CS224W Final Project Report

Player Rankings and Outcome Prediction in Tournaments

Nopphon Siranart, Karen Truong, Vincent-Pierre Berges

1 Introduction

In a tournament where individuals or teams are randomly drawn to compete with others, predicting the outcome of a game between contestants who have yet to play against each other based on historical results is a challenging problem. Typical methods used include ranking players by the total points or percentage of game wins. This way of predicting is generally inaccurate as it is likely that top players in the ranking might have only faced weak opponents. In this project, we will apply network analysis techniques and methodologies to provide a better way of ranking the players and predicting the outcome in any kind of tournament. This will give us a better understanding of the relationship and superiority among the players and will be useful in many applications such as calculating fair match betting odds and player matching recommendation.

2 Problem Formulation

We formulate the problem of predicting the outcomes as a graph network prediction problem by constructing a weighted directed graph to represent results of games between two players in a tournament. The nodes represent the players and the edges represent the outcomes of games between A and B. In a chess tournament, node A has an edge with weight 1 directed to node B if player A lost that match to B. In case of a tie, there will be two directional edges connecting A and B, one in each direction, with a half weight each.

3 Prior Work

There have been several studies performed on ranking algorithm analysis. In "Generalizing Googles PageRank to Rank National Football League Teams," Govan et. al discuss different ranking algorithms for nodes in a graph, including a simple ordering by win percentage and a PageRank approach [1]. "BeatPower" is also another method that is widely used to rank sport teams by looking at the number of winning, losing and total relationships/paths, both directly and indirectly through other opponents in the network, and converts these stats into a team score [2]. Another popular ranking algorithm applied over 1-vs.-1 games is the Elo Rating System in which the winning player takes points from the loser proportional to the difference in their rating, such that a win by the underdog with presumably lower ranking results in taking more points than if he/she was actually the favored victor according to prior rankings [3]. In "Ties in Paired-Comparison Experiments: A Generalization of the Bradley-Terry Model," Rao and Kupper provide us a way to generate a probability distribution of the outcomes that, given a ranking of the players, allows for draws by introducing an additional parameter that serves as a 'threshold' in comparing the ranks of the two players to discern between a win/loss and a draw [4].

Most prior work only presents ways to rank teams without showing how to use them to predict the outcome, especially when a draw can happen, which is a very challenging problem. Although there are previous efforts to develop models when a draw is possible, none of them have a concrete way that show explicitly how to evaluate the performance of the models or make comparisons between them.

4 Data

As we want our methods and algorithms to be able to apply in any kind of tournament, we consider three different datasets - chess, soccer, and basketball. Each of them represents a tournament with different kinds of characteristics as shown in the table below.

Characteristics	Chess	Soccer	Basketball
Large Graph	Yes	No	No
Draws allowed	Yes	Yes	No
Score	No	Yes	Yes
Home Advantage	Yes	Yes	Yes
Betting Odds	No	Yes	No

- **Chess -** Real historical datasets provided by a world chess federation (FIDE) available on Kaggle [1]. The dataset is already provided in a clear format, where each row represents a match and contains a gameID, two player IDs, month, and result. The network contains 54,000 chess players (nodes) spanning an 11-year period and over 1.84 million games (edges). We maintain some metadata about each match on its corresponding edge in the network, including who played as the white pieces and which month this match occurred. There are three possible outcomes for a chess game - win, loss and draw, as seen from the perspective of the white player. From the data, we see that a white player who plays first has a slight advantage. The distribution of the results is 39.1% win, 28.5% draw and 32.4% loss.

- **Soccer -** English football leagues historical results from 5 divisions from 2014 to 2016. [2]. The dataset contains 149 teams and 3004 matches. The distribution of the results is 43.5% win, 26.5% Draw, 30.0% Loss. This dataset also provides metadata for each match, including its historical fair betting odds. This can be used as a benchmark to compare our outcome probability distribution against what bookmakers expect.

- **Basketball -** NCAA men's basketball Division I historical game outcomes from the 2015 and 2016 seasons[3]. The dataset contains 364 teams and 4801 matches. There is no draw in basketball and the distribution of the results is 63.9% win, 36.1% loss.

5 Description of Algorithms

5.1 Ranking-based Approaches

We consider multiple ways of generating the ranking for players. Once we get the ranking, we will use this to determine the probability of possible results and predict the outcomes by choosing the one with the highest probability.

5.1.1 Ranking System

- **Baseline -** For a simple baseline, we set the ranking score for each player to be the percentage of game wins from past match results. This is a general approach typically used in the real world.

- **PageRank -** Another intuitive approach we took was a variation of PageRank, a ranking system designed to give a node higher rank if it is linked to by other high ranking nodes. To calculate the rankings of each player in a vector $\mathbf{r}$ by power iteration, we begin with $\mathbf{r} = [\frac{1}{|N|}, \frac{1}{|N|}, \ldots, \frac{1}{|N|}]^T$, where N is the number of nodes or players in our network. For each match, we draw an edge from the loser to the winner. In cases of draws, we draw two edges: player1 $\rightarrow$ player2, and player2 $\rightarrow$ player1. The PageRank for player j is calculated as:

$$r_j = \sum_{i \rightarrow j} \beta \frac{r_i}{d_i} + (1 - \beta)\frac{1}{N} \tag{1}$$

where d_i is the outdegree of node i, and $1 - \beta$ represents a teleportation or probability of some random player winning. In a given iteration of the PageRank computation,

$$\mathbf{r}^{(t+1)} = M\mathbf{r}^{(t)} \tag{2}$$

where M is the adjacency matrix representing who has played whom in our game history. The iterations continue until the rank reaches a steady state such that $|\mathbf{r}^{(t+1)} - \mathbf{r}^{(t)}| < \epsilon$.

[1] https://www.kaggle.com/c/ChessRatings2/data
[2] http://www.football-data.co.uk/englandm.php
[3] https://www.kaggle.com/c/march-machine-learning-mania-2016/data

In the application of the original PageRank to our dataset, edges representing draws are equally weighted to those derived from wins, which is not the best configuration. Consequently, we introduced a weighted PageRanking scheme, where draw edges were given half the weight of a win edge. Weighing more recent matches as more valuable and more reflective of a player's skill than old match results, each edge weight is further scaled by month using a sigmoid such that $match_i$ is scaled by $\frac{1}{1+e^{-\text{monthID}_i}}$ in which case more recent matches have a higher monthID and are therefore dampened less. The outdegree of a node is then determined as the sum of the weights of outgoing edges for the PageRank computation.

- **BeatPower -** The situation where $A \to B \to C \to A$ happens very often in the chess world. The issue is that this configuration does not give any indication on who might be the strongest of the three. BeatPower is an algorithm that removes the oriented cycles in the graph before ranking the players [2].
  In our simplified version of the BeatPower algorithm, we remove the cycles of length 2 and length 3, i.e. $A \to B \to A$ and $A \to B \to C \to A$ situations. We iterate through the edges and when a cycle is found, we remove all of the edges that compose it. For computation time reasons, we limited ourselves to cycles of length 3 or less. This new graph is called a BeatGraph.
  Once the cycles are removed, we can rank the players based on their number of victories (beatWin) and losses (beatLoss) in the BeatGraph. The ranking follows the equation :

$$\forall p \ : \quad beatPower(p) = \frac{beatWin(p) - beatLoss(p)}{beatWin(p) + beatLoss(p) + 1} \tag{3}$$

  The beatPower of each player is a value between -1 and 1. The higher the beatPower, the stronger the player. Similar to PageRank, we make a prediction by comparing the beatPower of the two involved players. The beatGraph can also be used to compute another PageRank based on this new graph to improve performance.

- **TrueSkill -** We also configured an existing two-player ranking system called TrueSkill to our datasets. In TrueSkill, each player is modeled by a Gaussian distribution, where μ is the average measure of his/her skill and σ is a degree of uncertainty in that skill measurement [5]. All players start off with the same skill. As we process edges of the graph, we update the two players' skills and uncertainties using Bayesian inference. Similar to the Elo rating system, the update depends on how "surprising" the outcome is. For a given edge between player A and player B, if A beats B (i.e. an edge from B to A), we would make the following updates:

$$\begin{gathered} \mu_A \Leftarrow \mu_A + \frac{\sigma_A^2}{c} \cdot v\left(\frac{\mu_A - \mu_B}{c}, \frac{\epsilon}{c}\right), \quad \mu_B \Leftarrow \mu_B - \frac{\sigma_B^2}{c} \cdot v\left(\frac{\mu_A - \mu_B}{c}, \frac{\epsilon}{c}\right) \\ \sigma_A^2 \Leftarrow \sigma_A^2\left(1 - \frac{\sigma_A^2}{c^2} \cdot w\left(\frac{\mu_A - \mu_B}{c}, \frac{\epsilon}{c}\right)\right), \quad \sigma_B^2 \Leftarrow \sigma_B^2\left(1 - \frac{\sigma_B^2}{c^2} \cdot w\left(\frac{\mu_A - \mu_B}{c}, \frac{\epsilon}{c}\right)\right) \\ c^2 = 2\beta^2 + \sigma_A^2 + \sigma_B^2 \end{gathered} \tag{4}$$

  where β^2 is the variance of the performance around the skill of each player, and w and v are multiplicative correction terms for the approximate marginals as a function of the draw margin ϵ. As the system learns about more match outcomes, its uncertainty about a player's TrueSkill decreases. If draws are possible, as in the case of soccer and chess matches, ϵ uses the likelihood of a draw to control the impact a non-drawn game has on a player's ratings. For our model, we set this draw probability to be the draw percentage of our training set.

5.1.2 Predicting Outcomes From Ranking

- **Empirical data -** We use the players' ranks to calculate the probability of each outcome. First, for each historical match, we assign it a rating by computing the difference between the ranking scores of two players. We then aggregate all the games with the same rating and see how many times these games result in a win, loss and draw. Based on these empirical results, we can fit an appropriate curve to predict the probability for each outcome given a match rating.

- **Modified Bradley-Terry -** If we simply predict that the player with the higher rating wins, then it is difficult to predict draws. To accommodate this aspect of our chess and soccer datasets, we calculated the probability of each outcome using a modified version of the Bradley-Terry model, which defines P(i beats j),

the probability that i defeats j, as $\frac{\pi_i}{\pi_i+\pi_j}$, where each π_i, π_j are variables representing the rankings/ratings of i and j, respectively. We adjust the model and introduce a term dependent on the geometric mean between the two players' ratings, so that it can predict whether i and j draw

$$P(\text{i beats j}) = \frac{\pi_i}{\pi_i + \pi_j + v\sqrt{\pi_i\pi_j}}, \quad P(\text{j beats i}) = \frac{\pi_j}{\pi_i + \pi_j + v\sqrt{\pi_i\pi_j}}, \quad P(\text{draw}) = \frac{v\sqrt{\pi_i\pi_j}}{\pi_i + \pi_j + v\sqrt{\pi_i\pi_j}} \tag{5}$$

where $v \geq 0$ is a constant of proportionality used as an index of discrimination between the three distinct outcomes. One other variation of calculating the probability of a draw was proposed by Rao & Kupper, where θ is the discriminating parameter in:

$$P(\text{i beats j}) = \frac{\pi_i}{\pi_i + \theta\pi_j}, \quad P(\text{j beats i}) = \frac{\pi_i}{\theta\pi_i + \pi_j}, \quad P(\text{draw}) = \frac{(\theta^2-1)\pi_i\pi_j}{(\theta\pi_i + \pi_j)(\pi_i + \theta\pi_j)} \tag{6}$$

However, we found that the geometric mean version consistently outperformed the Rao & Kupper configuration, so metrics below are only reported for ties according to probabilities generated from Eqn 5 with v=2. The predicted result is then determined as the scenario with the highest probability.

- **TrueSkill prediction -** As with Bradley-Terry, the prediction is chosen as the outcome with the highest likelihood, but the probability of i beating j is calculated differently. Given the TrueSkill rating of player i and player j, represented by normal distributions $\mathcal{N}(\mu_i, \sigma_i^2)$ and $\mathcal{N}(\mu_j, \sigma_j^2)$, respectively, we generate a win probability for i using the cumulative density function Φ as:

$$P(\text{i beats j}) = \Phi\left(\frac{\mu_i - \mu_j}{\sqrt{\sigma_i^2 + \sigma_j^2}}\right) \tag{7}$$

5.2 Machine Learning Approach

- **Local Properties -** This novel approach is different from the ranking based approaches as it uses machine learning on the relationships between matches and players to make a prediction. That is, we want to look specifically how two players interact with each other instead of looking at the global ranking.

  We first build a graph using past games and extract the features from it. For each new game, we predict the result by using all the previous interactions of the two players. There are three kinds of features that we use. First, we count the number of times A beats, loses or draws against B in the graph. Second, we include the number of paths from A to B of lengths 2 and 3 and their nature. We could include longer paths but because of computational limitations, we only consider paths of length less than 4. Finally, we add characteristics of each player such as the numbers of win, draw, loss, and its ranking computed from the ranking systems mentioned in the section above.

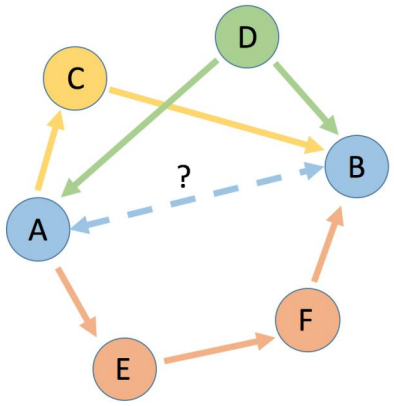

Figure 1: Local properties between node A and B

In Figure 1, if we want to make a guess on whether A beats / loses / draws against B, we can count the historical actions between A and B. In the case presented in Figure 1, there is one [loss, loss] path through C (yellow), one [win, loss] through D (green), and one [loss, loss, loss] through E and F (orange). (Remember, if an arrow points from A to B, B beats A). We store this information in a table and use machine learning on those predictors.

Once the train and test set are generated, we run various machine learning algorithms to identify patterns in the local properties of the two considered nodes. We tried adding more features such as minimum path length but the computation was taking too many resources. We use multiple models including Random Forest, Logistic Regression and Gradient Boosting. Decision Trees methods seem to give the best results.

For the fine tuning of the regression parameters, we split the train set into a train and validation set and tested a set of parameters to evaluate the performance on the validation set. The parameters that provide the best results are then used to make the prediction on the true test set.

6 Evaluation Metrics

We denote $y_i, \hat{y}_i$ to be the actual and predicted outcomes of match i respectively. Each of them belongs to the set $\{1.0, 0.5, 0.0\}$ representing win, draw and loss. We will evaluate the models using three metrics as follows.

- **Mean Difference** - This metric is calculated by taking the average of the absolute difference between the predicted labels and the actual labels. Intuitively, we want to see how far the predicted labels are from the actual labels on average. That is, if the ground truth is 1, then it is better to predict 0.5 than 0. However, this metric is equivalent to the accuracy matrix in case of a tournament with no draws such as basketball.

$$\text{Mean Difference} = \frac{1}{N}\sum_{i=1}^{N}(|y_i - \hat{y}_i|)$$

- **Macro-Averaged Precision** - This metric is calculated by taking the average of the precision for each class. Macro-averaging allows us to give equal weight to each class. Below is an example when there are three categories.

		Prediction		
		Win	Draw	Loss
Truth	Win	A	B	C
	Draw	D	E	F
	Loss	G	H	I

Table 1: Confusion Matrix of the Prediction

$$\text{Macro-Averaged Precision} = \frac{1}{3} \times \left(\frac{A}{A+B+C} + \frac{E}{D+E+F} + \frac{I}{G+H+I}\right)$$

- **Accuracy** - This metric provides us the actual accuracy i.e. how often we predict the results correctly.

$$\text{Accuracy} = \frac{1}{N}\sum_{i=1}^{N}\mathbb{1}_{\{y_i=\hat{y}_i\}}$$

Note: For soccer, we also have historical match betting odds data as well. This betting odds can be converted into an outcome probability distribution. Hence, we will compare our result with this distribution as well.

7 Results and Analysis

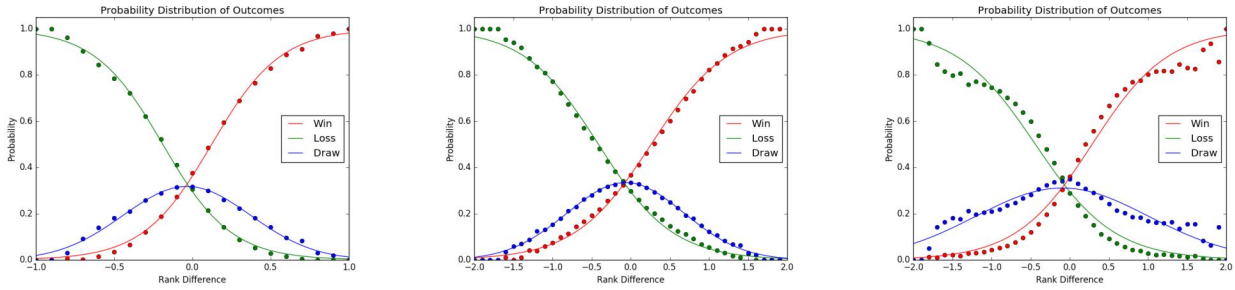

Figure 2: Chess - Probability Distribution of the Outcomes: Baseline(left), PageRank(center), BeatPower(right)

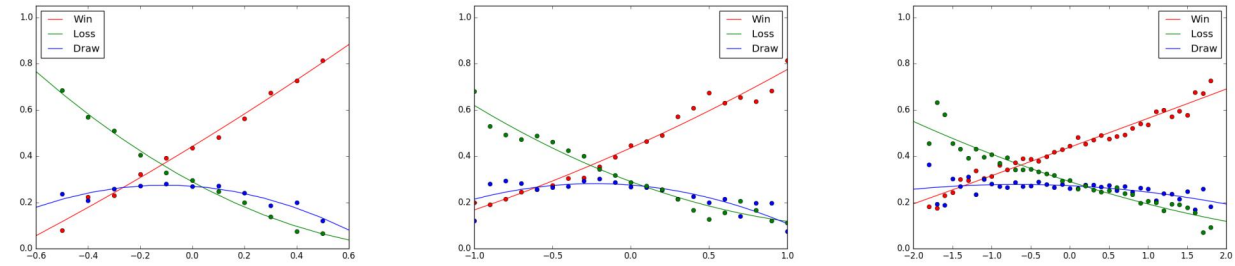

Figure 3: Soccer - Probability Distribution of the Outcomes: Baseline(left), PageRank(center), BeatPower(right)

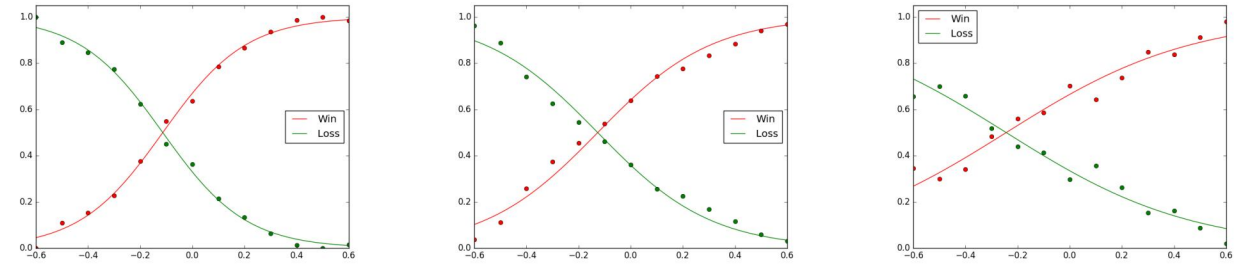

Figure 4: Basketball - Probability Distribution of the Outcomes: Baseline(left), PageRank(center), BeatPower(right)

7.1 Discussion

Figure 2, 3, and 4 show the plots of the empirical probability distribution of win, draw and loss based on the rank difference for each ranking system for chess, soccer and basketball, respectively. We use a sigmoid curve to fit the probability of win and loss, and a Gaussian function to fit the probability of draw for chess and basketball. However, the soccer outcome probabilities seem to follow a quadratic curve. We see from the plots that this is not a good way to predict the outcomes since our system only predicts a draw when it has the highest probability and that only happens when A and B have the same rank, which occurs very rarely. We also see from the graphs that for chess, a white player has a little advantage since the probabilities of win and loss are equal at the point where the rank difference is slightly below zero, whereas a home team has a much greater advantage for soccer and basketball.

Tables 2, 3, and 4 report the results of running our methods on the datasets. Models that support ties using modified Bradley-Terry probabilities perform better when we consider the mean difference because it tends to predict more draws, which also leads to small improvements in the macro-average precision. However, the accuracy does not improve much. We assume that this is because draws are very hard to predict and it occurs randomly even if two players are about the same level in the ranking.

Table 2: Chess Model Results

Method	**Mean Difference**	**Macro-Avg Precision**	**Accuracy**
Random	0.452	0.330	0.330
Baseline	0.419	0.408	0.433
Baseline + Ties	0.382	0.410	0.416
PageRank	0.370	0.455	0.483
PageRank + Ties	0.322	0.463	0.484
BeatPower	0.380	0.444	0.471
TrueSkill	0.318	0.488	0.501
Logistic Regression Local only	0.345	0.483	0.508
Random Forest Local only	0.330	0.494	0.509
Logistic Regression Local with PageRank	0.339	0.491	0.514
Random Forest Local with PageRank	0.315	0.510	0.526
Gradient Boosting with PageRank	**0.308**	**0.539**	**0.523**

Table 3: Soccer Model Results

Method	**Mean Difference**	**Macro-Avg Precision**	**Accuracy**
Random	0.465	0.333	0.333
Baseline	0.446	0.383	0.418
Baseline + Ties	**0.391**	0.417	0.428
PageRank	0.44	0.389	0.424
PageRank + Ties	0.409	0.391	0.419
BeatPower	0.412	**0.413**	**0.449**
TrueSkill	0.4	0.391	0.407
Logistic Regression Local only	0.440	0.356	0.397
Random Forest Local only	0.431	0.380	0.424
Logistic Regression Local with PageRank	0.443	0.351	0.389
Random Forest Local with PageRank	0.424	0.373	0.414
Gradient Boosting with PageRank	0.417	0.402	0.428

Table 4: Basketball Model Results

Method	**Mean Difference**	**Macro-Avg Precision**	**Accuracy**
Random	0.5	0.5	0.5
Baseline	0.236	0.774	0.764
PageRank	0.257	0.753	0.742
BeatPower	0.220	**0.792**	0.780
TrueSkill	0.229	0.778	0.771
Logistic Regression Local only	**0.210**	0.746	**0.790**
Random Forest Local only	0.229	0.737	0.771
Logistic Regression Local with PageRank	0.213	0.754	0.787
Random Forest Local with PageRank	0.219	0.748	0.781
Gradient Boost with PageRank	0.239	0.731	0.761

While our ranking models do outperform the baseline, we see that Gradient Boosting with local properties performs the best in all metrics on the chess dataset. This is because the ranking methods only make predictions based on the ranking of players, which relies on a global ranking and does not take into account the local properties and interaction of the graph. Meanwhile the machine learning approach considers the activity and relationships between players nearby the two given players, in addition to PageRank. Also, it seems decision tree models are much more suitable than linear models as there is no real linear dependency between the variables and the outcomes.

Although it performs quite poorly on chess, BeatPower gives very good results for both soccer and basketball. BeatPower is better adapted to sports networks given that they are smaller and more dense, whereas it may be that the chess dataset is not dense enough once loops and draws are removed. Given the large size of the chess dataset, if good players only play against good players, the beat scores will be biased or artificially lower, and this might also explain why BeatPower under-performs on the chess dataset, whereas the sports matches are not biased since the schedules have teams play one another regardless of skill difference. As we can see, BeatPower and Local Properties analysis rely on very different methods and properties of the tournament topology and it is important to try multiple approaches when given a new dataset.

Logistic Regression seems to be the best model for the basketball dataset and it outperforms the decision tree models. This might be because the number of features is much lower with the basketball dataset (no Draws features) so there is less overfitting.

One might notice that our algorithms performs worse on soccer than chess. We suspect that this is due to multiple reasons. First, we have a significantly smaller amount of data for soccer than chess. Unlike chess, soccer is a multi-player sport, so the performance of a team does not only depend on a single player. The performance of soccer teams also varies a lot over the season. That is, they seem to perform well only on a temporary basis which makes it harder to rank the teams and predict the outcomes, whereas the performance chess players tends to be consistent or improve gradually over time. For instance, Figure 3 shows us that even for a game where the home team is significantly stronger than the away team, there is still a 15-20% chance that the home team won't win. Essentially, this is because more factors have to be taken into account when predicting the soccer outcomes such as weather, current roster, travel distance, etc. This is not the case for an individual event like chess in which the environment does not vary much.

7.2 Local Properties Feature Analysis

The analysis of feature importance in Table 6 of the Appendix corresponds to the fraction of splits made using each feature in the decision trees. For instance, the feature DVL corresponds to the number of paths from A to B of the sequence/form [Draw, Victory, Loss]. Features starting with "White_" correspond to the total number of victories, draw and losses of the white player.

The ranking and the numbers change when looking at different datasets but the results are quite similar overall. Surprisingly, the distribution of the importance seems to be quite even among the features of the same type. Furthermore, the same kind of features appear to always hold the most relevant information: the global rankings, followed by the total win/loss/draw counts, and finally, the different kinds of paths sorted by decreasing length. The fact that the longer paths are more important than shorter ones might be because the longer paths hold more relevant information that is less reflected in the global features. Note that for basketball, there is no draw. Hence, all the scores of the features containing draw are zero.

7.3 Soccer Betting Odds Comparison

Ranking Method	Win	Draw	Loss
Baseline	14.6% (11.8%)	4.1% (3.8%)	12.3% (9.8%)
PageRank	11.4% (7.7%)	2.2% (1.8%)	10.1% (7.3%)
BeatPower	11.3% (8.5%)	2.4% (2.1%)	9.9% (7.8%)

Table 5: Average error between our predicted probability distribution with the historical fair betting odds. The numbers in the parenthesis are standard deviations

We compare our probability distribution with the historical fair betting odds provided with our soccer dataset. Table 5 shows that our methods give us more precise estimates of the probability for each outcome with smaller

variance. For instance, the top left number is 14.6% (11.8%) which means that if the fair probability is 20%, the baseline method will predict around $20\% \pm 14.6\%$ with standard deviation of the error around 11.8%. We see that our methods perform significantly better than the baseline methods in terms of both the average errors and their variances, especially for the probability of draw.

8 Future Work

Given our current results, there are still efforts we can take on in the future to improve performance. We will develop new algorithms and try to tweak what we currently have to attain better predictions. Learning from the features which were important in the random forest run, we will try to focus on creating similar features and incorporate them into our model.

As we initially started off with the chess dataset, which only awards 1 point to the winner or 0.5 to both players in the case of a draw, we chose to maintain a similar point system for the basketball and soccer datasets when constructing the PageRank and other edge-based computations. Therefore, our existing models do not have information about whether a match was won by a small or large point difference. We believe that a point-weighted graph would yield more predictive power than the current unweighted scheme, so this would be a natural next step for this project, and adjustments will need to made to accommodate different weighted paths for the machine learning models.

Generally, it is trivial to predict that the better player will beat the other player. Given that we have models that output the probability of each outcome but we do not have a concrete way to evaluate them, we want to use real-world betting odds to measure how well our probability prediction is and see how much money we can make, as one potential application of our models. Our research shows how difficult it is to predict game outcomes and that there is not one technique that suits all tournament graphs. Overall, the results of our models demonstrate promise in applying relationship/network-based analysis to prediction systems.

References

[1] A. Govan, C. Meyer, R. Albright. Generalizing Googles PageRank to Rank National Football League Teams. *SAS Global Forum 2008, Data Mining and Predictive Modeling*, 2008

[2] Beatpaths: The Winning Ways of Winners. *http://beatpaths.com/2005/11/09/what-is-beatpower*

[3] How We Calculate NBA Elo Ratings. *http://fivethirtyeight.com/features/how-we-calculate-nba-elo-ratings*

[4] P. V. Rao and L. L. Kupper. Ties in Paired-Comparison Experiments: A Generalization of the Bradley-Terry Model *Journal of the American Statistical Association* Vol. 62, No. 317, pp. 194-204, 1967.

[5] R. Herbrich, T. Minnka, T. Graepel TrueSkill(TM): A Bayesian Skill Rating System. *Advances in Neural Information Processing Systems*, 2007.

Appendix

	Chess		Soccer		Basketball	
Rank	Feature	Importance	Feature	Importance	Feature	Importance
1	PRank_ratio	10.35%	PRank_ratio	4.48%	PRank_ratio	11.00%
2	White_PRank	7.58%	White_PRank	4.04%	White_PRank	8.51%
3	Black_PRank	7.52%	Black_PRank	3.31%	Black_PRank	8.25%
4	White_L	5.11%	DVL	3.23%	White_V	6.81%
5	Black_L	5.09%	DDD	3.11%	VLV	6.66%
6	Black_V	4.72%	DDL	2.84%	Black_L	6.62%
7	White_V	4.67%	VDD	2.79%	White_L	6.09%
8	Black_D	4.60%	LVD	2.71%	LVL	5.85%
9	White_D	4.55%	DLD	2.68%	Black_V	5.19%
10	LLV	1.83%	DVD	2.63%	LVV	4.23%
11	LVV	1.82%	DDV	2.61%	LLV	4.09%
12	LDV	1.75%	White_D	2.49%	VVL	3.94%
13	LVL	1.72%	LDD	2.43%	VLL	3.71%
14	VLV	1.70%	VLV	2.41%	LLL	3.51%
15	LLL	1.63%	VLD	2.40%	LV	3.15%
16	VVV	1.61%	White_V	2.39%	VVV	2.96%
17	DLV	1.55%	DLV	2.39%	LL	2.80%
18	LLD	1.54%	LVL	2.38%	VL	2.70%
19	LVD	1.54%	Black_V	2.37%	VV	1.91%
20	DDV	1.50%	LDV	2.36%	L	1.26%
21	LDD	1.49%	LDL	2.33%	V	0.75%
22	DVV	1.48%	VDV	2.31%	D	0.00%
23	VLL	1.47%	White_L	2.17%	VVD	0.00%
24	VVL	1.46%	Black_L	2.16%	VD	0.00%
25	DDD	1.45%	DVV	2.16%	DVV	0.00%
26	LDL	1.42%	LLD	2.06%	LLD	0.00%
27	VLD	1.42%	VVD	2.02%	DDL	0.00%
28	DVL	1.42%	VLL	1.99%	DLL	0.00%
29	DLD	1.36%	DD	1.93%	DDD	0.00%
30	VDV	1.35%	Black_D	1.92%	DVD	0.00%
31	DVD	1.34%	LLV	1.90%	White_D	0.00%
32	DDL	1.27%	LVV	1.88%	DLV	0.00%
33	DLL	1.25%	DLL	1.88%	VDV	0.00%
34	VVD	1.24%	VDL	1.83%	DL	0.00%
35	VDL	1.24%	VVL	1.81%	Black_D	0.00%
36	VDD	1.23%	DL	1.50%	DD	0.00%
37	LV	0.68%	DV	1.48%	VDD	0.00%
38	VL	0.60%	VD	1.31%	DV	0.00%
39	LL	0.59%	VL	1.30%	VDL	0.00%
40	DD	0.56%	LD	1.27%	LD	0.00%
41	VV	0.56%	LV	1.17%	LVD	0.00%
42	LD	0.55%	LLL	1.10%	LDD	0.00%
43	DV	0.52%	VVV	0.97%	DLD	0.00%
44	DL	0.50%	VV	0.82%	VLD	0.00%
45	VD	0.48%	LL	0.75%	LDL	0.00%
46	D	0.24%	L	0.75%	LDV	0.00%
47	L	0.24%	D	0.64%	DDV	0.00%
48	V	0.22%	V	0.52%	DVL	0.00%

Table 6: Features ranking for different datasets