Aaron Shabanian

Matthew Olivo

Julian Lange

Chess Analysis Writeup

Chess Analysis Writeup

Intro

The Problem

Chess is one of the most complex games with thousands of different moves that can be completed in each game. This is a game that has been studied by both the world's smartest people and computers. Our mission was to try to gain insight and make predictions on Chess data using a variety of Deep Learning models and Data Science methods.

Dataset

The dataset that we are using is the chess game dataset from the kaggle website. It contains 20,000 games collected from a selection of users from the site lichess.org using the lichess api. The information that it contains are: Start time, end time, number of turns per game, which side was the winner of that game, game status, time increment, white and black player rating, all moves in standard chess notation, number of moves in opening phase, opening eco which is the standardized code for any given opening, the opening name, and white/black/game ID.

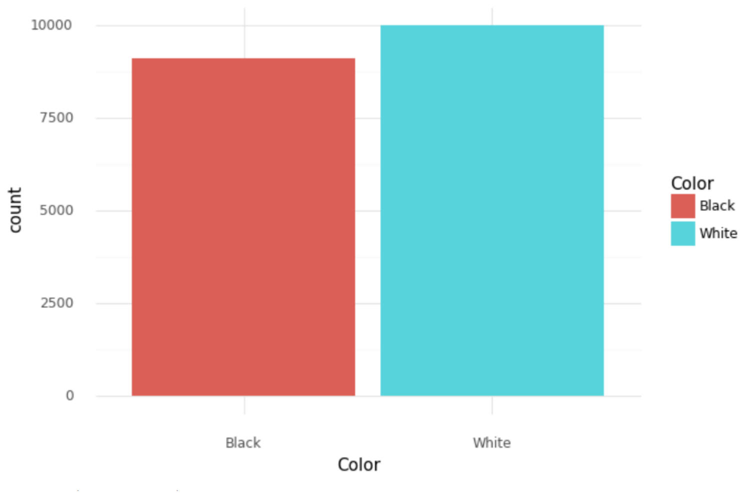
Background

Chess has variations dating back to as old as 600 A.D. Our project focused on what is referred to as "Western Chess or "International Chess" which was created during the 15th century in Southern Europe. The entire ruleset of the game are a bit complex to explain however each piece can make a variety of different moves under certain circumstances. A winner is declared when the opposing players' king piece is in what's called "check" and cannot resolve this state on the following turn. In 1985 the first chess intelligence began development at Carnegie Mellon University. This project was later picked up by IBM and given the name "Deep Blue". Since this historic Chess Engine made headlines, there's been an ongoing competition between large tech companies as to who can make the strongest Chess engine.

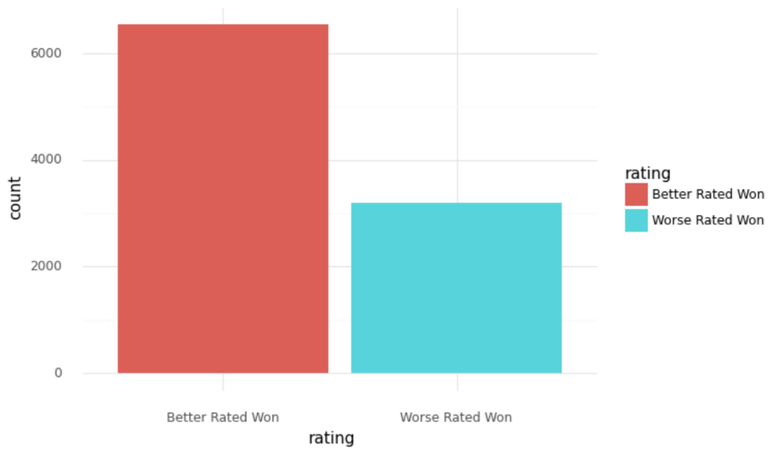
Findings

Data Exploration

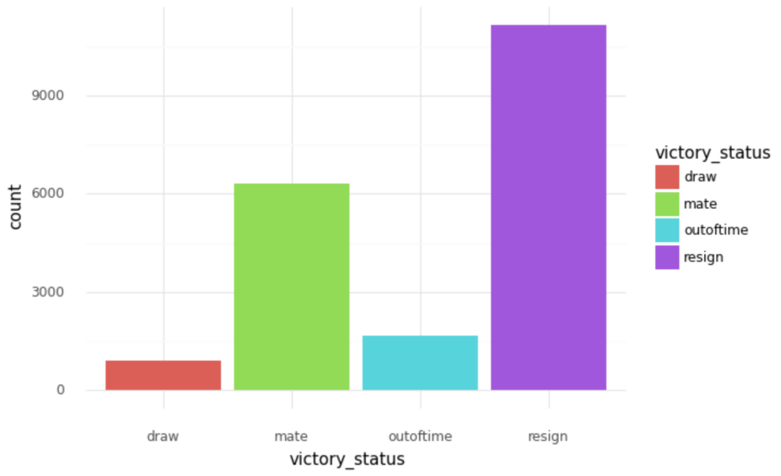
To start our chess analysis, we wanted to observe to see if different traits can increase the likelihood of winning. The first trait that was examined was to see if the color of the player (white or black) increases the chance of winning the game.

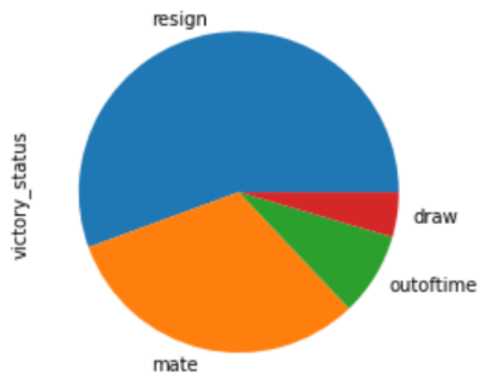


There was a very slight advantage given to the white player which does make sense since they go first. It was interesting to see how the advantage between the two is small when the chess community says that white usually has a huge advantage. This is mostly because white can freely do any opening move while black has to adapt to their opening. Next, the rating of a player was examined to see if it could be a viable way of predicting the winner.

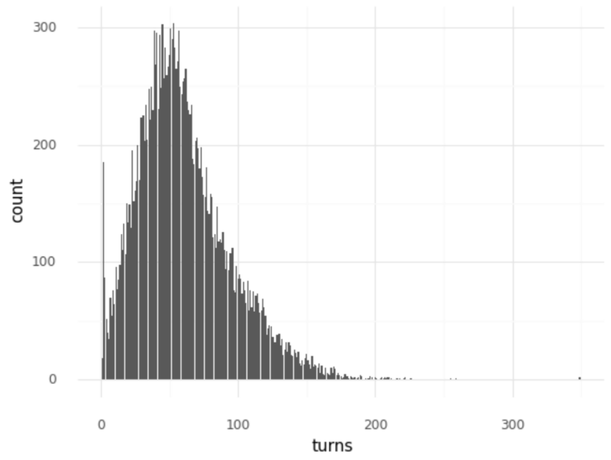


67% of players with the higher rating ended up winning the game. This shows that the elo ranking system used by lichess is actually pretty good at predicting who is a better player. Going through the victory status feature for the lichess dataset, we created two visualizations, a bar graph and a pie chart.





We can see from the two, that resign has the biggest number of results from the 20,000 games collected. 55.57% of the victory status for players were from resigns. This is probably due to most players resigning when they are at a point of disadvantage where they cannot come back rather than letting themselves get checkmated at the end for quicker games. 31.53% were checkmates. 8.38% was either black or white running out of time and lastly 4.52% were draws. Last exploration was the average amount of turns per game and the average turns needed for each player's opening phase per game.



Looking at the bar graph, the average was 60 turns per game and we calculated that an average of 4 to 5 moves were used for each player's opening phase.

Predicting the winner

One of the first models that we decided to create was to attempt to predict the winner of the game with just the first couple of moves. I decided to build this model to see how much weight the beginning of a chess game held when trying to predict the winner.

For this model, I decided to use a support vector machine, as I didn't think there was enough information for it to work with a complete deep learning model like a feed forward neural network. The results showed that there was some weight on the first couple of games but it wasn't enough to assume the actual results of the game. The accuracy of the support vector machine was 53.9%. This is around 9% better than just guessing accuracy which is around 44.5% since not all games end with an actual winner. This makes sense because while the opening moves do have an impact on the game, there is a lot that has yet to happen and could influence the game one way or another.

Next, I tried to add the rating of both players to the SVM to see how much of an influence that would have on predicting the winner of the game. When adding the rating of both players the SVM accuracy skyrocketed to over 67%. This means that in a game of chess on Lichess.com, the winner of a game could be predicted over ⅔ of the time with just a small amount of information. That other ⅓ of the time likely occurs because of uncontrolled moves that happen after the beginning.

Predicting the rating

Being able to predict the rating of a player was a goal of ours because we wanted to see if there was a general trend intactix or game style in the high ratings. In order to accomplish this, we not only leverage linear regression models to predict the number associated with the players rating, but also created categorical ranges associated with beginner, intermediate, or expert players to see if rating could be predicted in that way using a logistic regression or a feedforward neural network.

For our logistic regression and feedforward neural network, we took the opening moves and number of turns that occurred in a particular match and attempted to predict the tier of the white player. No matter what range we set for each of the three categorical boundaries, the accuracy score of either model never exceeded guessing accuracy. In the case of the feedforward neural network, a persisting issue with the accuracy score never improving lead us to believe that there was simply not enough difference in the tactics played at lower ELO ratings than higher ones.

Our final model to predict a player's rating was a simple linear regression which took in the rating of the opposing player, the opening played, and the number of turns the match lasted. This model also did not perform with amazing accuracy, however it was able to establish a very strong and obvious correlation between the ELO rating of the white player and the ELO rating of the black player.

Predicting Number of Turns

Attempting to predict the number of turns a match lasted wasn't exactly one of the more important questions we were trying to figure out, however we felt as though it could further discover strong correlations between variables. To answer this question, we built a linear regression model which took in white ELO rating, black yellow rating, the opening that was played, and whether or not the match was rated. While the accuracy of the model wasn't noteworthy, the strong correlation that we found between a match lasting longer and the rating of the players in it as well as whether the match was rated or not was an interesting discovery. Intuitively, this finding makes sense as more skilled players will tend to draw the game out longer and make it more competitive, and if the game was for actual rating, it will tend to be taken more seriously and less early forfeits.

Predicting the best counter move

The goal of this model was to predict the best counter move for black to use after white went. This would be with whites move and blacks previous move as inputs and would output blacks predicted next move.

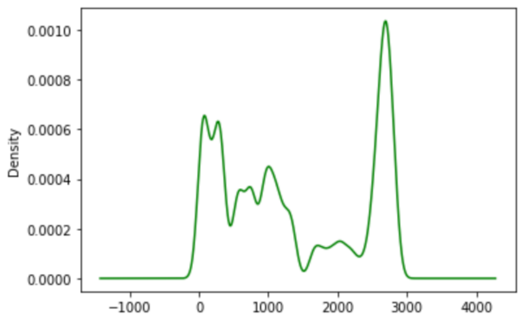
Before the data was ready to be fed into different models, the data had to be prepared. The original moves for each game were stored in one big string in standard chess notation. This all had to be divided so the individual moves could be analyzed. We did this by creating a separate data frame that had white's move, black's previous move and black's next move. The moves had to then be encoded into a numerical format so that it could be fed into the model. With all of the data separated and formatted correctly, it was now possible to feed it into the models.

Before creating a deep learning model, I wanted to see how it would perform on a normal support vector regression model. The inputs of "Move1Code" and "PreviousMove2Code" were used to predict "Move2Code". Move1 is white and Move2 is black. The support vector regression ended up with a mean absolute error of 983.3.

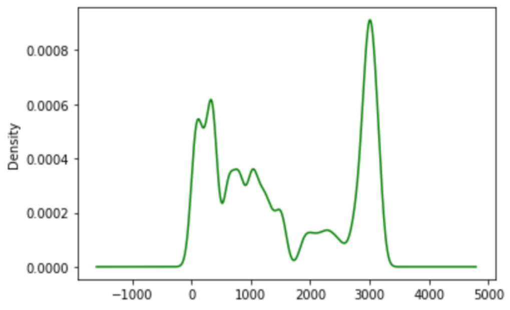
We then tried to improve upon this with a feed forward neural network. The inputs and outputs were the same as those in the support vector regression. The results of this model were not very good either with an accuracy of only .002%.

The inconclusive results led me to explore to see if there was much of a correlation between white's move and black's next move. The first step was to examine the distributions of the encodings for the moves of both black and white.

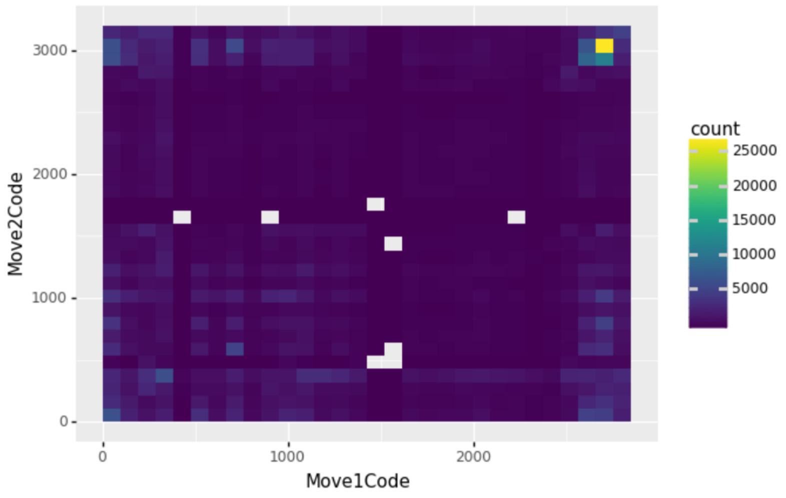
White Move Distribution



Black Move Distribution



As can be seen with this, there are over 3000 possible moves for both players. That number of moves might be too big of a feature space for the model to be trained on given that there are only 600,000 pairs of whites move to blacks move. Further examining the data, we looked at the correlation between white's and black's move



Looking at this "heat map" there does not seem to be a clear correlation from white's move to black's move. If there is a correlation, it may be too small to be noticed in this large feature space. There are many varieties in how a chess board can be laid out mid game and that changes the possible moves that black can use to retaliate with. Because of all of these possible different factors it may be impossible to predict one possible move out of 3000 just based off of the previous move. There are other possible models that can solve this problem that are discussed in the future experiments section of this paper.

Future Experiments

One example of a possible model that could solve the problem from before is an LSTM. In future experiments an LSTM could work better as it could limit the feature space and would take into account all previous moves rather than just rely on the previous move. This would work better as there would be less illegal moves since we know where all the pieces are. This would likely require a much bigger neural network as we would likely need a lot more moves. 20k different games is likely not enough games as there are a multitude of different games. There would also need to be a lot more computing power for all of this to work as I would have to divide every game into a subset of different data points to always predict the next move.

Most research with chess is currently done with images rather than just sequence and image processing. In the future it may be better to make a chess model with images instead of just a list of moves.

Conclusion

In conclusion, we were very ambitious in our goals for this project. We tackled a problem that big tech companies like IBM and Google are still trying to solve. It was interesting to learn how much goes into chess and how complex modeling it could be.