Predicting Chess Win Probability

For more information, please visit the Github repository.

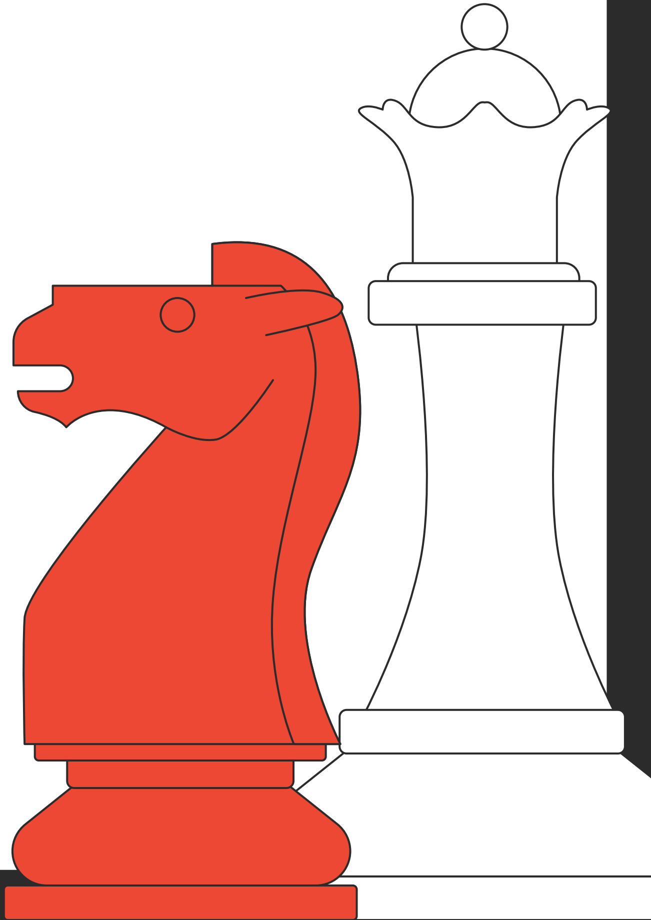
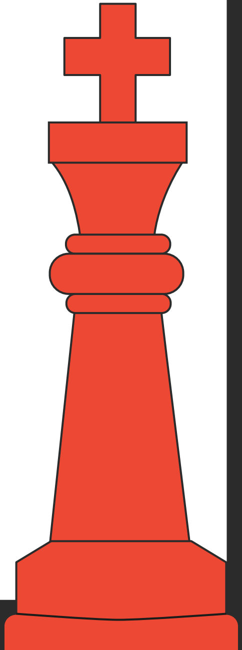
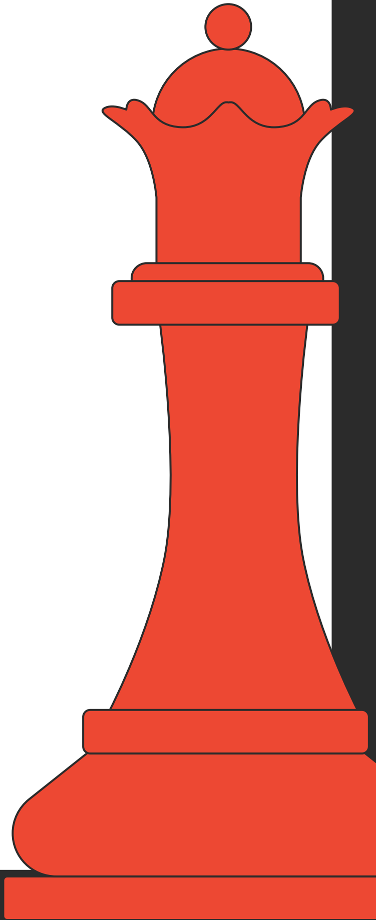


Table of contents



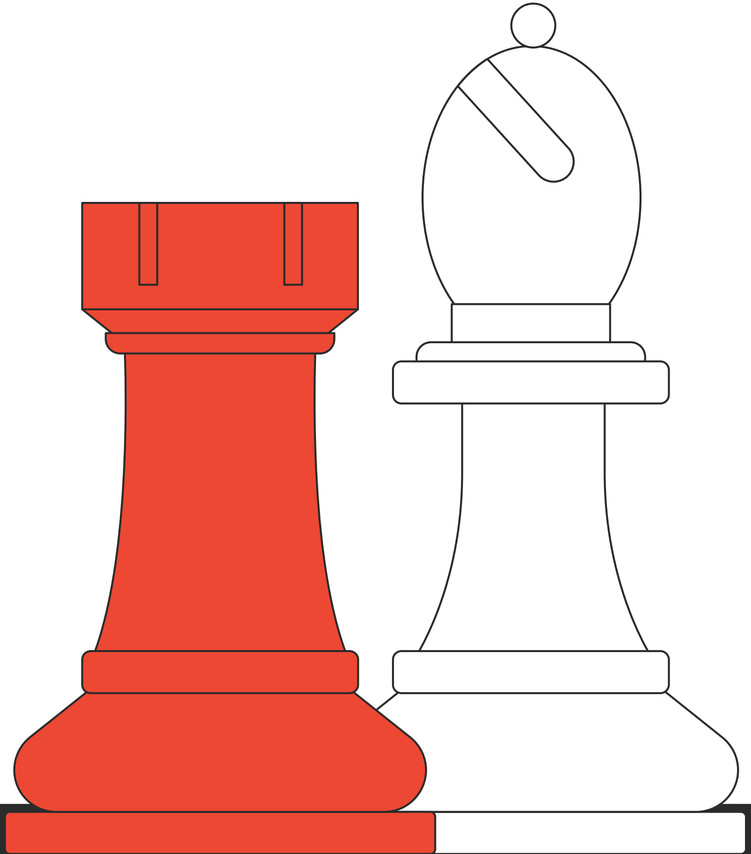
Motivation:

We aim to create a groundwork for
win probability models in esports.



Reasoning for Chess

- **Data**: Millions of chess games exist
  - Standard game notations ease the data cleaning process
- **Discrete Time**: Chess is turn-based, rather than played on a continuous timer
- **Chess Engines:** Many chess engines exist to extract quantitative features from the chess board
  - Our chosen chess engine is Stockfish.



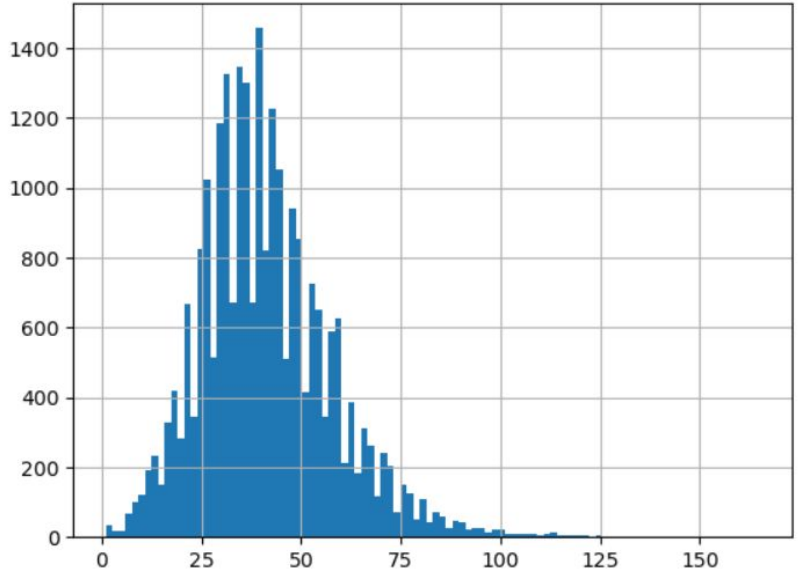
Exploratory Data Analysis

Our Data

- Initial dataset taken from Kaggle
- 25,000 games, each with:
  - **Elo** of White and Black
  - **Move Sequence**
  - Sequence of **Stockfish scores** for each move

Frequency of Length of Games

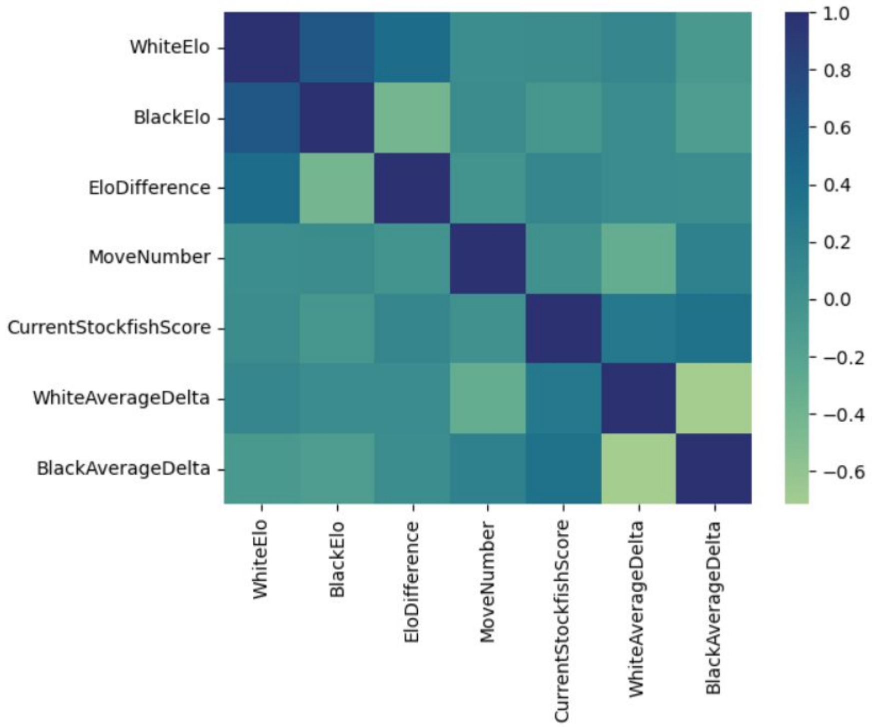
- Mean of **41 moves**
- Right-skewed
- When finding win probability at later points in the game, we have less data to work with



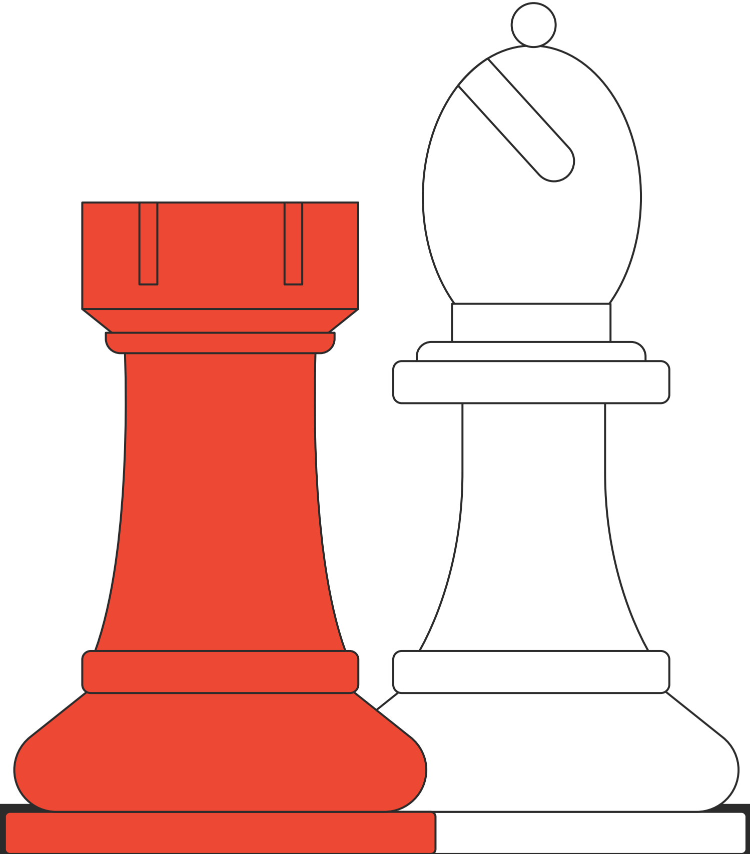
Building Our Dataset

- For **each game** in our data with ***n*** moves, we create ***n*** rows in a new dataset that have partial knowledge of the game**.**
- We add columns for:
  - The current **move number**
  - The **difference in both players' elo**
  - The **stockfish score** of the current board
  - The average **change in stockfish score** for white and black
    - Aggregates the moves both players make over the entire game

Correlation Heatmap



- **White elo, elo difference, and Black elo** correlated
  - We remove black's elo, as it is a calculated column
  - White & Black elo may be correlated because of online matchmaking
- Note that **White average delta** and **Black average delta** also correlated, while less strongly so



Building and Comparing Models

Train-Test-Split

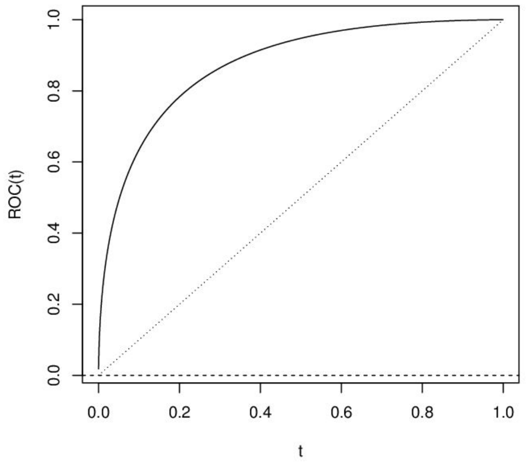
- **80-10-10** train-test-validation split
- Split on **entire games, rather than per move,** to limit data leakage
- Stratify based on **game length** (3 bins) and **game result.**

Model Types

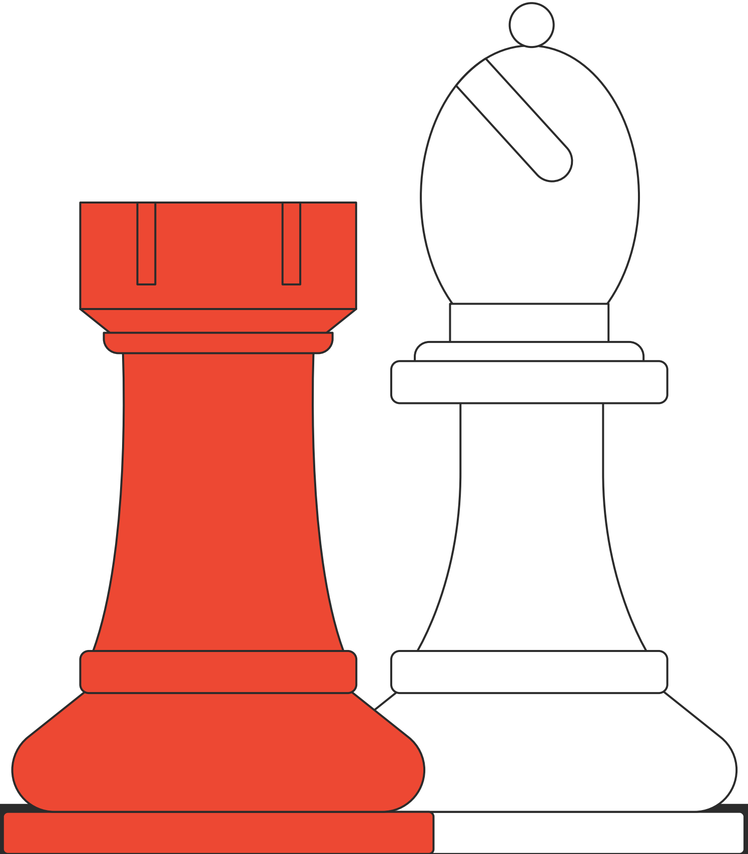
- We will evaluate different forms of models:
  - **Logistic regression**
  - **XGBoost**
  - **K-Nearest Neighbors (K-NN)**
- We also consider **feature interactions**, particularly between the current move number and the features extracted from the chess board.

Model Criteria

- **Accuracy**
    - our model is considered accurate for a specific row if its **observed outcome** matches our **model's most probable outcome**.
- OvR-ROC AUC score (**AUC score**)
    - The Receiving Operator Characteristic (**ROC**) curve is a visual representation of the **tradeoff between a model's precision and recall.**
    - ROC is specifically for **binary classification**, so we aggregate the **area under the curve** (AUC) of three one-versus-the-rest curves.
    - AUC score ranges from 0.5 (worst) to 1 (best).



Example ROC curve



Best Model

Model Comparisons

Metric	Logistic Regression	19-NN	XGBoost
Accuracy	**0.667**	0.626	0.659
ROC-OvR AUC	**0.84**	0.80	0.83

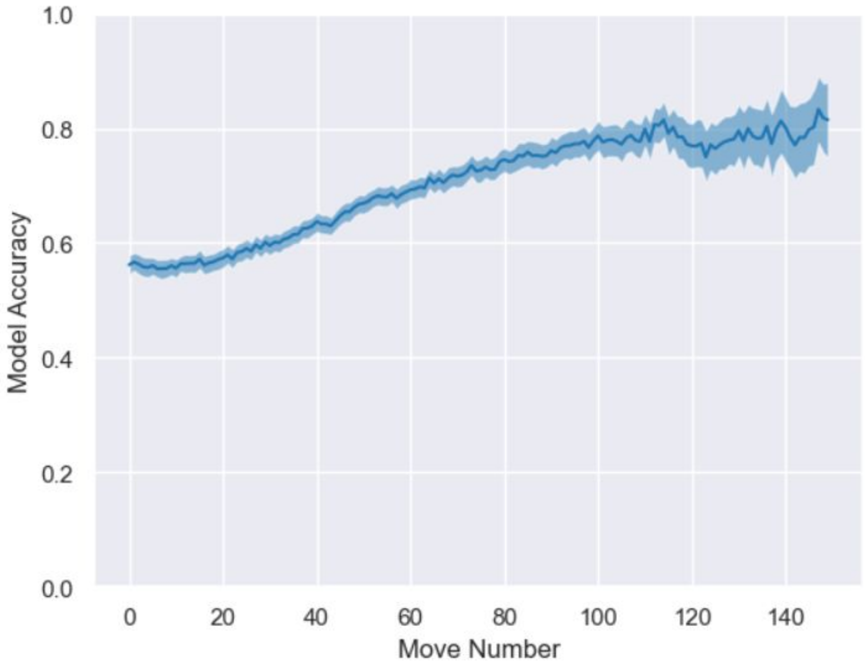
- **Logistic Regression** is our best model from both metrics.
- XGBoost performs better with various **nonlinear relationships** between features; stockfish values are limiting for this model.

Model Comparisons

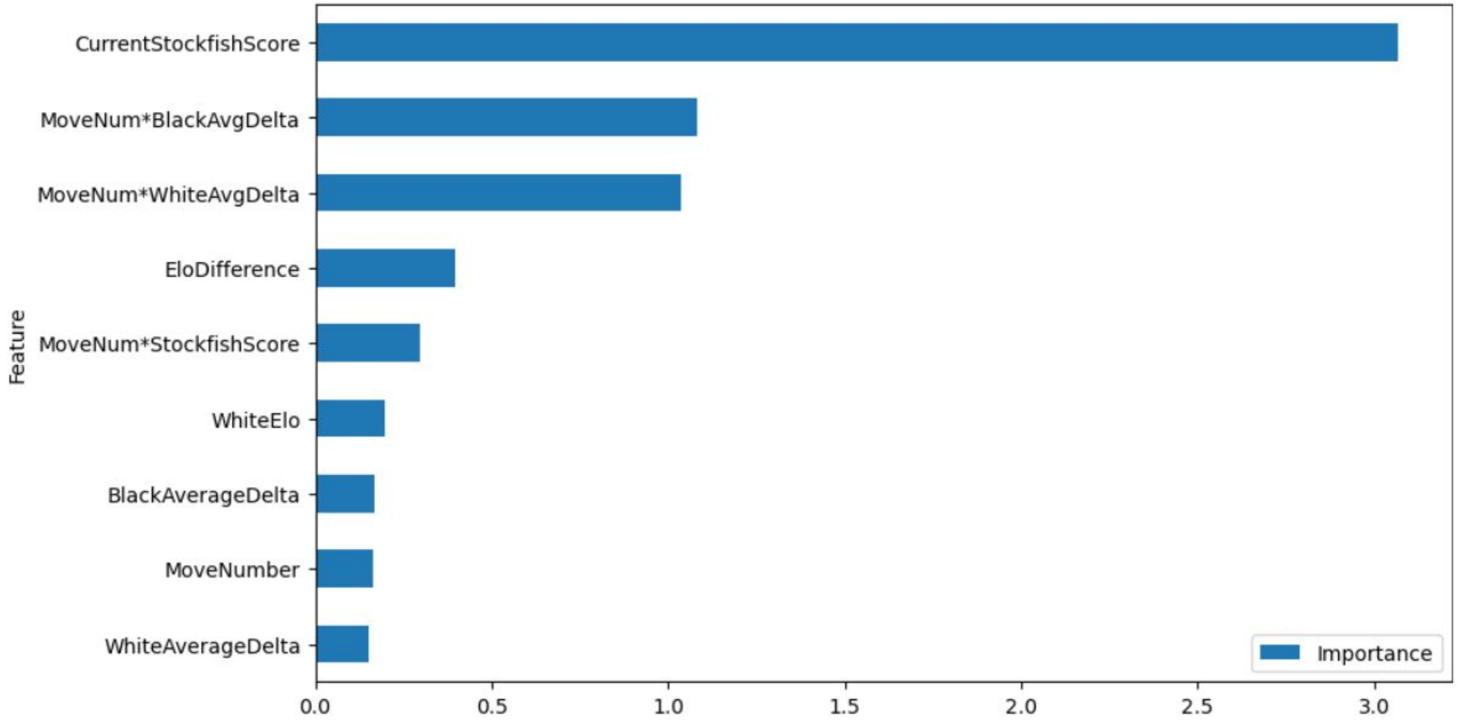
Metric	Logistic Regression
Accuracy	0.667
ROC-OvR AUC	0.84

Model Accuracy Over Time

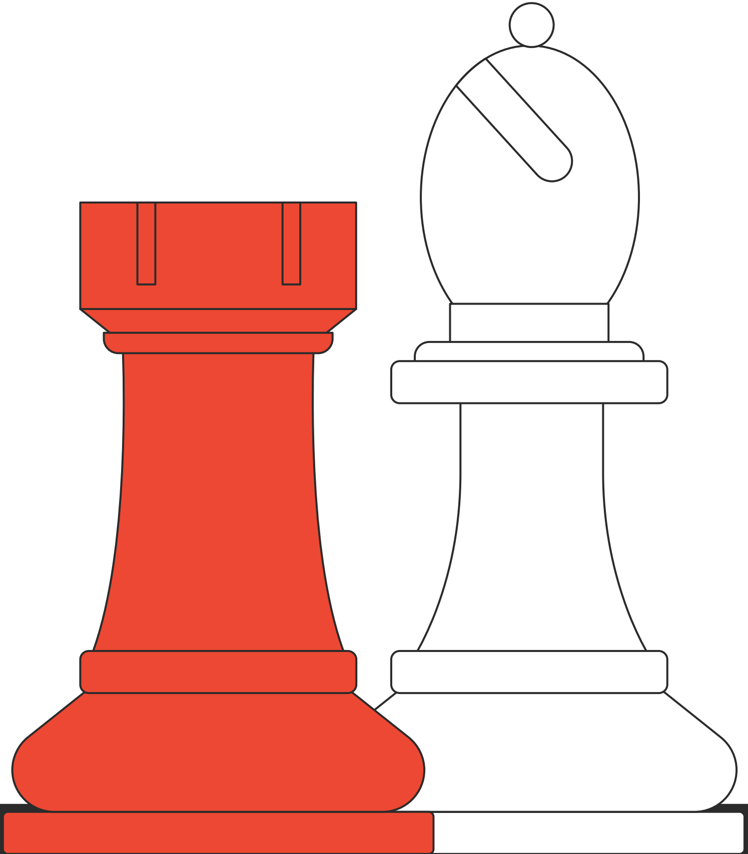
- The figure calculates accuracy for all games that are **at least** that long,
    - e.g. “average accuracy at move 80, for all games that lasted 80 or more moves”
- **Accuracy improves as the game goes longer.**
- We calculate accuracy error by **bootstrapping** the accuracy for each move number; it **increases over time** because of the fewer long games in our data set.



Logistic Regression Features & Importance



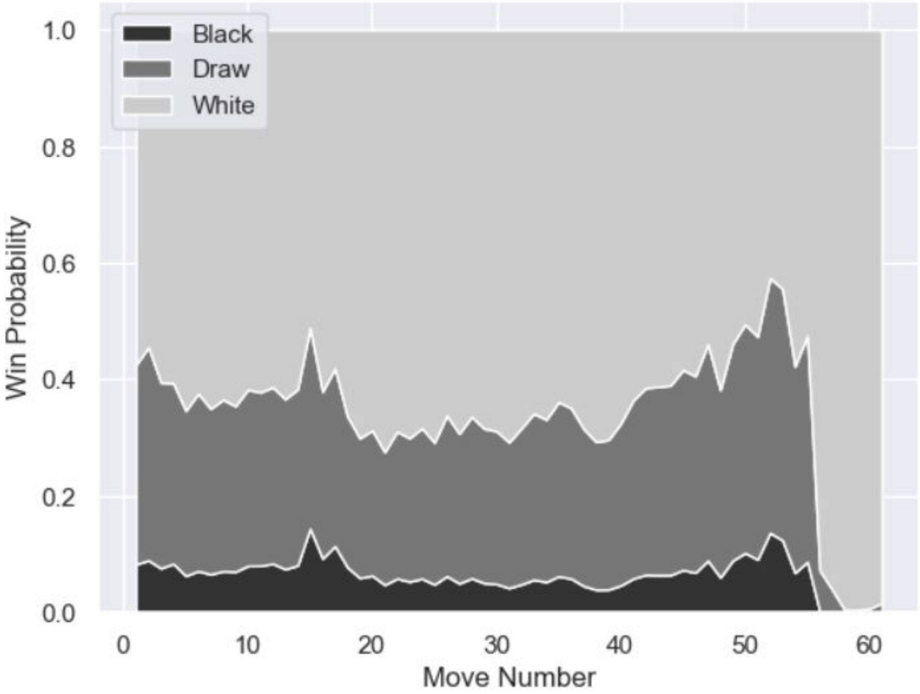
Feature importance in model. Note the significance of each interaction term.



Further Analysis

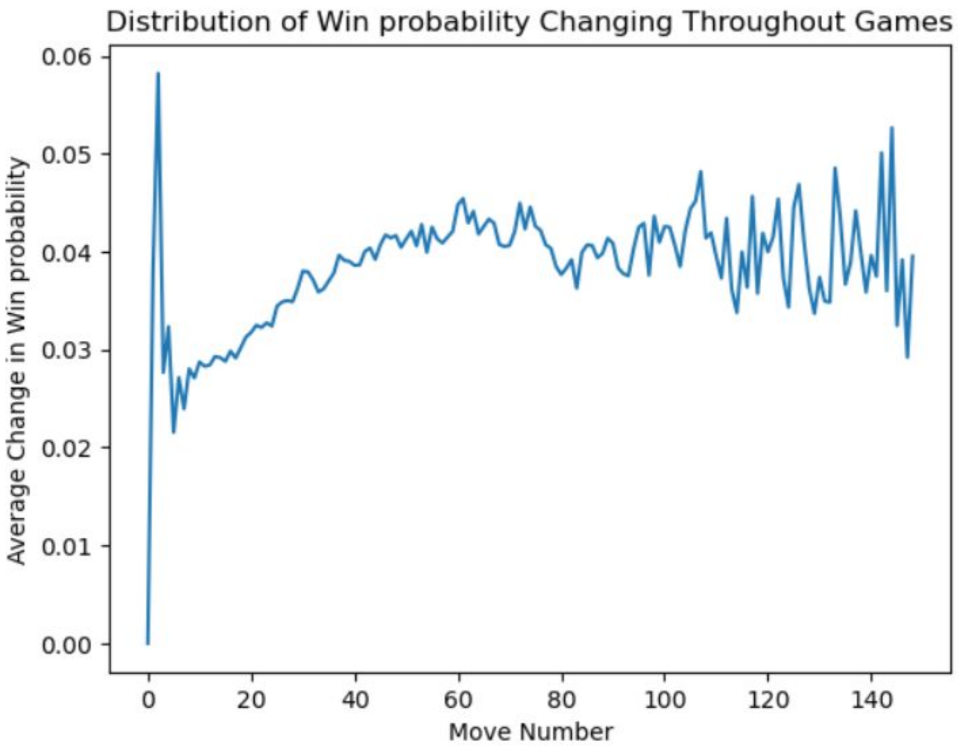
Change in Win Probability for One Game

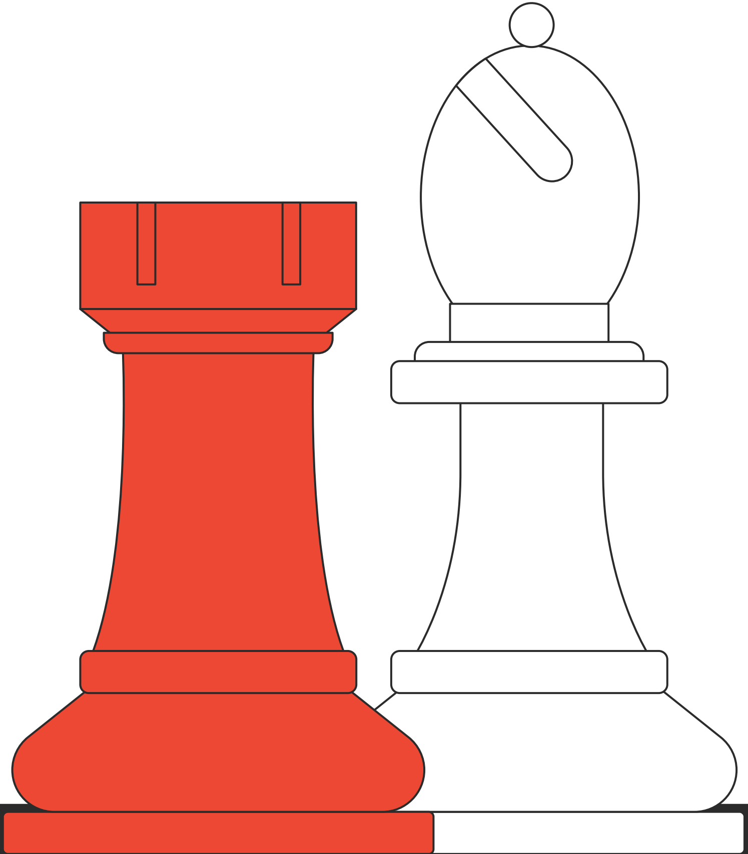
- We can use the model for a **single game** to see win probability change as it progresses.
- The diagram shows a game between a White (Elo 2556) and Black (Elo 2329), so white begins with a significantly higher win probability **because of the elo difference**.
- This win probability spikes at **move 55,** likely a move that guaranteed checkmate.



Aggregate Change in Win Probability

- We could gain insights into which parts of a game are most important based on **when win probability changes the most on average.**
- Win probability seems to change the most in **the first move**, as well as the **midgame and endgame.** Perhaps because chess openings are fairly solved, the moves directly after the opening move aren't as influential.
- **Less data in the endgame** leads to the erratic model.





Future Steps

Future Steps

- Changing and comparing the **chess engine** used
    - Most have a focus on **winning games,** rather than predicting win probability from independent players. There is room for optimization.
    - Extracting more features than a single score of the entire board may be useful.
- Compare and assess the model using data **across different online leaderboards**.
- Extend the work completed in this research to **other esports**, where we may use computer vision or game mods to extract variables from a game directly.

Acknowledgements

I want to thank:

- My colleagues for their support and feedback throughout this project,
- All my undergraduate professors for believing in me, pushing this work forward, and giving insightful feedback, and
- **All of you for listening.**

Thank you!