	0.35	0.63	0.46	0.62	0.25	0.60	0.25	0.56
XGBoost	NDCG	0.82	0.93	0.78	0.79	0.95	0.96	**0.97**	**0.99**	0.88	0.90
	mAP	0.69	0.85	0.54	0.77	0.79	0.81	**0.91**	**0.86**	0.63	0.67
XGB+SNN	NDCG	0.95	0.96	0.74	0.75	0.92	0.93	**0.97**	**0.98**	0.75	0.88
	mAP	0.79	0.87	0.48	0.72	0.74	0.82	**0.86**	**0.85**	0.54	0.68
SNN only	NDCG	0.55	0.68	0.54	0.57	0.61	0.65	0.55	0.60	**0.78**	**0.80**
	mAP	0.33	0.75	0.35	0.69	0.45	0.75	0.33	0.73	**0.61**	**0.69**
Transformer-based model	NDCG	0.82	0.93	0.67	0.73	0.94	0.94	**0.98**	**0.99**	0.81	0.88
	mAP	0.74	0.86	0.44	0.73	0.74	0.82	**0.84**	**0.86**	0.54	0.67

TABLE VI
AVERAGE MAP AND NDCG SCORES FOR $k = 4$ AND $k = 20$ ACROSS ALL TEST SEASONS

		Average Scores	
		$k = 4$	$k = 20$
Rank-net	NDCG	0.43	0.58
	mAP	0.37	0.63
XGBoost	NDCG	**0.88**	**0.91**
	mAP	**0.71**	**0.79**
XGB+SNN	NDCG	0.87	0.90
	mAP	0.68	**0.79**
SNN only	NDCG	0.61	0.66
	mAP	0.45	0.72
Transformer-based model	NDCG	0.84	0.89
	mAP	0.66	**0.79**

Regarding individual model performance, the SNN model lagged, particularly when tasked with predicting full list-wise rankings. This shortfall is attributed to its output reflecting team similarities rather than definitive rankings. However, in its best season (2022/23), the SNN model achieved its highest mAP and NDCG scores of 0.69 and 0.80, respectively. Conversely, the standalone XGBoost model excelled, notably in the 2021/22 season with an NDCG of 0.99 and a mAP of 0.86 at k=20, and impressive k=4 scores with an NDCG of 0.97 and a mAP of 0.91. This indicates it accurately predicted all the teams in the top 4 while also predicting the first and second position correctly, but third and fourth place were switched. When supplemented with SNN activation values, the SNN+XGBoost model significantly outperformed the sole SNN model and achieved comparable, albeit slightly lower, average scores than the standalone XGBoost model, with an average NDCG of 0.90 and mAP of 0.79.

Finally, the transformer-based model surpassed both the baseline RankNet and the SNN model, attaining an average NDCG of 0.89 and a mAP of 0.79 at k=20. This performance was corroborated by Table VI. Notably, the transformer-based model had its most successful season in 2021/22, as evidenced in Table V, achieving an NDCG of 0.98 at k=4 and 0.99 at k=20, beating the XGBoost model in predicting the exact positions of the top three teams.

C. Future improvements

Our model's labels or targets are determined by the relative positions of teams. For example, if one team finishes above another, it gets a label of 1 to indicate a win. This method could be refined by using the actual scores from games over 11 seasons to make our target values more accurate.

Including external data in the Premier League, as shown in previous studies [16], can influence the final league standings. Information on transfer expenditures, club transactions, and managerial changes has been linked to higher predictive accuracy of the NDCG score with transfer learning [29].

When predicting standings in basketball and rugby league, using triplet loss has yielded more accurate results than contrastive loss [32]. If we switch our loss function to triplet loss in our SNN, the activation values relayed to the XGBoost model would carry more relevant information about a teams rank rather than the similarity between teams [9]. This approach helps to better reflect the teams' relative rankings, ensuring that stronger teams are distinctly separated from weaker ones.

D. Limitations

Although the findings presented in this research paper are promising, it's important to acknowledge some limitations. The methodology employs a pairwise approach, where two teams are compared directly to predict their standings in the league. This method may not capture the full complexity of league dynamics, as it only considers teams in isolation of

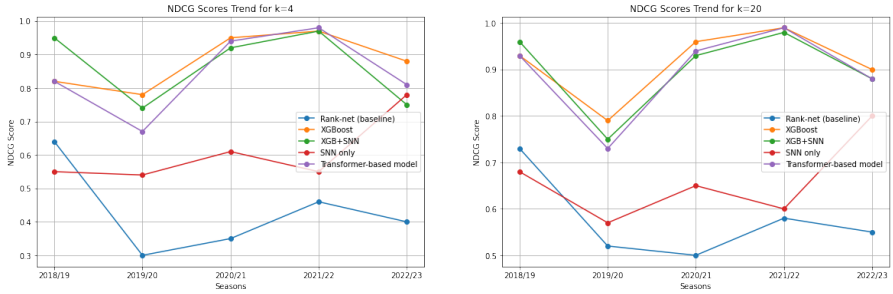

Fig. 1. NDCG scores at different degrees of relevance

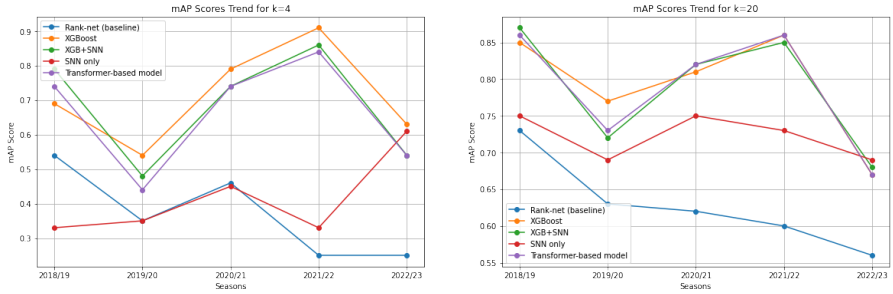

Fig. 2. mAP scores at different degrees of relevance

their direct competitors.

A list-wise approach might yield improved results [3]. This is because, while there is significant variability in a team's final standing, the fluctuations in their performance throughout the season are even more pronounced. The pairwise approach might overlook the impact of such performance changes over time, whereas a list-wise method could take into account the broader context of all the teams' performances in relation to each other, offering a more holistic and perhaps more accurate prediction of league standings.

V. Conclusion

The final standings of the English Premier League were predicted using several models: a RankNet, a Siamese Neural Network (SNN), Gradient Boosting methods, a combination of SNN and XGBoost, and a Transformer-based model. The Premier League is not only the most physically demanding soccer league in the world but also the toughest to predict.

All models yielded satisfactory results. On average, the XGBoost model performed the best, achieving an NDCG score of 0.91 and an mAP score of 0.79. The SNN model, however, was the least satisfactory, with an mAP score of 0.45 and an NDCG score of 0.61.

Combining the SNN model with XGBoost provided valuable insights into the use of activation values. This combined approach outperformed the standalone SNN model in every season and at every level of relevance. It achieved the same average mAP as the standalone XGBoost model at 0.79. Additionally, it surpassed XGBoost in the 2018/19 season with an NDCG score of 0.95 at k=4 and 0.96 at k=20, compared to XGBoost's scores of 0.82 and 0.93, respectively. This supports our hypothesis that including activation values can yield insightful predictions.

The use of a transformer-based model was also significant. It exceeded the performance of both the RankNet and SNN models across all metrics and seasons. It shared the highest average mAP score of 0.79 with the XGBoost and SNN+XGBoost models. In the 2021/22 season, it achieved a score of 0.98 at k=4, accurately predicting three out of the top four teams' positions, with only the fourth and fifth positions swapped.

Notably, the 2021/22 season was the best across all models due to its similarity to the preceding season. The following season saw a downturn due to several factors, including new ownership at Newcastle, Arsenal breaking into the top four after five years, and Liverpool falling out of the top four. Future work could improve model accuracy by incorporating external data that impact individual teams [16].

Significant findings include the use of an SNN combined with XGBoost to predict football standings and the adaptation of a transformer-based model, traditionally used in recommender systems, for football ranking predictions.

In conclusion, the models and results effectively address our research questions. This type of machine learning could be utilized by sports betting companies to predict the final Premier League standings accurately. We've built on previous research to contribute to the sports prediction market, and these

methods can be further explored across various sports.

REFERENCES

[1] Ratnadip Adhikari and R. K. Agrawal. An introductory study on time series modeling and forecasting. *arXiv preprint arXiv:1302.6613*, 2013.

[2] Punam Bedi, Neha Gupta, and Vinita Jindal. I-siamids: an improved siam-ids for handling class imbalance in network-based intrusion detection systems. *Applied Intelligence*, 51:1133–1151, 2021.

[3] Zhe Cao, Tao Qin, Tie-Yan Liu, Ming-Feng Tsai, and Hang Li. Learning to rank: from pairwise approach to listwise approach. In *Proceedings of the 24th international conference on Machine learning*, pages 129–136, 2007.

[4] Vítor Cerqueira, Luís Torgo, and Igor Mozetic. Evaluating time series forecasting models: An empirical study on performance estimation methods. *Machine Learning*, pages 1–28, 2019. Available online.

[5] Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pages 785–794, 2016.

[6] Davide Chicco. Siamese neural networks: An overview. *Artificial neural networks*, pages 73–94, 2021.

[7] Kimberly L Elmore and Michael B Richman. Euclidean distance as a similarity metric for principal component analysis. *Monthly weather review*, 129(3):540–549, 2001.

[8] Ivan Fung and Brian Mak. Multi-head attention for end-to-end neural machine translation. In *2018 11th International Symposium on Chinese Spoken Language Processing (ISCSLP)*, pages 250–254. IEEE, 2018.

[9] Benyamin Ghojogh, Milad Sikaroudi, Sobhan Shafiei, Hamid R Tizhoosh, Fakhri Karray, and Mark Crowley. Fisher discriminant triplet and contrastive losses for training siamese networks. In *2020 international joint conference on neural networks (IJCNN)*, pages 1–7. IEEE, 2020.

[10] Kai Han, An Xiao, Enhua Wu, Jianyuan Guo, Chunjing Xu, and Yunhe Wang. Transformer in transformer. *Advances in Neural Information Processing Systems*, 34:15908–15919, 2021.

[11] Thorsten Joachims. Optimizing search engines using clickthrough data. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 133–142, 2002.

[12] Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. Lightgbm: A highly efficient gradient boosting decision tree. *Advances in neural information processing systems*, 30, 2017.

[13] Amy N Langville and Carl D Meyer. Deeper inside pagerank. *Internet Mathematics*, 1(3):335–380, 2004.

[14] Yuesen Li, Runqing Ma, Bruno Gonçalves, Bingnan Gong, Yixiong Cui, and Yanfei Shen. Data-driven team ranking and match performance analysis in chinese football super league. *Chaos, Solitons & Fractals*, 141:110330, 2020.

[15] Xuezhi Liang, Xiaobo Wang, Zhen Lei, Shengcai Liao, and Stan Z Li. Soft-margin softmax for deep classification. In *International Conference on Neural Information Processing*, pages 413–421. Springer, 2017.

[16] Habeebullah Manack and Terence L Van Zyl. Deep similarity learning for soccer team ranking. In *2020 IEEE 23rd International Conference on Information Fusion (FUSION)*, pages 1–7. IEEE, 2020.

[17] Björn Münstermann and Tim Weitzel. What is process standardization? 2008.

[18] Amichai Painsky and Gregory Wornell. On the universality of the logistic loss function. In *2018 IEEE International Symposium on Information Theory (ISIT)*, pages 936–940. IEEE, 2018.

[19] Changhua Pei, Yi Zhang, Yongfeng Zhang, Fei Sun, Xiao Lin, Hanxiao Sun, Jian Wu, Peng Jiang, Junfeng Ge, Wenwu Ou, et al. Personalized re-ranking for recommendation. In *Proceedings of the 13th ACM conference on recommender systems*, pages 3–11, 2019.

[20] Sayan Putatunda and Kiran Rama. A comparative analysis of hyperopt as against other approaches for hyper-parameter optimization of xgboost. In *Proceedings of the 2018 international conference on signal processing and machine learning*, pages 6–10, 2018.

[21] Jerome Revaud, Jon Almazán, Rafael S Rezende, and Cesar Roberto de Souza. Learning with average precision: Training image retrieval with a listwise loss. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5107–5116, 2019.

[22] Kaushik Roy. Tutorial-shodhguru labs: Optimization and hyperparameter tuning for neural networks. 2023.

[23] Peter Shaw, Jakob Uszkoreit, and Ashish Vaswani. Self-attention with relative position representations. *arXiv preprint arXiv:1803.02155*, 2018.

[24] Jian Shi and Xin-Yu Tian. Learning to rank sports teams on a graph. *Applied Sciences*, 10(17):5833, 2020.

[25] Transfermarkt. Transfermarkt: Football database, 2023. Accessed: 2023-11-09.

[26] Abdullah Erdal Tümer, Zeki Akyıldız, Aytek Hikmet Güler, Esat Kaan Saka, Riccardo Ievoli, Lucio Palazzo, and Filipe Manuel Clemente. Prediction of soccer clubs' league rankings by machine learning methods: The case of turkish super league. *Proceedings of the Institution of Mechanical Engineers, Part P: Journal of Sports Engineering and Technology*, page 17543371221140492, 2022.

[27] Jesse Vig. A multiscale visualization of attention in the transformer model. *arXiv preprint arXiv:1906.05714*, 2019.

[28] Yining Wang, Liwei Wang, Yuanzhi Li, Di He, and Tie-Yan Liu. A theoretical analysis of ndcg type ranking measures. In *Conference on learning theory*, pages 25–54. PMLR, 2013.

[29] Karl Weiss, Taghi M Khoshgoftaar, and DingDing Wang. A survey of transfer learning. *Journal of Big data*, 3(1):1–40, 2016.

[30] WhoScored. Whoscored - football statistics and live scores, 2023. Accessed on November 5, 2023.

[31] Vincent Xia, Kavirath Jain, Akshay Krishna, and Christopher G Brinton. A network-driven methodology for sports ranking and prediction. In *2018 52nd Annual Conference on Information Sciences and Systems (CISS)*, pages 1–6. IEEE, 2018.

[32] Daniel Yazbek, Jonathan Sandile Sibindi, and Terence L Van Zyl. Deep similarity learning for sports team ranking. *arXiv preprint arXiv:2103.13736*, 2021.

[33] Bayya Yegnanarayana. *Artificial neural networks*. PHI Learning Pvt. Ltd., 2009.

[34] Zhilu Zhang and Mert Sabuncu. Generalized cross entropy loss for training deep neural networks with noisy labels. *Advances in neural information processing systems*, 31, 2018.