Best Method for Predicting Winner in Chess

Heewon Oh

301268860

Problem Statement and Background

Chess, a centuries-old timeless board game, has been played online for several decades. As with most games, chess players, reviewers, and audience-members often try to predict the winning player. It is widely known that the white-position is slightly more favorable (when compared to that of black) due to the first-move advantage.

Nevertheless, with the explosive growth of online chess, we would like to determine the most appropriate classifier to predict the winner of matches based on colour on a popular platform called Lichess by comparing accuracy on cross-validated data. Thus, we must factor in the advantage white holds over black as well as information regarding the match and the players (i.e. rating).

This is important to chess players looking to improve as well as place bets on matches. Having an accurate predictor would allow people to realize potential profit from gambling and track individual performance of players.

On a website called Medium, we found that another student attempted a similar project using the same dataset. He found that Random Forest and Voting Classifier produced the best results, with accuracy around 0.63 and precision 0.63 for Random Forest. However, he conducted a more complex model that including draws; for simplicity, we eliminated draws.

Data

	Chess_Data (N = 19,108)
Turns	
mean (sd)	59.19 ± 32.31
min	1
max	349
Rated	
mean (sd)	0.81 ± 0.39
min	0
max	1
White Rating	
mean (sd)	1,593.70 ± 289.95
min	784
max	2700
Black Rating	
mean (sd)	1,586.23 ± 290.15
min	789
max	2723
Opening Play	
mean (sd)	4.81 ± 2.78
min	1
max	28
Winner	
mean (sd)	0.52 ± 0.50
min	0
max	1

Figure 1: Data Summary

The data we used was obtained from Kaggle. The creator of the dataset compiled the data using the Lichess API. The dataset initially contains 16 variables and 20058 observations. After filtering out draw outcomes, we are left with 19108 observations. Of the 16 variables however, most were unusable for our methods as they contained background information (i.e. time), or when converted to factors, contained too many levels. Prior to modifications, the dataset originally contained mostly a mix of character and numeric variables, with just a logic variable for rated.

Methods

From the dataset, we used all observations excluding draws; 70% of observations were assigned to our training set and the remaining 30% to our testing set. We did this because we believed that the size of the data would be sufficient and not overwhelming as none of the classifiers that we wanted to attempt had higher computation time. We initially started the project with the objective of using smaller sampled training and testing set as we wanted to include neural net results, but because of difficulties with creating an appropriate confusion matrix and accuracy results, we abandoned neural nets. We also encountered some similar issues with KNN and similarly decided not to pursue it (see Appendix Part C for neural net and E for KNN).

Initially, we sought to include draw-results; however, between difficulties with various methods (as the result would no longer be binary, but rather 3-level factor or 3 integers), we opted not to add draws. Furthermore, draws were not imperative to our initial research question.

For logistical regression, lda, qda, and naïve Baines classifier methods, we edited the dataset to change class-types for the variables, removed draws outcomes, and replaced the victory status variable with three variables: mate, resign, and out of time. We then selected rated, turns, both ratings, number of opening plays, mate, resign, out of time, and winner as variables for the applied dataset to use to construct our training and test datasets. We did this to include the variable in our methods in a useful format. Also, for these methods, we ran two simulations: one with just the two player's ratings and the number of turns, as well as one that included. We did this because when we ran logistic regression, ratings and turns were significant, and mate showed weak significance.

For the Tree-based methods: bagging, boosting, and random forest, we changed all variables to factor classes, filtered out draw outcomes and categories that had more than 53 outcomes. To do this, we also grouped results in rating and turns. We know that tree tends to have poor results, including in the previous paper with 3 outcomes; it also has a maximum 32-level factor limit so we chose not to use it as it would unnecessarily complicate our methodology. We then drew new training and testing datasets. We believed, given the size of the datasets, this will not effect results significantly.

Results

We used accuracy, precision, and recall measure from cross-validated data. Runtime was not a concern in this experiment due to the size and nature of the dataset and methods. We primarily value accuracy; however, we measure precision and recall in case there is a very high value.

```
      method          accuracy  precision recall
 [1,] "Logistic"      "0.65719" "0.72353" "0.65821"
 [2,] "Log w Mate"    "0.65719" "0.72353" "0.65821"
 [3,] "LDA"           "0.65649" "0.73183" "0.65586"
 [4,] "LDA w Mate"    "0.65527" "0.73282" "0.6548"
 [5,] "QDA"           "0.6523"  "0.70727" "0.66078"
 [6,] "QDA w Mate"    "0.63852" "0.69565" "0.66078"
 [7,] "Naive Baines"  "0.63608" "0.80584" "0.62018"
 [8,] "NB w Mate"     "0.6419"  "0.80053" "0.61894"
 [9,] "Bagging"       "0.6419"  "0.63358" "0.75614"
[10,] "Random Forest" "0.65725" "0.62471" "0.79861"
[11,] "Boosting"      "0.65719" "0.66677" "0.69575"
```

Figure 2: Results

Unfortunately, this report was limited in scope due to restrictions in resources, time, and data-science fluency by the author. Nevertheless, we conclude that amongst the methods we were successful in testing, random forests are the most appropriate method in predicting the winner. This is because it has the highest accuracy. We must acknowledge however that because the training and datasets had to be redrawn due to the changes in format for the tree-based tests, including random forest. Thus, Logistic regression has, within a reasonable margin for error, potential to outperform or equally perform the random forest method. Thus, we can not confidently conclude that random perform outperforms logistic regression here with certainty, but such is our results.

Moreover, sometimes, when gambling precision or recall may be more important when accuracy is similar (due to odds). Thus, when it is important to note that while Naïve Baines with Mate, and logistic regression (equally with or without mate), LDA(both with and without mate), perform similarly to random forest in accuracy, but have significantly higher precision(especially Naïve Baines). Nevertheless, random forest also has the highest recall. Thus, overall – random forest is the best model for predicting the winner and our current model has ~ 65.7% accuracy. However, if someone wanted a method that best balances accuracy, recall and precision results, logistic regression is recommended.

References:

(n.d.). Chessgames.com: Chess Games Database & Community. Retrieved December 10, 2021, from https://www.chessgames.com/chessstats.html

How to round up to the nearest 10 (or 100 or X)? (n.d.). Stack Overflow. Retrieved December 10, 2021, from https://stackoverflow.com/questions/6461209/how-to-round-up-to-the-nearest-10-or-100-or-x

J, M. (n.d.). *Chess Game Dataset (Lichess)*. Kaggle: Your Machine Learning and Data Science Community. Retrieved December 9, 2021, from https://www.kaggle.com/datasnaek/chess

K, A. (2021, April 12). *How to predict an outcome of a chess game?* Medium. https://achyutk23.medium.com/how-to-predict-an-outcome-of-a-chess-game- c4f452362575