Chess Elo Guesser

Owen Badgley and Bryan Katuari

12/16/2025

Motivation

Our motivation for attempting to make a chess elo guesser as our machine learning project was mostly our combined interest in the game. We've both played chess for a while, and thought that it would be a fun and interesting problem to tackle.

Background

Ever since machine learning was invented, people have been designing models that could play games. People thought that a chess bot would never beat a human, and now the inverse is true. ChatGPT started as a bot that played Dota. Playing games is a prime training ground for machine learning experimentation, as the ruleset and strategy for games act as a clear focus and goal for developing models.

Related Work

As we mentioned before, there is extensive work related to chess in the machine learning space. Stockfish, which we used for this project, is a premier chess engine that actually plays the game, rather than just analyzing it. There is another project that did something similar to ours, which took a collection of someone's games and tried to guess their elo, rather than just one game such as our model takes. These two options are only scratching the surface of what chess, and games as a whole, have done for the field of machine learning.

Claim / Target Task

Our goal is to, given a chess game as represented by a PGN file, output a guess for white's elo and black's elo that is within 200 elo of the actual result. While this is difficult, we managed to make a model that could consistently reproduce this.

A single chess game is "noisy", players often perform better or worse than their elo, similar to how athletes sometimes play well or poorly. The distribution is wide, so a within 200 elo goal is our target given the variance.

Proposed Solution

Our solution uses a regression model over 4000 elo bins, and takes into consideration the proximity of those bins when making a final prediction.

To train this model, we took 5000 games from the lichess database (as .pgn files) and put them through preprocessing with stockfish to extract evaluations at each move.

This was then saved in a CSV file, where the model extracted the features it needed and used them to train.

Implementation

- We created a pipeline that took in a .pgn file, then extracted the move list and ratings from that file.
- With that move list, at every move the evaluation was calculated so as to give the model more informative and helpful training data.
- The ratings were used as labels for training the model.

Implementation cont.

- All of this was fed into the model, which used the average distance from the true elo to evaluate itself.
- Used 20 dimension variable vectors, an 80/20 training/validation split over 5000 total games, 25 epochs and 3 seeds.
- Also implemented early stopping if no improvement was shown in 4 epochs.
- Once the model is trained, a user can input a .pgn file that is then preprocessed in the same way as the training data, and then is fed into the model to predict an elo for each player.

Data Summary

Our training dataset is from the lichess open database, which is free to use for anything that people need it for. The data is distributed in sets of about 90 million games. We had to go through these games get a usable subset. After we got this subset, we went through and took only the standard games from it, as we thought that time pressure could change the way that ratings are displayed. The majority of players in this data are rated between 1200-1800, which heavily influenced the final model.

Code Walkthrough

Experimental Results

=== Summary over 3 seeds ===

Random Baseline ValMAE: W 432.2 B 446.8 ALL 439.5

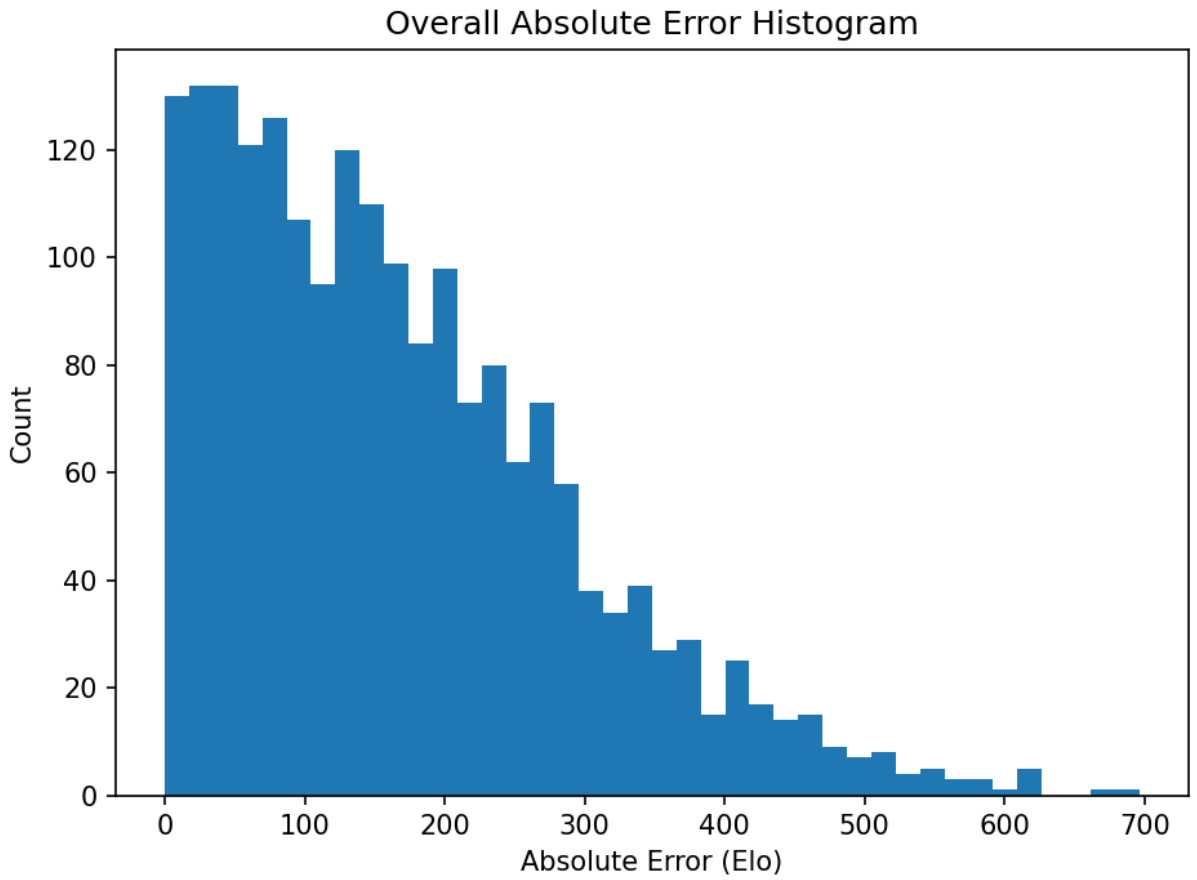
Strong Baseline ALL MAE: 207.0 ± 2.5

Model W MAE: 166.6 ± 1.1

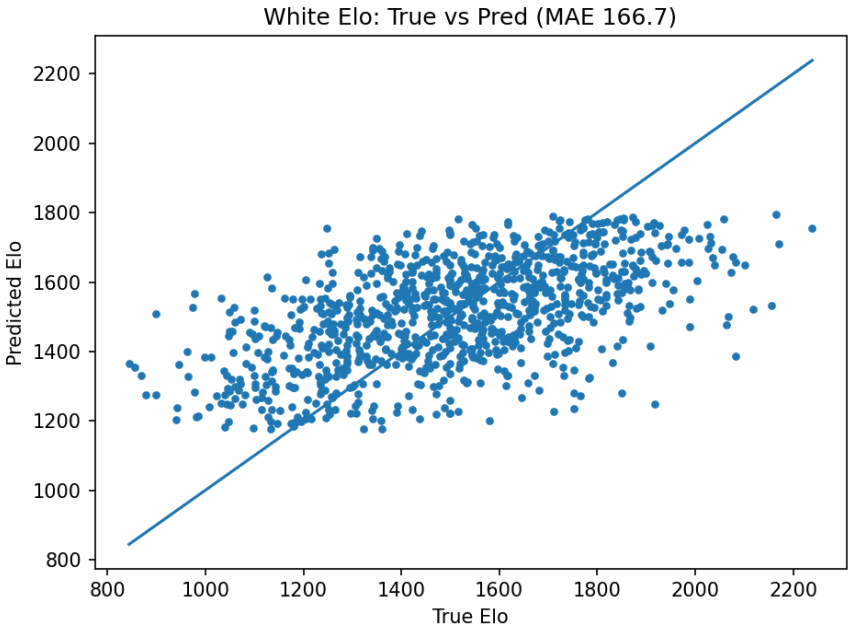
Model B MAE: 171.6 ± 3.0

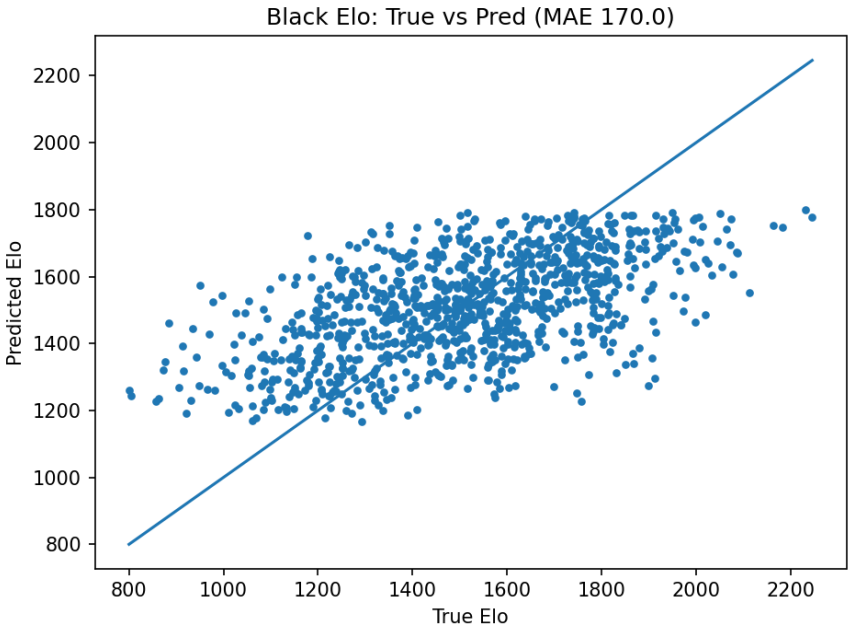
Model ALL MAE: 169.1 ± 1.0

Experimental Results cont.

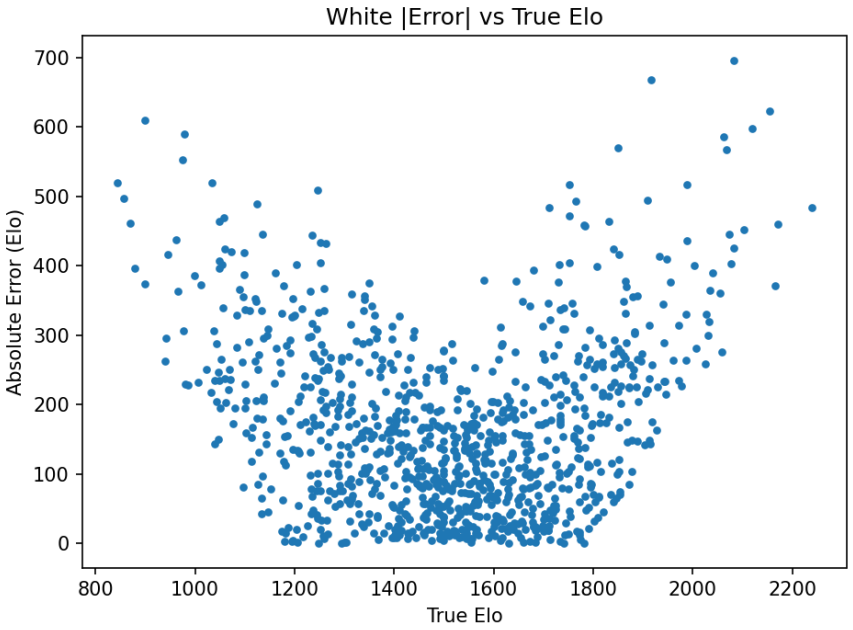


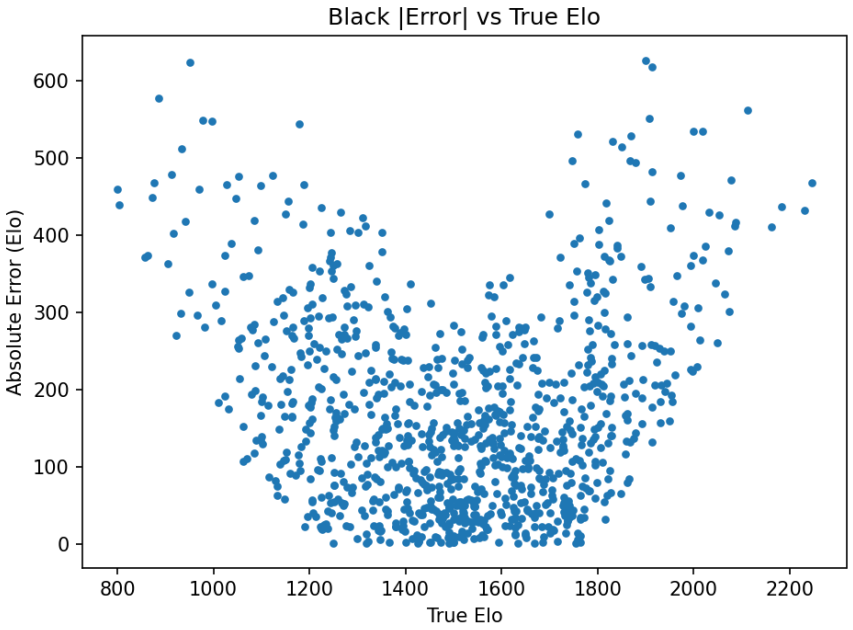
Experimental Results cont.





Experimental Results cont.





Experimental Analysis Overview

As shown by our results, as well as our histogram, the majority of predictions were within the 200 elo margin we were trying to hit, with an average distance of about 170 elo.

The distributions of predictions show the difficulty that the model had with predicting points at the extremes. With very little data to pull from for those extremes, the model was not able to accurately predict someone at, say, 2000 elo.

Conclusion and Future Work

- To further improve this model, we would likely need a significantly larger amount of data, or at least data that included a more even spread of elos. The challenge of this is that the way that elo works is that it is specifically designed to be a normal distribution, which will naturally pool most people in the center.

- We could also try to force the model to be more willing to make guesses in the extremes, but we couldn't get it to in the time we spent refining it.

References

https://database.lichess.org/

https://github.com/HliasOuzounis/Ai-Guess-the-elo?tab=MIT-1-ov-file

Links & Acknowledgments

Video Link: https://youtu.be/SgSRFI_cDYs

GitHub Fork Link:
https://github.com/bryankatuari/uva-machine-learning-25f-projects.git

Owen Badgley:
- Preprocessing Script
- Final Prediction Maker Script
- Video Recording

Bryan Katuari:
- CSV Extraction Script
- Training Script
- Graph Plotting Script
- Model Script