92

NBA ELO and Predicting Games

Ty Buckley
MSU NetID: buckl113

Project Category: General Machine Learning
https://github.com/buckl113/CMSE381Project

Final Report

1 Introduction

In professional basketball, predicting the outcome of games can be quite challenging, since so many factors impact who wins a game. Sometimes, teams that are perceived as "bad teams" even beat title contenders. In order to predict these outcomes and to track each team's historical success, statistical measures such as ELO make this easier. This project will attempt to find a model which predicts the outcome of basketball games at high accuracy using multiple classification methods. Some of these methods which will be used are . The features which will predict which team wins a game will be , , and .

2 Related Work

A previous project aimed to track each professional basketball team's success on a game-by-game basis. In FiveThirtyEight's data visualization project [1], the concept of ELO is examined on a team-by-team basis to determine each National Basketball Association franchise's historical strength. The ELO ratings are determined by using the final score of a game and the home team and updating each team's ELO based on these results. When a team wins, it gains ELO points, which can be a lot of points in an upset victory, and can also be few points if they narrowly win a game that they were favored in. The losing team loses the same amount of ELO points that the winning team gains.

There are a few other rules that are important to note about how the ELO is calculated in the project. Each team starts with an ELO rating of 1300 in their first professional season. For existing teams, the NBA ELO ratings carry over 75% of last season's points, while 25% of their points come from the average ELO for

teams not in their first season (1505) [2]. These rules can be explained with Algorithm 1:

Algorithm 1 Team ELO for New Season

if team already exists **do**
$ELO_{New} = (ELO_{Old} * .75) + (1505 * .25)$
end if
else do
$ELO_{New} = 1300$
end else
return ELO_{New}

In this project, a few specific teams and their ELO were examined in depth. First was an example of the historical ELO of the team tied with the most NBA championships, the Los Angeles Lakers. The Lakers have spent a majority of their history above the average ELO (1500) and have had extremely high ELO ratings numerous times, as shown in Figure 1.

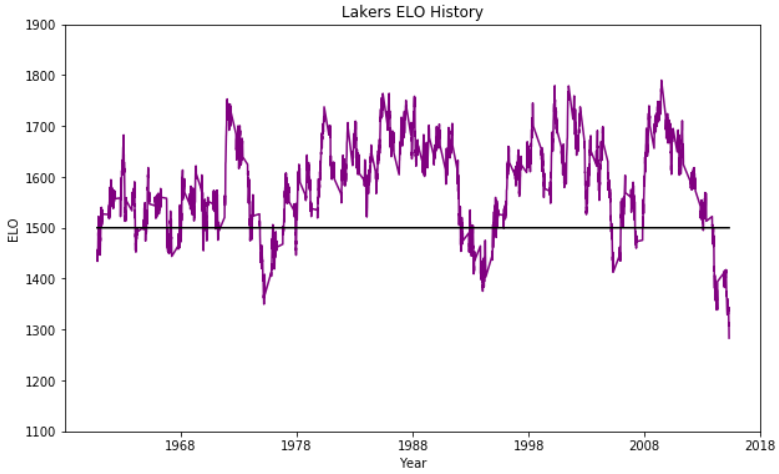


Figure 1: Los Angeles Lakers Historical ELO

Next, the 1990s Chicago Bulls were examined. This team dominated the 1990s, winning 6 championships and achieving the highest team ELO rating ever recorded, with a rating of 1853 in 1996. These results can be seen in Figure 2.

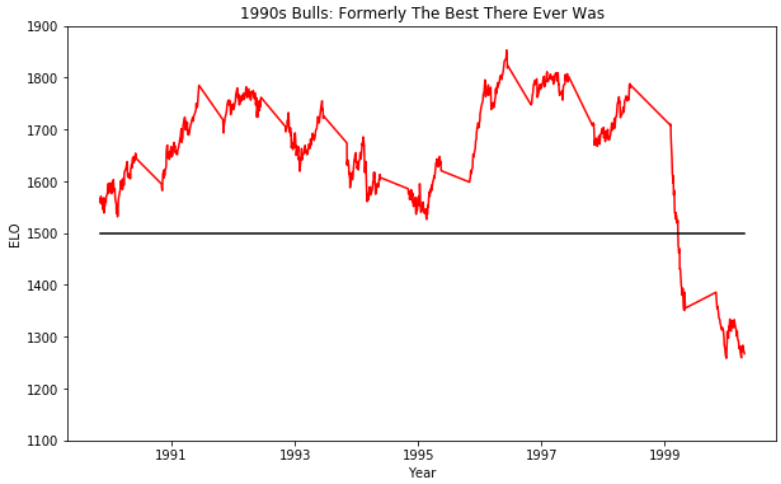


Figure 2: Chicago Bulls 1990s ELO

The Los Angeles Lakers' ELO was then revisited in closer detail. In the early stages of the 1971-1972 season, the Lakers won 33 straight games, which is the longest win streak in NBA history. Figure 3 shows their rapid increase in ELO over this time period.

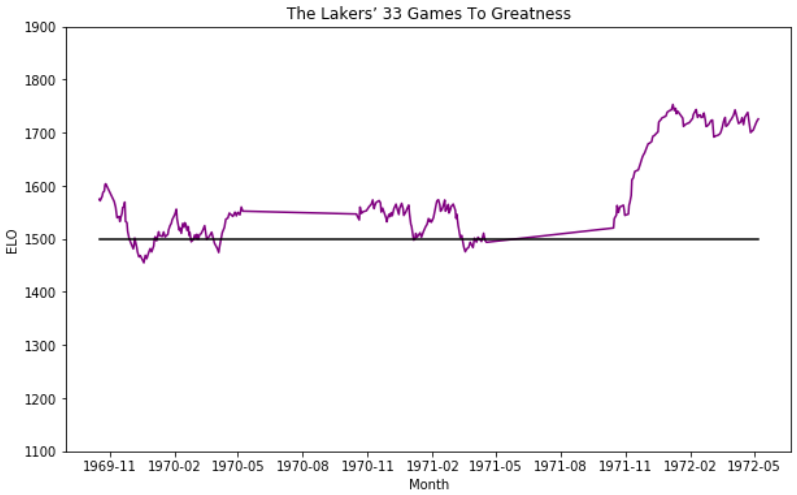


Figure 3: Los Angeles Lakers early 1970s ELO

The following team that the project wanted to find out about was the 1960s Boston Celtics. This team won 11 of 13 championships, so it would be expected that their ELO would be extremely high. However, the Celtics never reached a peak similar to the Lakers or Bulls in this time period, since they played a lot of teams with low ELO ratings, as shown in Figure 4.

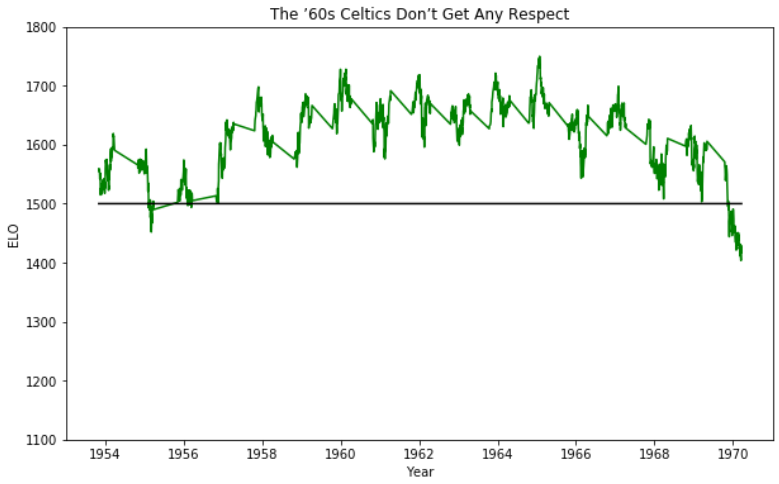


Figure 4: Boston Celtics 1960s ELO

Lastly, the project examined a team who has historically been bad: the New York Knicks. They hit their ELO peak in 1969, which makes them the team with the longest amount of time since their ELO peak.

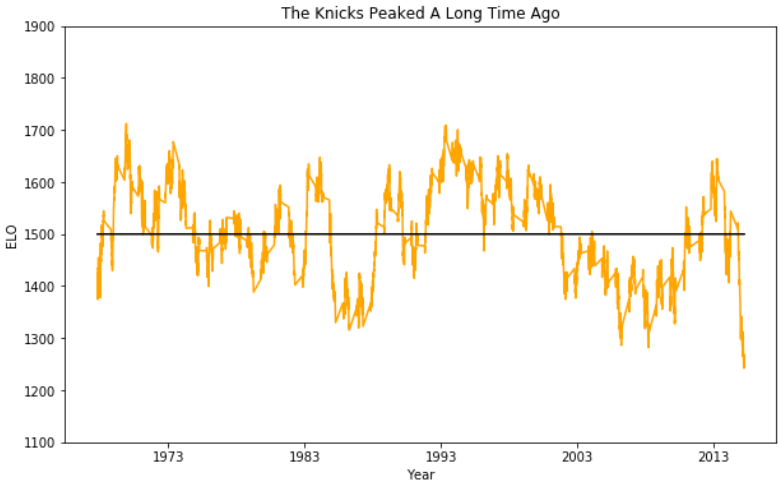


Figure 5: New York Knicks Historical ELO

As shown by the project, each team's ELO tends to change a lot over time, and they can change very quickly. Because of this, basing in-game predictions solely off the teams' historical success is not good enough. We need to take a more in-depth look at their current ELO, and other factors such as the point of the season each game occurs at, what each team's ELO trend is, and which team has a home-court advantage.

This project will use the findings from FiveThirtyEight's ELO project as a starting point to predict games. ELO is a very valuable tool in determining which team should win a game, and it could provide insights to why the team favored to win loses certain games.

3 Dataset

The dataset we are using is the same dataset as is used in the FiveThirtyEight NBA ELO project. It can be found at the FiveThirtyEight data repository. [3] Figure 6 shows the total distribution of ELO for every game, and Figure 7 shows the distribution of team ELO of every team over time.

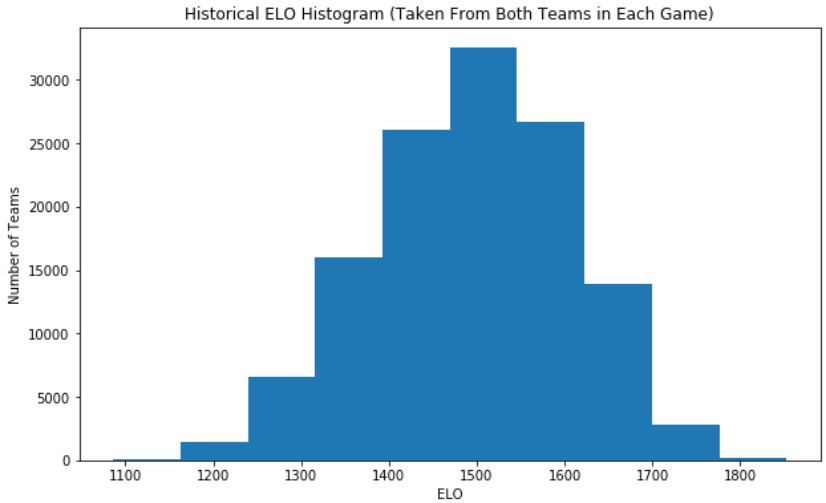
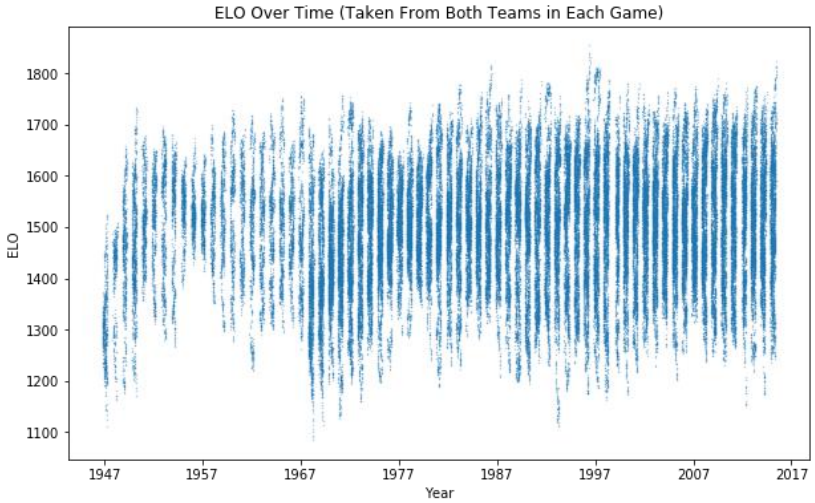


Figure 6: ELO Histogram



Better to color them using different colors.

Figure 7: ELO Over Time

In order to prepare the data to predict games, there were a few necessary data preprocessing steps. First, the year_id column was changed from a string to a datetime format in order to better visualize trends over time. Next, the dataset was cut in half, since each game is represented twice: once from the home team's perspective, and once from the away team's perspective. Since the home team wins 62% of the time in the dataset, half of the games removed were from the dataset were home games, and half were away games in order to eliminate bias. Finally, the categories corresponding to the game result (Win or Loss) and

categories corresponding to the game's location (Home or Away) were changed to one-hot vectors representing if the team won and if the team is at home.

3.1 Feature Selection

Once the data was ready to use, there were a total of 25 columns of data. Many of these columns were qualitative values such as team names, and most columns were unnecessary for prediction. The column named forecast was kept as a baseline model to compare our predictions to. This column represented the probability that the team would win the game.

The first features used are the two columns elo_i and opp_elo_i, which give the elo of both the team and their opponent entering the game. Our last feature is home_game, which has a value of 1 for a home game and a value of 0 for an away game.

4 Methods

As a baseline, FiveThirtyEight's dataset already had a forecast feature. This feature gives the probability that the team will win the game. If the probability is greater than 0.5, then the baseline model will predict that the team will win; otherwise, the model will predict the team will lose the game. Since the source of this feature is not explained, other than that is based on elo ratings and game location, we do not know which model was used to generate this forecast. Therefore, we should use this as a baseline to compare each model.

The following models were used in this project: an Ordinary Least Squares linear model; a Polynomial Ordinary Least Squares linear model; a K-Nearest-Neighbors model, and a Random Forest Classifier. The two Ordinary Least Squares linear models were imported from statsmodels [4]. The other two models were imported from scikit-learn [5].

4.1 Ordinary Least Squares

The Ordinary Least Squares linear regression model is used from statsmodels. This model minimizes the sum of squares between the predicted training labels based on the training features and the actual training features. The output of the model gives us a floating point number, and if this number is less than 0.5, we predict that the team does not win (win=0). Otherwise, we predict that the team wins the game (win=1).

4.2 Polynomial Least Squares

This model is similar to the OLS model, but with one change. Using the PolynomialFeatures function from scikit-learn, each feature is multiplied to each other individual feature. Since there are 3 features in our original model (elo_i, opp_elo_i, and home_game), our new features are elo_i, opp_elo_i, home_game, elo_i * opp_elo_i, elo_i * home_game, and home_game * opp_elo_i. Then, we fit

the same OLS linear regression model from statsmodels to the new feature dataset and the same labels.

4.3 K-Nearest Neighbors Classifier

This model is taken from the KNeighborsClassifier function in scikit-learn using k=25. Since K-Nearest-Neighbors is a form of unsupervised learning, this model is not trained. Instead, the training data finds the 25 closest data points based on Euclidean distance using the features. Using these 25 data points, it computes the most common label value (win = 1 or win = 0) and uses this as the prediction.

4.4 Random Forest Classifier

This model is taken from the RandomForestClassifier function in scikit-learn. This model randomly generates multiple decision trees and gets the most common resulting label value from these trees (win = 1 or win = 0). This method is the most complex model of the 4 used in this project.

5 Experiments and Discussion

Since this was a classification model, the testing accuracy was evaluated by obtaining the percent of data points in the testing set in which the predicted labels were the same as the true labels. The dataset was split into 75% training data (47353 samples) and 25% testing data (15785 samples). This was done using the train_test_split function from scikit-learn. Results for the final models after splitting the dataset using a random state of 42 are provided below.

Model Name	Testing Accuracy
forecast (Baseline)	0.6795058599936649
Ordinary Least Squares	0.6800126702565727
Polynomial Ordinary Least Squares	0.6803927779537535
K-Nearest-Neighbors Classifier	0.6304719670573329
Random Forest Classifier	0.6806461830852075

The models were re-run for 50 iterations with different, random train-test splits each time. The average testing error for each of these models are shown below.

Model Name	Testing Accuracy
forecast (Baseline)	0.6812809629394995
Ordinary Least Squares	0.6809946151409565
Polynomial Ordinary Least Squares	0.6814216027874564
K-Nearest-Neighbors Classifier	0.6268685460880583
Random Forest Classifier	0.6788862844472601

Examining the results of these models, there are a few clear patterns. First, the testing error of the K-Nearest Neighbor model seems to be very low. This may be too simple of a model to predict the results. It is unclear if our value of k=25 is too high and therefore has too high of bias, or if it is too low and it has too high of

variance. Regardless of if either is true (or if neither is true), the K-nearest-neighbors model using k=25 is not a good model to predict which team wins the game.

Next, the Random Forest Classifier does an alright job of predicting the victor of a game. In some circumstances, such as the first testing split, the random forest classifier can have the highest testing accuracy of all of the models. However, on average, its testing error is lower than the baseline model. This likely means that hyperparameters should be tuned to eliminate variance, which seems to be an issue with this model. If this cannot be achieved, it is possible that the Random Forest Classification model is too complex for our dataset.

The Ordinary Least Squares linear model is the simplest of our models, and it performs rather consistently. However, it tends to be very similar to the baseline model. Perhaps this means that the baseline model was computed using a similar model, and since linear classification is the most widely used classification model, it is very likely that is true.

Lastly, the Polynomial Ordinary Least Squares model performed the best. It is the only model that consistently outperformed the baseline model. The testing error was almost always above .68, while most of the other models struggle to get past that threshold.

6 Conclusions and Future Work

This project uses a few models to attempt to predict winners of professional basketball games based on ELO and the location of each game. Each model (Ordinary Least Squares, Polynomial Ordinary Least Squares, K-Nearest-Neighbors, and Random Forest) is fit to training data in order to have the best percent correct predictions for the testing data. Out of these machine learning techniques, the best results came from the Polynomial Ordinary Least Squares model, with an average of 0.6814216027874564 testing accuracy over 50 different train-test splits. This testing accuracy is good, since it is higher than the original predictions by FiveThirtyEight. But if the project were researched more in-depth, this prediction accuracy could be better.

Future work could include many possibilities. One is the possibility of using the ELO trend from the last 5 games. If a team is on a hot streak, maybe they are harder to beat than their ELO suggests. Another possibility is bringing in feature data from outside the dataset being used. If feature data like offensive and defensive data were used to complement the ELO data, the prediction models might have a more complete understanding of how the result of NBA games is caused by both team success and statistical performance.

References

[1] Silver, Nate. “The Complete History Of The NBA.” FiveThirtyEight, 7 Dec. 2015, projects.fivethirtyeight.com/complete-history-of-the-nba/#lakers.

[2] Silver, Nate. “How We Calculate NBA Elo Ratings.” FiveThirtyEight, FiveThirtyEight, 21 May 2015, fivethirtyeight.com/features/how-we-calculate-nba-elo-ratings/.

[3] Silver, Nate. “Nba-Elo.” GitHub, 2015, github.com/fivethirtyeight/data/tree/master/nba-elo.

[4] https://scikit-learn.org/stable/, *scikit-learn Machine Learning in Python*.

[5] https://www.statsmodels.org/stable/index.html, *statistical models, hypothesis tests, and data exploration*.