ChessQuery

Computational Data Science

Devin Lu, Edmond He

Fall 2021

1 Introduction.

The well-known, popular game of chess is often described as a game of perfect information between two players. There is no luck involved, and players take turns moving one of their pieces back and forth with the objective of capturing the opponent’s king.

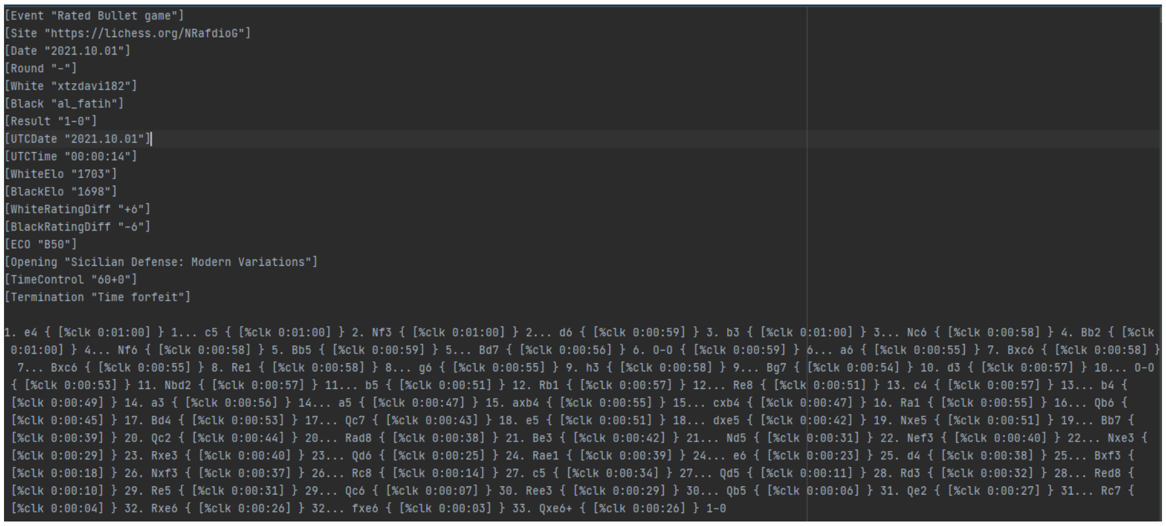
In this project, we’re going to explore the meaning of ELO ratings and win rates in chess. ELO is a rating system used in games for matchmaking.

In online chess, the main way people play chess is with time controls. A ‘blitz game’, for example, is a format where each player has 5 minutes to play out their moves throughout the game. If they run out of time, they lose (with few exceptions). Another example would be classical (tournament) games, where the time control is much longer (often hours long).

We will be using games from October 2021 on Lichess.org. The main packages that we used were pandas and sci-kit for data processing, and matplotlib for visualizations. There are some interesting questions we can try to answer.

- What is the rating distribution of players
- Do certain moves played affect win rate?
- Is it possible to train a ML model to predict the outcome of a game?

2 Data.



```
[Event "Rated Bullet game"]
[Site "https://lichess.org/NRafdioG"]
[Date "2021.10.01"]
[Round "-"]
[White "xtzdavi182"]
[Black "al_fatih"]
[Result "1-0"]
[UTCDate "2021.10.01"]
[UTCTime "00:00:14"]
[WhiteElo "1703"]
[BlackElo "1698"]
[WhiteRatingDiff "+6"]
[BlackRatingDiff "-6"]
[ECO "B50"]
[Opening "Sicilian Defense: Modern Variations"]
[TimeControl "60+0"]
[Termination "Time forfeit"]

1. e4 { [%clk 0:01:00] } 1... c5 { [%clk 0:01:00] } 2. Nf3 { [%clk 0:01:00] } 2... d6 { [%clk 0:00:59] } 3. b3 { [%clk 0:01:00] } 3... Nc6 { [%clk 0:00:58] } 4. Bb2 { [%clk
 0:01:00] } 4... Nf6 { [%clk 0:00:58] } 5. Bb5 { [%clk 0:00:59] } 5... Bd7 { [%clk 0:00:56] } 6. O-O { [%clk 0:00:59] } 6... a6 { [%clk 0:00:55] } 7. Bxc6 { [%clk 0:00:58] }
  7... Bxc6 { [%clk 0:00:55] } 8. Re1 { [%clk 0:00:58] } 8... g6 { [%clk 0:00:55] } 9. h3 { [%clk 0:00:58] } 9... Bg7 { [%clk 0:00:54] } 10. d3 { [%clk 0:00:57] } 10... O-O
 { [%clk 0:00:53] } 11. Nbd2 { [%clk 0:00:57] } 11... b5 { [%clk 0:00:51] } 12. Rb1 { [%clk 0:00:57] } 12... Re8 { [%clk 0:00:51] } 13. c4 { [%clk 0:00:57] } 13... b4 {
 [%clk 0:00:49] } 14. a3 { [%clk 0:00:56] } 14... a5 { [%clk 0:00:47] } 15. axb4 { [%clk 0:00:55] } 15... cxb4 { [%clk 0:00:47] } 16. Ra1 { [%clk 0:00:55] } 16... Qb6 {
 [%clk 0:00:45] } 17. Bd4 { [%clk 0:00:53] } 17... Qc7 { [%clk 0:00:43] } 18. e5 { [%clk 0:00:51] } 18... dxe5 { [%clk 0:00:42] } 19. Nxe5 { [%clk 0:00:51] } 19... Bb7 {
 [%clk 0:00:39] } 20. Qc2 { [%clk 0:00:44] } 20... Rad8 { [%clk 0:00:38] } 21. Be3 { [%clk 0:00:42] } 21... Nd5 { [%clk 0:00:31] } 22. Nef3 { [%clk 0:00:40] } 22... Nxe3 {
 [%clk 0:00:29] } 23. Rxe3 { [%clk 0:00:40] } 23... Qd6 { [%clk 0:00:25] } 24. Rae1 { [%clk 0:00:39] } 24... e6 { [%clk 0:00:23] } 25. d4 { [%clk 0:00:38] } 25... Bxf3 {
 [%clk 0:00:18] } 26. Nxf3 { [%clk 0:00:37] } 26... Rc8 { [%clk 0:00:14] } 27. c5 { [%clk 0:00:34] } 27... Qd5 { [%clk 0:00:11] } 28. Rd3 { [%clk 0:00:32] } 28... Red8 {
 [%clk 0:00:10] } 29. Re5 { [%clk 0:00:31] } 29... Qc6 { [%clk 0:00:07] } 30. Ree3 { [%clk 0:00:29] } 30... Qb5 { [%clk 0:00:06] } 31. Qe2 { [%clk 0:00:27] } 31... Rc7 {
 [%clk 0:00:04] } 32. Rxe6 { [%clk 0:00:26] } 32... fxe6 { [%clk 0:00:03] } 33. Qxe6+ { [%clk 0:00:26] } 1-0
```

The number of games on Lichess are 88,092,721 for the month of October. That's a lot! The compressed file from Lichess' database is about 24 GB, but the uncompressed version is almost 200 GB.

Above we can see that the data is stored as a PGN file. It's the main way of sharing a chess game. We need to convert this to a more usable type first, as Pandas doesn't contain a neat read_pgn() function.

Our convert.py module does exactly this.

Unnecessary information, such as the key with 'Opening' conveniently also has a code which refers to exactly the same thing. The list of moves played are frankly the largest part of this file, so that will be downsized.

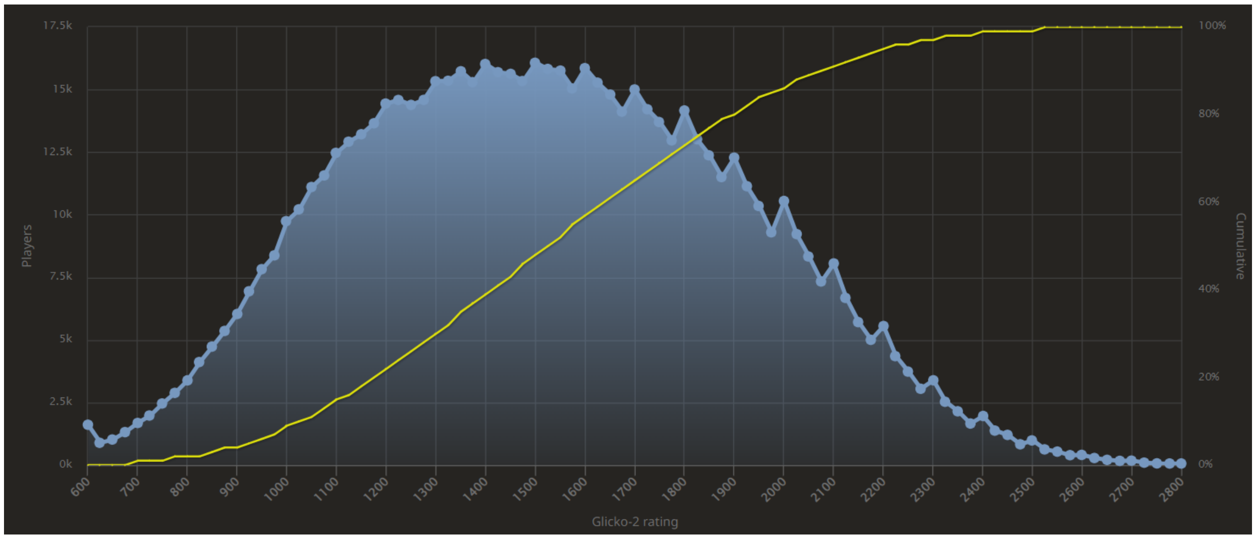
After downsizing, the file is still about 7.5 GB. This is a bit hard to work with, so we're only using 1 million games from October instead.

Data from Lichess can be obtained from https://database.lichess.org/. Lichess also has its own published API that we used to send GET requests, but the majority of our work was based off of the 7.5 GB file.

3. Information using data plots

3.1 Ratings of Players.

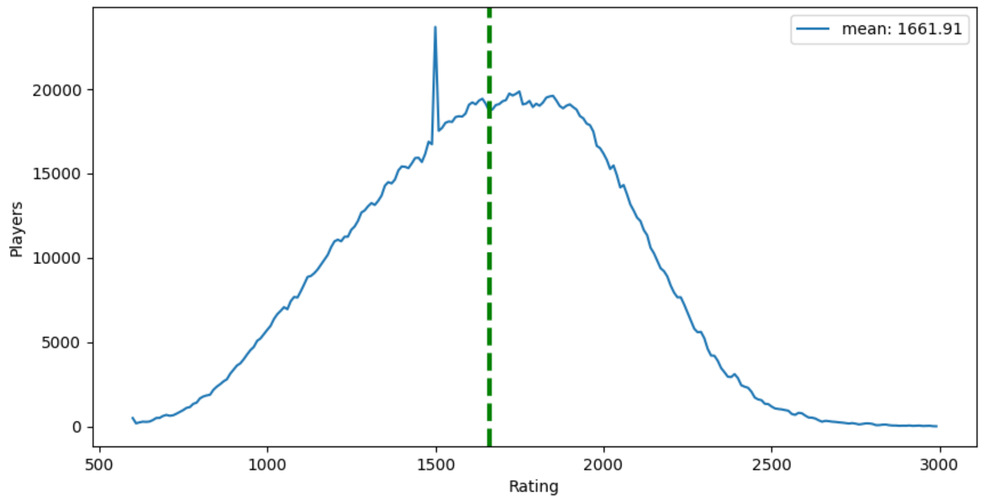
From Lichess's website: Blitz is their most popular format. The rating distribution of all their players is kind of a bell curve around 1500 ELO - that's the rating that new players start with. The graph below from Lichess.org shows their distribution of all their players, and we can see it's at about 1500 from the cumulative distribution.



Img src: https://lichess.org/stat/rating/distribution/blitz

However, the average rating of games played is probably not 1500 - players with higher elo will probably tend to play more games. Also notice how the distribution seems to spike every 100 ELO. Players seem

From the games in October 2021, the distribution is as follows:

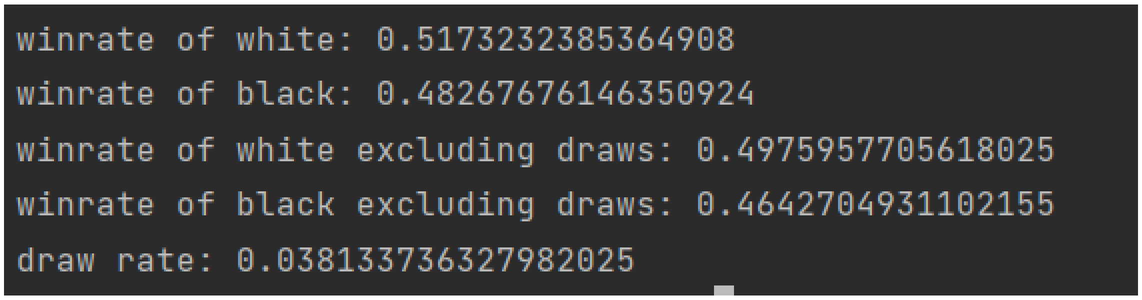


What does this mean? Well, if you're an active player on Lichess, you're about 162 ELO better than the average player.

3.2 Rating Gap

What does the rating gap mean? If you have an x rated player and an x - n rated player, what are the chances that the x rated player will win?

Well firstly, let us see the results of the games analyzed:



```
winrate of white: 0.5173232385364908
winrate of black: 0.48267676146350924
winrate of white excluding draws: 0.4975957705618025
winrate of black excluding draws: 0.4642704931102155
draw rate: 0.038133736327982025
```

Notice how white has a small edge over black. This is because white goes first, which is actually somewhat significant in chess.

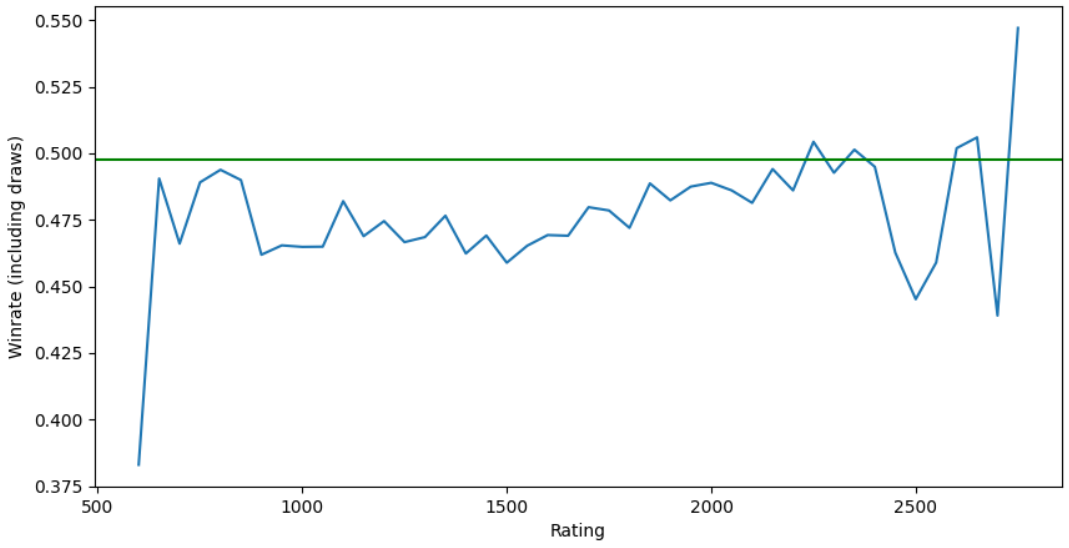
3.3 Playing F3

In the words of Ben Finegold, a popular chess Grandmaster: Never play f3.

f3 is when you move your pawn on the f column forward by one square.

In chess, f3 is generally a weakening pawn move that exposes the king for attack. Suppose that white played f3. Does this have a significant impact on what the outcome of the game is? Remember that from earlier, white wins 51.7% and black wins 48.3% of the time. A green line is used to show what the normal win rate is.

Notice how the win rates seem to skyrocket or plummet near the ends. From the player distribution earlier, there’s not too many people at 600 or above 2700. Data near these ends may be a bit inaccurate.



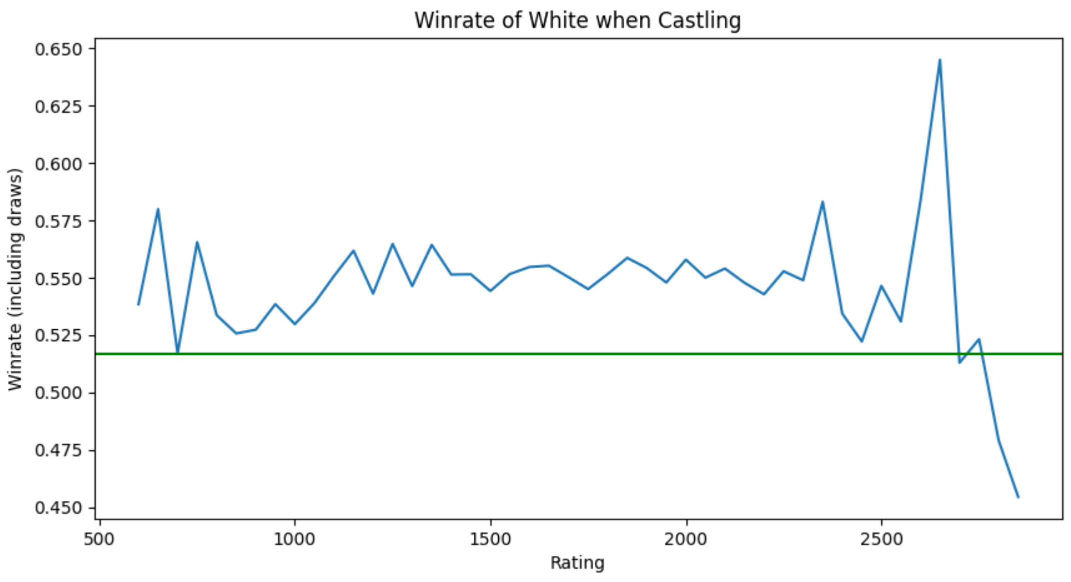
When playing f3, it seems that at lower elo ratings, you would generally have a lower win rate. However, it does seem to increase a bit when the elo rises. Higher rated players would probably know when is the right time to play this risky move for an advantage.

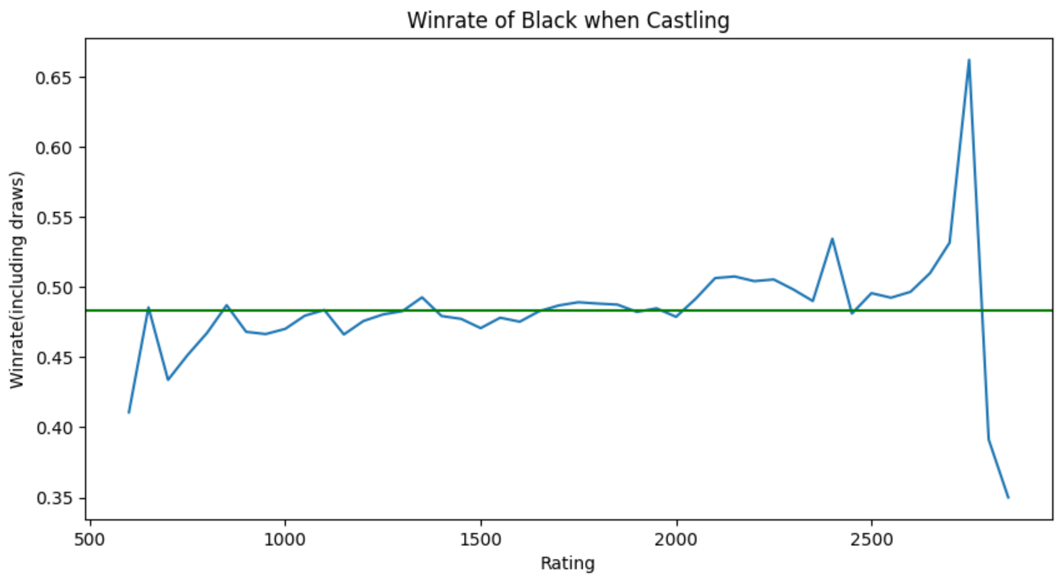
3.4 Castling

Castling is the only time in chess where someone can move two pieces in one move. This moves the king into safety while allowing one of the rooks to become more active, and allows one to solidify their position.

Is there an impact on the outcome of the game whether or not one side castles? Likewise from 3.3, a green line is used to show the normal winrate.

The following graphs show the win rates when either black or white castles, and the other side doesn’t.





It does seem that castling has a positive win rate. Once again, the win rates skyrocket along the extremes, so they aren't too indicative of the results.

Once again, it seems that the winrate minorly grows when the rating increases. Castling can sometimes be a mistake, as it could mean that a player misses a key tactical move, so it could mean that higher rated players aren't as tunneled in on castling when they should.

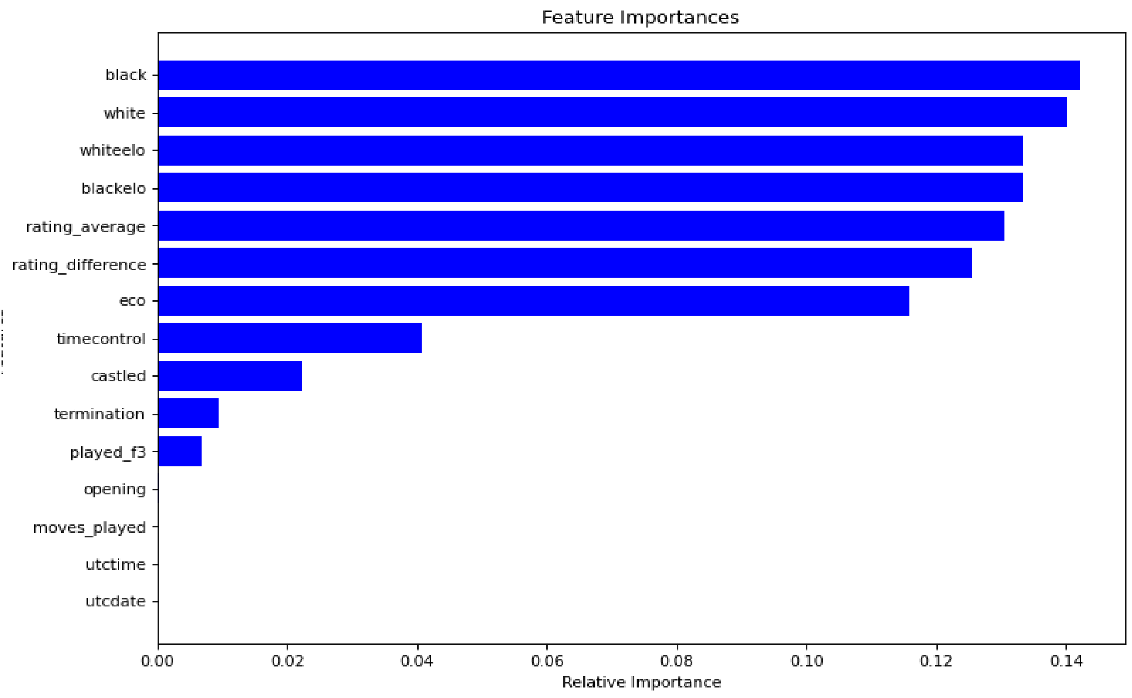
4. Machine Learning

4.1 Introduction

After some data exploration, we are now interested in applying machine learning models/techniques to see how well we can predict the outcomes of chess games. We aim to train a couple different machine learning models with our dataset from Lichess.

4.2 Feature Selection

From our dataset, we have a lot of unnecessary columns like 'utctime' for example (which is when the game took place). Clearly, this can be removed when funnelling our data into our model. However, we would like to know the *importance* of each feature. Therefore, by training a Random Forest Classifier, we are able to get the feature importances of each column with the `feature_importances_` attribute. Here is how it looks:



Here, we can see that “utctime” or “utcdate” weren’t really used well in the prediction for our model. We can filter this out. Additionally, we filter out “black” and “white”, since these are just the player names. Finally, we also filter out rating_average, and rating_difference since these are derived from the ELO ratings of both players. After these filters, we now work with “whiteelo”, “blackelo”, “eco”, and “timecontrol”.

4.3 Model Training

After feature selection, we can now filter out the dataset for only the input columns that we’re interested in. In addition, we encoded all values into numerical data where needed. The models that we will be creating are:

- Random Forests, since ensemble models generally give out good predictions.
- MLP, since we believe a neural network would be useful to model more complex, non-linear relationships. We also have a lot of data points which are useful for a neural network.

- Gaussian Naive Bayes, since the input features are independent from each other.

From using the default parameters of all the models, we now funnel the data into each model respectively. The results are somewhat expected:

GaussianNB	0.509795
Random Forest	0.51763
MLP	0.535515

4.4 Hyperparameter Tuning

From our results above, it's clear that our models aren't quite accurate. This makes sense, since we're only using the default parameters for each model. Therefore, we need to hypertune the parameters (for MLP and random forest) by using Scikit's GridSearchCV function, as well as iteratively testing out different parameters. The result for random forest and MLP are now:

Random Forest (Hypertuned)	0.54314
MLP (Hypertuned)	0.5356

Here, we can see that our MLP didn't do too well with the adjusted parameters, but our random forest increased by 3%. Although there are probably more optimal parameters to be adjusted, it is also worth noting that we are a bit limited with the data provided by Lichess (as we aren't using too many input features). Some other possible features that could improve our model could be hours of chess played by a player, their total number of games played, their win rate, and so on. But, these weren't available to us from Lichess' database. Not a bad increase though for random forests.

5. Conclusion

ChessQuery is a project aimed to explore the ELO ratings and other similar data statistics in chess. As avid chess competitors and enthusiasts, this was a very interesting and cool data science project that allowed both of us to apply our domain knowledge with our technical, programmatic skills to explore the board game that we love. In the future, we would love to incorporate more input features to our model to increase the accuracy further, and perhaps implement more deep neural networks from a library such as Tensorflow, where deep learning is the bread and butter.

5.2 Problems and Limitations

One slight problem was the difficulty of extracting data from the Chess.com website. Even though it's the largest online chess website and would be very useful to run our analyses on, they aren't open source compared to Lichess. Games can only be extracted via player ID, which is not too useful to the goal of our project.

6. Project Experience Summary

Devin Lu:

- Utilized HTTP requests to extract data from Lichess' published API.
- Applied different machine learning techniques such as feature selection to accurately train multiple different models.
- Hypertuned parameters to gain an accuracy increase of 10%.

Edmond He:

- Processed data extracted from Lichess' database and converted file format from PGN to CSV.
- Analyzed ELO ratings and chess data by applying various plot visualizations using matplotlib.