Sports Analytics

Alex Hentzen   Ziyi Wang   Jinming Zhang   Yiran Zhong   Andrew Zimmer

April 2022

1 Introduction

Someone hoping to make money by betting on the outcome of a college football game would benefit from being able to know the outcome of the game before it begins. A person betting randomly on the outcome of a game has a 50 % chance of guessing the outcome correctly, so any model with a better accuracy rate than that would be an improvement. However, the amount won on any game will depend on how the result of the game differs from the bookmaker's predicted outcome; therefore, the accuracy of a prediction model must be high enough to ensure that the money won will be greater than the money lost, which is to say that the model must be more accurate than the model used by the bookmakers.

2 The data

The biggest obstacle in generating a sufficiently accurate model is in the data available to build and train the model. The college football season consists of approximately 16 games, and college teams have significant turnover between seasons, which limits the amount of data strictly relevant to any given game to data from the same season. Of course, the data from previous seasons isn't something we need to disregard completely, as only part of each team will turn over between seasons, and the turnover will not be the same for every team. Furthermore, schools that are well known for their football team will have an easier time recruiting talented players and coaches. Therefore, to a certain extent, a team's quality is self-reinforcing. However, since 'quality' isn't something we can directly measure, we're left to check the data we can measure. Courtesy of 'collegefootballdata.com' we have the points scored by each team, what venue the game was played at, and the number of fans in attendance, among other things. Obviously, a myriad of other information is potentially available about each game that could have affected the outcome of the game; the weather on game day, injuries among each team's starting lineup, even recent natural disasters or local tragedies. In a particularly meta sense, another source of data is the bookmakers themselves, as consulting the predicted line scores for a team can give a sense for how they are expected to perform.

3 The Models

In this section we describe the several models we implemented. We will start with the basic model that only calculate win probability by performance of previous games in the same season. Next, we are trying to use Elo score updating to record and predict the probability. To compensate the shortcoming of few data, we then use logistic regression to learn the data from 2010 to 2019. And we also build the model for betting as most of people may be interested.

3.1 Naive win-loss record model

The Naive win-loss model is the basic model we used to predict the outcome of game. For this model we didn't predict the outcome based on the Elo score, instead we count the number of wins and losses for each team and calculate each team's odds of winning by reading the team's history record as a variable to calculate the winning probabilities for both team. To be specific, first we created the data frame based on the content we read from the csv files which contain the information of the teams in each game, then we expand the data frame with new features that we need to fill in with the probability we calculated. Then we check whether the team we read from the data frame is in the top rank list, if not we ignore them and read the next couple of teams, because we only wants to predict on the competitive team. For each team, we will count its number of winning games and number of all the games it played. Then if the team was searched by first time, we set the initial winning probability by their ranking, if not we read its past record and put in this formula

$$\text{(Prob. team 1 wins)} = \frac{1}{2}\left(\frac{w_1}{n_1} + \frac{n_2 - w_2}{n_2}\right)$$

where w is number of the wins and n is total games team played, to get the probability for team to win. Finally we fill these probabilities in our data frame.

3.2 The Elo model

The Elo model is another possible way to attempt to predict the outcome of games. An Elo model is relatively simple, both to build and understand. The first step is to create a data frame with all of the teams playing during the season. Next, initialize the data frame, either with uniform values across the teams or, alternatively using modified Elo scores from a previous season. Our model initialized with every team at 1000, for simplicity's sake. Then, we consider each game individually. The home team has a small increment added to their Elo score; in our model, 50 points. Then, each game is considered as a difference of Elo scores. The team with the higher score is judged to be the likely winner, with any game between teams with equal scores considered a likely tie game.

$$\Delta E = Elo1 - Elo2 \tag{1}$$

$$\text{(Prob. team 1 wins)} = p_1 = \frac{1}{10^{-\Delta E/400} + 1}$$

$$\text{(Prob. team 2 wins)} = p_2 = 1 - \text{(Prob. team 1 wins)}.$$

The difference between the Elo scores is used to generate the probability that each team will win. Then, the results of the game are compared to the predicted results. The winning team gets points added to their Elo score, while the losing team gets those points taken from their Elo score. The magnitude of the points gained or lost is determined both by the predicted probability of the results (i.e. a predicted blowout will change scores more than a predicted close game) and by the magnitude of the k value that is chosen for the model. The k value should be large enough to adjust the scores if the prediction was wrong, but not so large as to let a single unlucky game crater a team's score.

$$\text{(new Elo for team 1)} = \text{(old Elo for team 1)} + k(1 - p_1) \tag{2}$$

$$\text{(new Elo for team 2)} = \text{(old Elo for team 2)} + k(0 - p_2) \tag{3}$$

3.3 The Logistic model

Based on the Elo model we introduce in the previous section, we combine the data from different years and each year's Elo model to train in a logistic regression model for predicting the outcome of one game. And for each year's initial Elo score we use one third of the past season's final Elo scores, which we minimize while recording the performance of each team. Logistic regression, as a machine learning technique, is good at classifying the outcome as 1 or 0, which is the state of win or loss. By combining ten year's data, we deal with the problem of limited data for one session. The features we picked are Elo scores of two teams and Team recruiting rankings and ratings, which is a good indicator of teams overall performance of all the players.

3.4 The betting spread model

Other than the outcome of each game, we are also interested in predicting if we could win the betting line based on the features Elo scores, team rankings, and betting line. To achieve this goal, we again use logistic regression with three classes, 1 for bet above the betting line, 0 for the same as betting line, and -1 for lower than betting line. We calculate and combine the betting data to our data frame. To utilize this prediction, we will choose the probability with higher than 0.6 then to bet.

4 Evaluation

4.1 Naive win-loss record model

The goal of our model is calculate the winning probabilities to predict the outcome of the team without the Elo score. Our expectation is that our model will has lower cost of prediction than the Elo model which leads to the higher efficiency when we put the features get from win-loss model in training with logistic regression. However, the result indicate our basic model has higher cost which shown in the Figure 1, which is not what we expected but when we compare the accuracy of the both models we trained using logistic regression (as shown in Figure 2), the win-loss showed the higher accuracy. The win-loss has accuracy 0.7946 and the Elo model's is 0.7508. The Win-loss got slightly lower stand deviation (0.009654) than the Elo model (0.015443).

4.2 The Elo model

The goal of our model is to get the Elo score of each team based on their performance in previous seasons and then make further predictions based on these Elo scores. To calculate the Elo score, we first need to get the value of constant k (an important factor in our model). In the following we show how to find the ideal k, which makes the average cost of prediction (the proxy for efficiency of our model) low enough. The cost of predication is calculated by using the log loss function:

$$\text{(cost of prediction)} = -\Big(y_i \log(p_i) + (1 - y_i)\log(1 - p_i)\Big)$$

where y_i denotes the actual outcome (i.e., $y_1 = 1$ if team 1 wins and $y_1 = 0$ if team 1 loses). Then, we compute the average of the cost:

$$\text{(efficiency of model)} = -\frac{1}{\text{(number of games)}} \sum \Big(y_i \log(p_i) + (1 - y_i)\log(1 - p_i)\Big).$$

In Figure 3 we show how efficiency of model responds to different values of k. When the value k is equal to 25, the cost is about 0.5 which is not accurate enough for our model. As the value of k increases to around 100, we see no significant change in cost as the value of k changes. Therefore, we choose to use the value around 100 for our model which has a lower cost of around 0.42. Then, we use logistic regression to compute the average accuracy of our model, which is around 72%. Also, we compute the Mean Squared Error (MSE), which is about 0.28. Thus, we are confident that our model has sufficient accuracy to make the prediction.

4.3 The Logistic model

Using ten years' data, we get 3753 games as data, and we have found a great improvement on the accuracy. Increase from 60 percent to average 72 percent, logistic model shows its robustness. We use the pipeline with Standscaler method at first to normalize the data the we split the train set and test set into 8:2. As the following figure shows, we use mean square error to estimate how far our model prediction is from the correct outcome. And we compare to the log loss of using Elo model directly for each season.

4.4 The betting spread model

The accuracy from testing data achieves about average 62 percent, which is much better than 55 percent, the least probability to win money from the bet. To further increase the probability to win money, we only choose the prediction probability that is higher than 0.6 to make decision to bet. To examine the performance of predicting betting spread, we simulate to start from 11000 dollar to bet as the following figure shows. Each bet costs 110 dollar, if we win we will earn 210 dollar, and if we lose we will get nothing, and if the game is a draw, the 110 dollar will be returned. After splitting the data frame to train set and test set, we have total about 600 games for testing and to make simulation for betting. Among these 600 games, we get about 100 to 200 games that has probability higher than 0.6 so that we could bet. And we could win about 4000 dollars averagely. The result is quite good, although there is one disadvantage that for 600 games we need about two year's time.

5 Future directions

We can improve our Elo models by combining more data as features to updating the Elo scores. Data such as special sport star performance and players‘ injured situations may have big effect on the game outcome. It may also be important to collect the weather of the game day, which a terrible whether may also increase the uncertainty of the outcome. We may need to comply with the factors that effect the probability from Elo scores.

Figure 1: Comparison of cost

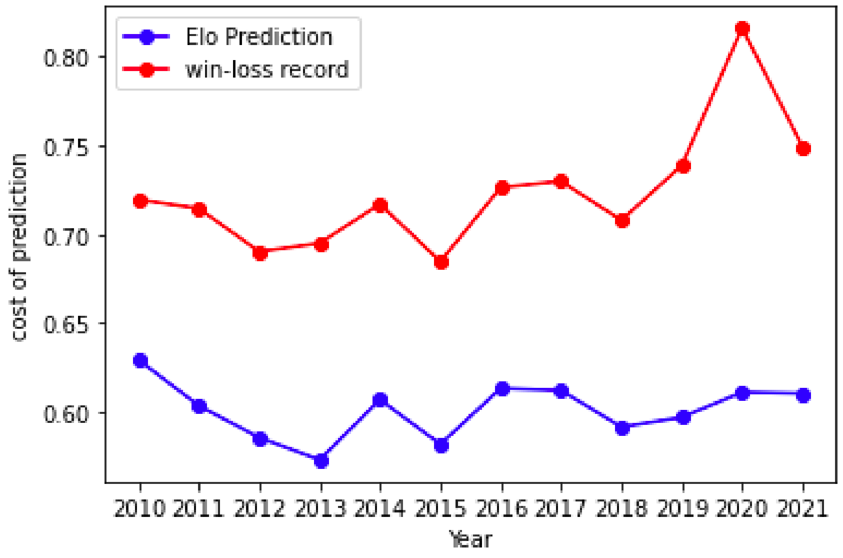


Figure 2: Comparison of Accuracy

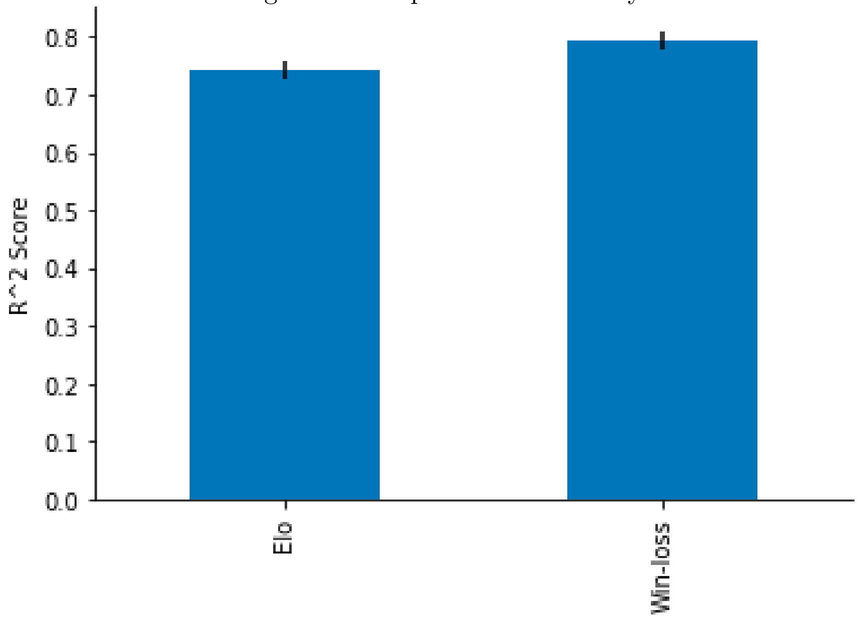


Figure 3: The Change of Log Loss With the Value of k.

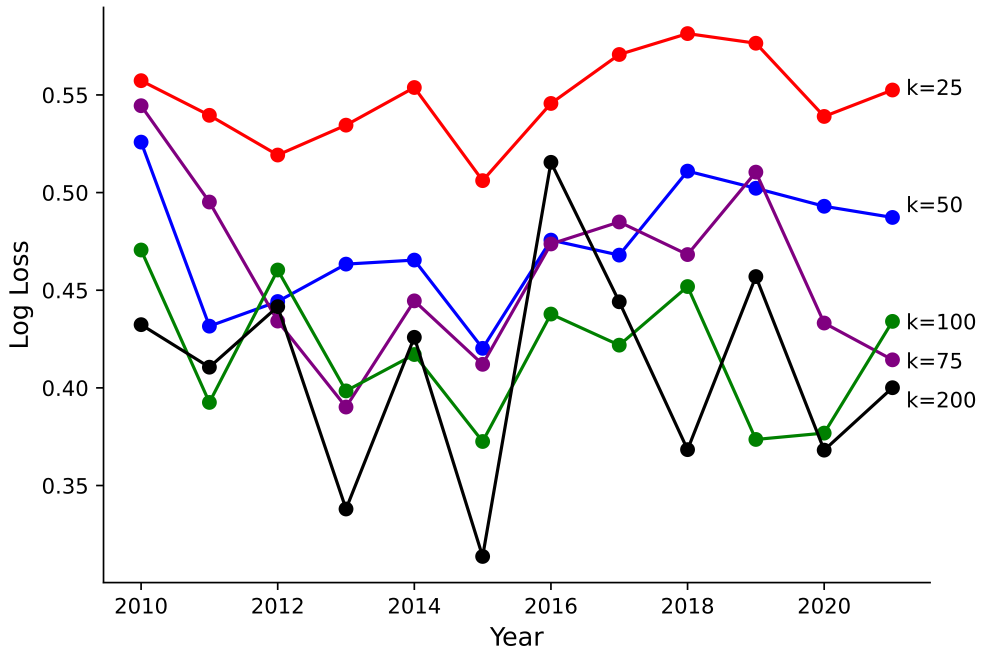


Figure 4: Comparison of Accuracy

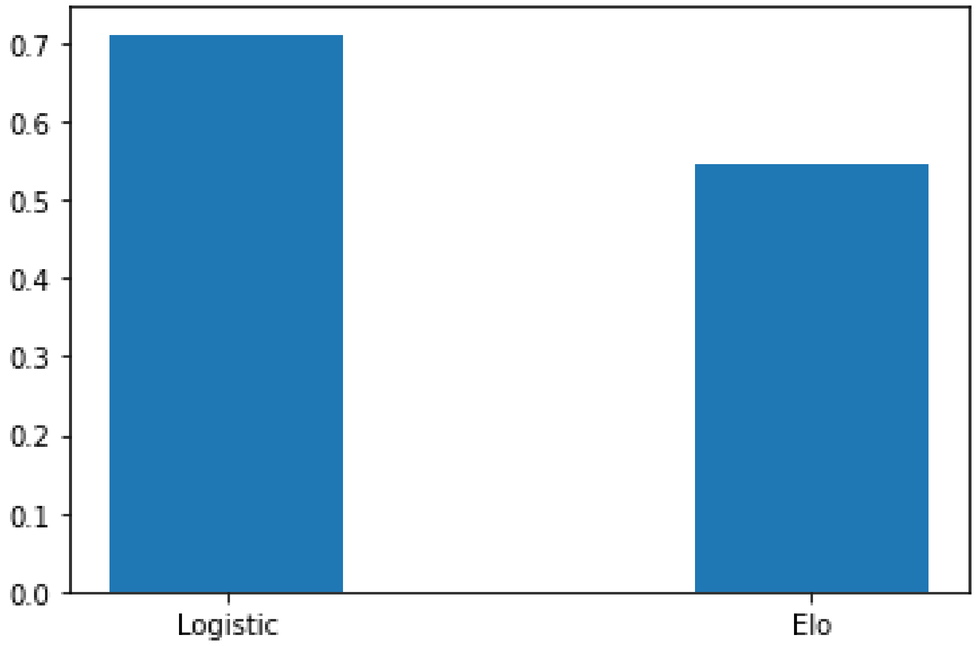
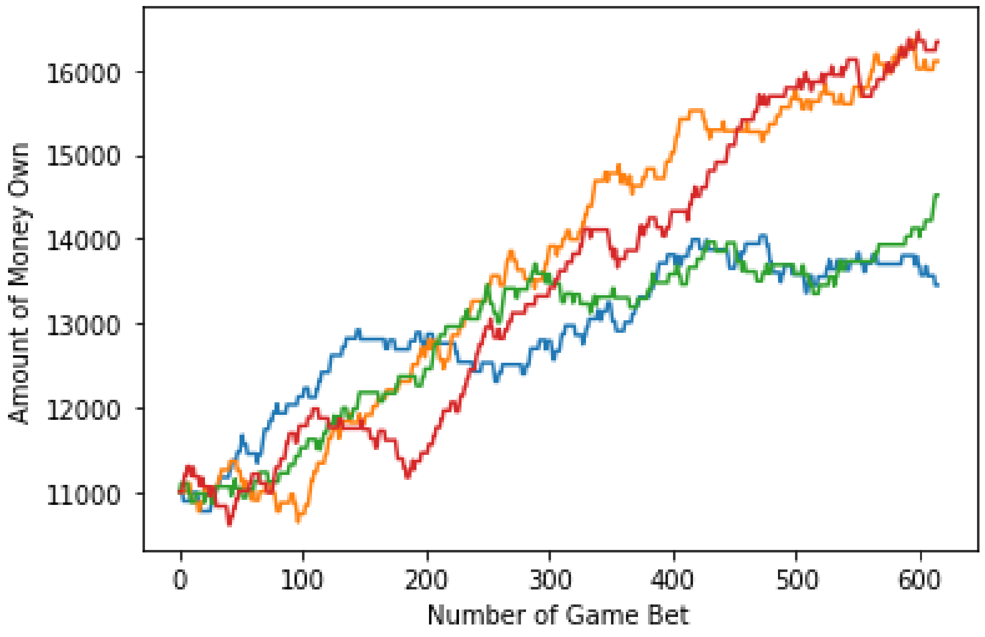


Figure 5: Betting Simulation


Amount of Money Own
Number of Game Bet