Luck vs Skill in College Basketball

Jake Browning, Rohan Mawalkar, Seth Peacock

November 2024

1 Letter to the Editor

In addition to this report, write a one-page letter to the newspaper chief editor, explaining the main results of the report and suggesting findings that can be communicated with the basketball fans reading the newspaper.

2 Overview

Describe the problem, your model, results and how your model performed.

3 Introduction

People love sports because they are a showcase of hard work and talent. People also love sports because they are unpredictable. How do we quantify the effect of one versus the other in a basketball game? On the one hand, we often see this framed as a strict dichotomy between luck and skill: sports outcomes depend in some part on one and in some part on the other. However, careful examination reveals that it is rather unclear where to draw the line between luck and skill. For example, if a team happens to study exactly the right film the night before a big game and is able to perfectly foil the other team's plans, is that luck or is it skill? On the one hand, it certaintly takes skill to study your opponent and use this information to your advantage. On the other hand, they could have easily picked the wrong film to study! Instead, we would like to approach this problem using a dichotomy between certainty and uncertainty. This also addresses the issue of a lack of data; there are many aspects of "skill" (such as individual player records) which require more data than we have been able to gather so far.

Our goal here is twofold: first develop a model which predicts the outcome of a basketball game, and second quantify the effect of luck/uncertainty in a basketball game. Since we want to incorporate the effect of uncertainty, our prediction model will contain a stochastic element which we will have to derive. To train our model, we will use data from the regular season games leading up to the 2024 March Madness Tournament, and then test our model's ability to predict the outcome of the tournament. However, we will also discuss additional markers of uncertainty in our data to better understand the effect of uncertainty on a basketball game in a more general sense.

4 Methods

Our model will be a modified version of the classic Elo ranking in chess, which we will use along with data from throughout the regular season to assign each team a ranking. (From here on, when we say "Elo ranking" we will mean our modified ranking, and will say "classic Elo ranking" if we mean the original.) Given a pair of teams, these rankings will be used to give us probabilities of an upset (that is, the probability that the team with the lower Elo rating wins).

4.1 Our "Elo" Ranking

Traditionally (see [5]), the Elo ranking system first calculates an expected score for the game as a function of the difference in the two teams' Elo ratings. Here, E is the "main" team while E_{opp} is the opposing team:

$$E = \frac{1}{1 + 10^{\gamma_{opp} - \gamma}/c} \tag{1}$$

where γ_n is the main team's pre-game Elo score (after game n), γ_{opp} is the opposing team's pre-game Elo score, and $c, K > 0$ are constants. The main team's Elo rating is updated after their $n + 1^{\text{th}}$ game based on the following formula:

$$\gamma_{n+1} = \gamma_n + K * (O - E) \tag{2}$$

Where O is 1 if the team wins and 0 if the team loses. This is essentially performing a stochastic gradient descent (taking a step after each game) for a logistic regression model, where the Elo ratings are the weights of the model [3].

However, this score update does not take into account the score differential of the outcome of the game. Thus we propose a modified Elo system which uses the score outcomes to update a team's Elo score in one of the two following ways:

$$\gamma_{n+1} = \gamma_n + K^{\frac{S_{\text{win}}}{S_{\text{lose}}}}(O - E) \tag{3}$$

$$\gamma_{n+1} = \gamma_n + K * (O - E) * \frac{S_{\text{win}}}{S_{\text{lose}}} \tag{4}$$

where S_{win}, S_{lose} are the scores of the winner/loser and the other variables are defined as above. We will calculate E in the same way setting c to the canonical 400 for simplicity, and choose K based on considerations described in the next section near a canonical value of $K \approx 32$ [5].

We considered several possibilities for incorporating the score differential. We considered defining the score differential as the absolute value of the difference of the scores. However, this would weight a 20–10 loss as the same as a 90–80 loss, which seems much less accurate to how basketball fans see the game. Thus we would like to instead use the ratio $\frac{S_{\text{win}}}{S_{\text{lose}}}$. The question now becomes: how much do we want to allow this ratio to effect the score update? In Equation 3, we allow it to have a much more significant effect than it does in Equation 4. Ultimately, we decided to use both; we will discuss this further in the following section.

4.2 Elo Calculation Algorithm

Our goal is to have an Elo ranking for every team that made it to the 2024 March Madness Tournament, using the results of the games in the regular season. However, many of these regular season games were not played against the teams that made it to the tournament. Thus, we also needed to assign Elo ratings to teams that did not play in the regular season. Note there aren't many games for these non-tournament teams; we will discuss this issue below.

We start all teams at the same Elo rating. This is typically a high positive value (to prevent giving a discouraging negative rating to low ranking chess players), but since the model just uses the difference in Elo ratings it doesn't matter what the initial value is. We chose to set it at 0 for simplicity. We then sort the games by date, and use one of the Elo update formulas to update the Elo ratings for every game. There are four cases to consider:

1. The update for a tournament team after beating a non-tournament team. In this case, we use Equation 4, as we don't want this game (or the score differential) to have a huge effect — the tournament team should have won anyhow.

2. The update for a non-tournament team playing any other team. In this case, we use Equation 4; there aren't very many games for these teams, so we don't want their Elo rating to change too much from zero just based on a few games. Of course, we still want a tournament team's score to go down significantly if they lose to this team; see next point.

3. The update for a tournament team after losing to a non-tournament team. In this case, we use Equation 3, as we want the score differential to have a large effect. This tournament team is likely to do much worse in the tournament if it lost in a blowout to a team that didn't even make it to the tournament. As mentioned above, these non-tournament teams will have average Elo ratings, so we want to emphasize the fact that this really is a team the tournament team should have beat.

4. The update for a tournament team after playing another a tournament team. In this case, we use Equation 3 to give the score differential a large effect; blowouts in this

After all of the games have created updates, we now have the Elo rating for every team that made it to the tournament. At this point, we will throw out the other teams that didn't make it to the tournament, as their Elo ratings are less meaningful due to the small numbers of games.

4.3 Bootstrapping

We would like to verify the accuracy of our Elo system. To do this, we "bootstrap" [2] samples from the regular season games to compare our model's predictions of the probability of an upset to the actual observed probability of an upset for a given difference in Elo rankings. Let $E_u(\Delta_{\text{Elo}})$ be the former probability and $\hat{p}_u(\Delta_{\text{Elo}})$ the latter, where Δ_{Elo} is the difference in (final) Elo ratings between the two teams. If our Elo ratings are accurate, we should see these are close to one another.

First, we find the minimum and maximum differences in (final) Elo rankings for all of the games played. We then divvy this up into equal parts, as small as we can but with each partition large enough to contain a significant portion of the games. See JAKE INSERT FIG for a histogram of how many games fall into the partitions we chose. For each partition, we then calculate $E_u(\Delta_{\text{Elo}})$ via Equation 1 using the middle Δ_{Elo} for that partition. Let N be the number of games played in the regular season. We then sample with replacement N times from the subset of games that falls into that partition. To calculate $\hat{p}_u(\Delta_{\text{Elo}})$, we simply take the number of upsets (lower Elo rating team wins) divided by

N. This is the maximum likelihood estimate (MLE) for the probability of an upset in the given partition [4].

To quantify the accuracy of our Elo system, we plot $E_U(\Delta_{\text{Elo}})$ against $\hat{p}_U(\Delta_{\text{Elo}})$ for each value of Δ_{Elo}, and calculate the MSE of these points vs the $y = x$ line. We then repeat this process for a range of K values ($K \in 10, 15, 20, 25, 30, 35, 40$), and choose the K that minimizes the MSE. (See INSERT REF TO PLOT.) Of course, this process increases the risk of overfitting to the regular season data. However, if we incorporate the tournament games, then we are not providing an unbiased assessment of our model's performance on the tournament data. See section 6 for further discussion of overfitting.

4.4 Weaknesses

An obvious weakness of our model is our lack of data for the other teams in the league. Ideally, we would like to have accurate Elo ratings for these teams so that we can get accurate Elo updates for the tournament teams. We have tried to address this by changing the update formula for non-tournament teams, but ultimately we would prefer to just have more data for those teams. Another weakness is the previously mentioned risk of overfitting to the regular season data due to the selection of K.

5 Results

5.1 Verifying Elo Rankings

As discussed in subsection 4.3, we plot $E_u(\Delta_{\text{Elo}})$ against $\hat{p}_u(\Delta_{\text{Elo}})$ along with the $y = x$ line. If our Elo system has perfectly captured the expected upset probability, then these points should be close to the $y = x$ line. To measure the inaccuracy, we calculate the mean squared error. Plot Y shows the effect of K on the MSE (as discussed in subsection 4.3) while plot X uses the value of K that minimized the MSE. JAKE GENERATE PLOTS???

5.2 Quantifying Uncertainty

To first order, we now also have two (hopefully similar) ways of quantifying the measure of uncertainty in a basketball game: the expected probability of an upset based on Elo rankings (as a function of the difference in Elo rankings), and the bootstrapped estimate for the probability of an upset (as a function of the difference in Elo rankings). Using the expected probability as a measure of the uncertainty assumes that the formula used to calculate the expected probabilities given Elo rating is a good model for the probability of an upset. Using the bootstrapped probability as a measure of uncertainty assumes that we have enough data to be able to approximate the true probability of an upset for a given class of difference in Elo rankings. Both of these measures assume that our Elo rankings — or at least their differences — accurately capture the differences in skill level between teams (which is no small assumption). We will make this aassumption for the remainder of this paper; while our Elo system might not be completely accurate, it could be improved with more time and the incorporation of more data (see section 6).

We would like to somehow combine these two measures of uncertainty for our predictions. One option here is to simply average the two upset probabilities. Another option is ROHAN???

We would also like to consider a "higher order" kind of uncertainty by comparing the bootstrapped probability of an upset to the expected probability of an upset. For example in cases where these probabilities differ significantly, we might consider increasing the chance of an upset??? ROHAN?

5.3 Predicting March Madness 2024

We would like to see how well our model could have predicted the results of the March Madness 2024 Tournament. To this end, we ran a large number of simulations using our Elo predictions with our predicted chance of upset and recorded what percentage of times we correctly predicted the outcome of the tournament. Since this was a very low value, we also recorded the average percentage of games for which we correctly predicted the outcome. JAKE? Plots?

Of course, due to the uncertainty everyone acknowleges exists, this percentage would be low no matter what. After all, no one has ever correctly guessed the result of the tournament [1]. However, we note that our model (on average) performed better than just picking the winner as the one with the better seed. Using this strategy would have yielded correct predictions on (JAKE?)% of the games.

6 Next Steps

There are several things we would have liked to check if we had had more time. For one, it would be interesting to explore the effect of overfitting by the choice of K. That is, does the value of K which minimized the MSE for the bootstrapping verification also maximize the likelihood of predicting the actual outcome of the tournament? If not, then we may want to look into a different way of picking K (perhaps one which incorporates past data) as this choice seems to have lead to overfitting on the regular season games. We also would have liked to explore incorporating the score differential in different ways (such as $\gamma_{n+1} = \gamma_n + K^{\frac{S_{\text{win}}}{S_{\text{lose}}}}(O - E)$ as previously mentioned).

Another thing we would have liked to have done is to see how our MSE estimates would change if we added noise to the data; for example, we could have multiplied each score in the game by some rangomd number between 0.8 and 1.2 and seen if we would still have obtained a similar ideal value of K and the similar MSE for this value. If so, this would indicate our Elo system is robust, and that our K value is not overfit to the regular season data.

6.1 Additional Data

There are several things we would like to do if we had the time and resources to acquire more data.

- We would have liked to incorporate the homefield advantage in the expected score calculated from the Elo system. If we had had this data, we would have added some number A from the expected score of the home team and subtracted it from the expected score of the away team. For example, if we found that home teams on average win 53% of games, then we would add/subtract $A = 0.03$. This way, the expected score of two teams with the same Elo score would be 0.5 ± 0.03.
- To pick the K that minimizes the MSE for our Elo system compared to the bootstrapped estimate, we could use data from past seasons. We could also use the data from past tournaments, since those data may be more accurate for predicting tournament games (e.g. higher levels of variability due to high pressure scenario).
- As previously mentioned, it would have been ideal to have an accurate Elo rating for the other teams in the league, which would require regular season data for those teams.

References

[1] Christopher Brito. Has anyone ever had a perfect bracket for March Madness? The odds and precedents for NCAA predictions — cbsnews.com. `https://www.cbsnews.com/news/perfect-bracket-march-madness/`. [Accessed 16-11-2024].

[2] Trist'n Joseph. What Is Bootstrapping Statistics? — Built In — builtin.com. `https://builtin.com/data-science/bootstrapping-statistics#:~:text=%E2%80%9CThe%20advantages%20of%20bootstrapping%20are,other%20groups%20of%20sampled%20data.%E2%80%9D`. [Accessed 16-11-2024].

[3] Steven Morse. Elo as a statistical learning model — Steven Morse — stmorse.github.io. `https://stmorse.github.io/journal/Elo.html`. [Accessed 16-11-2024].

[4] Joram Soch. Maximum likelihood estimation for binomial observations — statproofbook.github.io. `https://statproofbook.github.io/P/bin-mle.html`. [Accessed 17-11-2024].

[5] Stanislav Stankovic. Elo Rating System — stanislav-stankovic.medium.com. `https://stanislav-stankovic.medium.com/elo-rating-system-6196cc59941e`. [Accessed 16-11-2024].

7 AI Use Report

ChatGPT was used to fix bugs in data processing code as well as for preliminary exploration on the problem. The only ideas we used from this preliminaryexploration were those which we then found true sources for and cited in the paper.