Outcome Prediction in Speed Chess

O P

October 19, 2023

Abstract

Background and Objective Speed chess, a time-constrained variant of traditional chess, has surged in streaming popularity. Engine evaluation, displayed on these streams, aids viewers in discerning the leading player. However, it does a sub-optimal job because it focuses entirely on the game's positional aspect. This study seeks to enhance this metric by integrating additional parameters like time and player ratings, and present streaming-relevant insights into model performance and workings.

Methods Data from 1,188,000 speed chess games were sourced from lichess.org, balanced for time controls and player ratings. Each entry represents a random game snapshot, targeting the game's outcome. Two models were developed: a logistic regression relying solely on engine evaluations and an XGBoost model enriched with features like remaining time and player ratings. Model performance was assessed holistically, bifurcated based on moves (pre and post the 30th move), and analyzed across some important data subsets. SHAP values and feature performance metrics were reported.

Results The all-features model consistently outperformed the engine-only model, achieving a classification accuracy of 65.34% compared to 60.59% across the test dataset. Engine evaluation, player ratings, and time remaining were the most important features. While both models exhibited increasing accuracy with move progression, the all-features model demonstrated superior performance, especially before move 30 and in lower rating buckets. Across different time controls, the all-features model maintained consistent accuracy, whereas the engine-only model showed a preference for longer time controls.

Conclusions The metric currently used to demonstrate who is winning in speed chess streaming - engine evaluation - can be improved using additional information including time and player ratings. This research offers a proof of concept and highlights pathways for future enhancement.

1 Introduction

Speed chess is a variation of traditional chess in which the game can conclude via either traditional chess outcomes (checkmate or stalemate) or running out of time. In recent years it has rapidly gained popularity [1]. Speed chess streaming has also gained popularity, and as a result, commercial viability [2] [3]. A problem with speed chess streaming is that in suitably complex positions it is difficult for an observer to understand who is winning if the observer has lower skill than the players in the game they are observing, because the ability to evaluate who is winning is linked with the ability to play the best moves i.e. if someone can effectively evaluate who is winning, they would be at least as skilled as the players playing [4]. For example, if a match between two high-level players is being streamed, most humans will be less skilled and unable to comprehend a complicated position to the same level as the players, therefore be unable to tell who is winning, and experience reduced enjoyment.

In speed chess, a common solution is to display a metric that evaluates who is winning, allowing less skilled viewers to follow the game [5]. A program

with above-human playing ability evaluates who is winning (termed "engine evaluation") and this value is displayed alongside streams [6]. However, engine evaluation describes which player is winning if they were to both play with engine-level skill, which is far beyond human ability [7]. Generally, accuracy of the engine evaluation in predicting game outcomes decreases as the accuracy of play diminishes. Many factors influence a player's ability to play accurately, including time permitted to think and skill level, both of which vary in streams [8]. Engine evaluation also does not consider time-based outcomes - e.g. a player could have a winning position but be about to lose on time, and the engine would evaluate their position as winning. Therefore, engine evaluation is an imperfect metric for capturing who is winning a speed chess game. As such, there is utility for a metric that can more accurately predict game outcome than engine evaluation.

This field is termed "in-game outcome prediction", where the outcome of a game is predicted during the game using pre- and in-game information [4]. It has been studied in domains such as traditional sports and esports, typically using machine learning (ML) techniques [9][4]. Two studies have done this in traditional chess, but not speed chess (no time component) [10][11]. As a result, the first aim of this paper was to use ML techniques with pre- and in-game features (including engine evaluation) to predict the outcome of speed chess games, and compare predictive performance with the current standard of lone engine evaluation.

As this is the first speed chess outcome prediction study and primarily a proof-of-concept, we would also like to facilitate future speed chess outcome prediction research. A benefit of using ML techniques to predict outcome is that explainability techniques such as SHAP values can be used to gain insight into which features are most important to decision making, informing future feature engineering and model improvement [12]. Additionally, we would like to provide insights that can assist in model deployment decision making for streaming services. This can be done by examining how performance varies across pre-game subsets of the data [13]. For example, we may identify that outcome prediction performance is disproportionately poor during low time games between two players of low skill, so a streaming service can choose not to display predictions during these games.

As such, the second aim of this study was to provide provide insight with SHAP values and subset performance analysis techniques.

2 Related Work

Pre-game chess outcome prediction has been studied [14][15][16][17]. Primarily, optimisation of a singular "rating" value calculated from game history that quantifies playing strength within a pool of players, therefore roughly being able to predict outcome of a game before it begins [15][16][17]. For example, a popular Kaggle competition aimed to develop an optimal "rating" system by maximising pre-game outcome prediction based on a single value [18]. Insight to high-scoring entries have been documented [15].

As mentioned, two studies have examined non-speed in-game chess outcome prediction.

[10] used naive Bayes with in-game features including engine evaluation, but not time. They also only focus on high-rated games, which is sub-optimal for streaming outcome prediction because medium and low-rated game streaming is popular.

[11] used proprietary ensembling techniques without time, and with direct positional features rather than engine analysis. As explained in section 2.1, this is likely a more effective strategy, but is beyond the scope of this study. They also exclusively focused on long, high-rated games.

As a result, the primary novelties in this paper are:

- Chess outcome prediction using time (i.e. speed chess)
- Inclusion and control of all ratings and short time controls
- Model explainability

2.1 Feature Choice

Feature selection is important for optimal outcome prediction [9].

2.1.1 Pre-game

As mentioned, the chess pre-game outcome prediction studies primarily involved differing ways to calculate an ELO-type rating - a singular value calculated from game history used to roughly quantify playing strength within a pool of players. We have used the most recent iteration and current state-of-the-art rating measure, Glicko 2 [19].

The only time-based pre-game feature we included is time control. Time control refers to the time format of the chess game. It is in the format (time + increment) where refers to the each player's time at the beginning of the game (usually minutes), while increment refers to the amount of time added to a player's remaining time after they play a move (usually seconds)

Generally quality of a chess move has a causative relationship with time permitted to make the move [8]. Therefore, games where less time is permitted to make moves generally have lower average move quality, meaning shorter time controls (e.g. 1+0) generally have lower average move quality than longer time controls (e.g. 10+5).

Tournament was included as a feature. Tournament play lends itself to larger disparity in rating due to smaller pool, but this will be captured by the rating features. There may be some additional differences.

2.1.2 In-game

We will break in-game features into three categories - positional, time and other.

Positional Positionally, the information likely to impact winner can mainly be broken down into two components - who is likely to win if both players play the best possible moves (as calculated by top engines), and how difficult the position is to play.

Engine evaluation is a feature. The longer an engine is permitted to run, the better the evaluation will be [6]. In order to facilitate live predictions and varying move times, engine run time is varied and encoded as a feature, as outlined in section 3.2.

While engine analysis is a great proxy for who is winning positionally, it doesn't consider how difficult the position is to play for a human [20]. There is variation in how simple a position is to play, irrespective of the winning potential of a position. This simplicity can be imbalanced, leading to an imbalance in time required to think and therefore, in time-based games, provide an advantage for the player with the simpler position. For example, there may be only one legal move or obvious reasonable strategy for one player, but much decision making required for the other.

There is no established metric to quantify the complexity of a position, but attempts have been made [20][21]. A tool was created that uses a neural network to predict the "sharpness" of a position - specifically, the "expected centi-pawn [(CP)] loss and the expected blunder (CP loss > 200) probability" of a position "given a player's Elo rating" [21]. This metric was included as a feature.

[11] attempts to encode positional evaluation (winning and complexity) inherently. They encode all board information as features and allow the model to learn what is important for outcome prediction, potentially including complexity or a similar concept. This may be the best long-term approach to outcome prediction - using direct positional information to predict outcome, rather than just using the single engine evaluation value to predict outcome. However, this is a quick proof of concept study, so we utilised the highly researched and optimised engine evaluation value and pre-established complexity metric rather than reinventing the wheel. That said, we did arbitrarily include some direct positional information - piece counts for either side - to naively examine its impact when complexity and engine evaluation are already provided.

[10] uses engine evaluation directly, and a metric termed "winning score" for complexity. While winning score is reasonable, we have opted to use the aforementioned complexity metric as it is more demonstrably proven.

Time As mentioned, generally strength of a chess move has a causative relationship with time permitted to make the move [8][22].

No research has been done to identify time-based in-game features effective for outcome prediction, so we chose basic features - time remaining for each player and time remaining at the conclusion of the active player's previous turn. Time at the previous turn is included to capture how long the current turn has taken.

Snapshot Both in-game chess outcome prediction papers consider features as sequences from the beginning of the game - e.g. if outcome is being predicted at move 25, features (e.g. engine evaluation) will be calculated for move 25, move 24, move 23 and so on [11][10]. This is likely effective. However, this is a proof of concept study rather than exhaustive optimisation so we prioritised simplicity. Use of sequence data may well improve performance and should be tested in future studies if the goal is performance optimisation.

As a result, we will use only in-game features from the current turn, with the exception of time at the conclusion of the current player's previous turn, which facilitates calculation of the current player's ongoing turn length.

2.2 Model selection

Pre-game As mentioned, pre-game chess outcome prediction has primarily focused on creation of a single rating value. As will outlined in section 3, we include a state-of-the-art version of this metric as a feature in our study, so we will ignore the techniques used in calculation.

Other domains have employed various statistical techniques. A 2022 review of pre-game outcome prediction in traditional sports found no unanimously superior method, although more complex models including ANNs and boosting algorithms were more effective with larger datasets.

In-game The two papers that examined in-game non-speed chess outcome prediction - [10] and [11] - used naive Bayes and custom ensemble techniques respectively. However, while they provide useful information for feature selection, they provide no compelling reason to use these specific algorithms. Given all performance comparisons in this study will be intra-study so there will be no need for direct comparisons with previous studies, we have opted to instead use models more prevalent in recent in-game outcome prediction literature. Model choice in modern in-game outcome prediction esports studies varies depending on factors such as dataset composition and study objective (interpretability vs performance), but primarily involves logistic regression (LR), random forests (RF) or boosting algorithms such as XGBoost (XGB) [4].

For example, as highlighted in [4], the consensus of most effective models in game "Dota 2" are LR and RF. However, it is noted that dataset size was likely a limiting factor. As mentioned with pre-game outcome prediction, more complex models like XGB can be superior with larger datasets [9]. There is an abundance of appropriate speed chess data for this study, so we opted to include XGB along with LR and RF.

3 Methods

3.1 Dataset

Data were acquired from lichess.org (July 2023—30.8GB—95.3M games) [23]. Outcome distribution was 292673 white wins, 275322 black wins and 26005 draw.

The dataset does not reflect the rating and time control distributions observed in streaming. We sampled from the data to more appropriately represent the distribution found in speed chess streaming and reduce bias.

This paper is focused on speed chess, which generally constitutes three categories (from shortest to longest) - bullet, blitz and rapid [22]. Bullet generally refers to games where players start with two or fewer minutes, blitz to games that start with three to five minutes, and rapid to games with 10 to 20 minutes.

The Lichess database contained many more bullet games than blitz, and many more blitz than rapid. Due to game length, if the same quantity of time is spent playing each format, the number of games avail-

able for a given format will inversely correlate with the format's length. To avoid bias, we identified the two most popular bullet, blitz and rapid time controls - 1+0 and 2+1, 3+0 and 3+2, 10+0 and 10+5 respectively - and ensured the dataset contained an equal quantity of each.

For ratings, games from players of all ratings are streamed - high (pro) and low (beginner celebs)/ even distribution. In the Lichess dataset, ratings follow a bell curve distribution, meaning far more games are played at ratings close to the mean, and fewer at the extremes. To avoid bias, we divided ratings into nine buckets - less than 800, every 200 rating points from 800 to 2200 (800-999, 1000-1199...), and above 2200, and ensured the dataset contained an equal quantity of each.

Each permutation of time control [6] and rating bucket [9] contained 22000 samples, combining to a total 1,188,000 samples.

The dataset was split 80/10/10 train/validation/test.

3.2 Features

Directly extracted:

- Result
  - Outcome of the game - 0 is white win, 0.5 is draw and 1 is black win.
- White rating and Black rating
  - Glicko 2 ratings for each player.
- Time control
  - Time control - 1+0, 2+1, 3+0, 3+2, 10+0 and 10+5 are 0 to 5 respectively.
- Tournament
  - Whether the game was part of a tournament - 0 is not tournament and 1 is tournament.

The PGN 1 is included for each game. From the PGN, the following variables were acquired:

- Turn
  - Current player's turn - 0 is white's turn and 1 is black's turn.
- Move number
- Time remaining
  - Time remaining in seconds for current player at moment of snapshot (as described in "Time" below), and at the end of both players' previous turns.
- Engine evaluation
  - Engine evaluation in centipawns where positive represents advantage for white.
- Engine time
  - Length of time in seconds for which engine was run (detailed in "Engine Evaluation" below).
- Position complexity
  - Output of complexity NN (three floats)(as outlined in 2.1.2).
- Piece counts
  - Count of each piece for both players.

Feature engineering was performed for the below features, primarily to maximise the chance that any snapshot that could possibly occur in a speed chess game is appropriately represented, such that an outcome prediction can be made from any moment during a live streamed game. Figure 1 outlines the data preparation process.

Move number To permit prediction from any stage of the game, data need to contain samples from any stage of the game, aside from the final half-move ("half-move" is an individual player's move - one "move" in PGN is the combination of an individual move from each player). To achieve this, for every game a half-move which is neither the first or final half-move was randomly selected (Figure 1). Randomly selecting a move rather than splitting every half-move into a new sample (such that a game X

half-moves long becomes X samples) increases game diversity in the dataset for the same number of samples, which is limited by computation time.

Engine Evaluation Stockfish 16 was used [24]. Engines, including Stockfish, requires computational time where more time results in a more accurate evaluation. In order to permit live outcome prediction, predictions need to be made quickly enough to be relevant, while allowing sufficient engine depth for accurate analysis. On Lichess, pre-moves - a move made prior to receiving the opponent's move - do not reduce the player's time. However, due to internet latency there is a delay while the pre-move is sent to the opposition player and spectators resulting in a difference between real time and game time. As a result, there will always be X time between moves for a third-party spectator, permitting at least Y time for outcome predictions to be made. Therefore, to permit predictions during pre-moves, a minimum engine time of 0.08s was arbitrarily chosen (this and following steps are outlined in Figure 1. This will need to be adjusted based on hardware etc.. To allow for predictions using more accurate engine analysis as a player's turn progresses without a move being played, engine times of 0.32, 1.28, 5.12, 20.48 and 81.92 seconds were arbitrarily chosen. To permit prediction at each of the engine times before a move is made, the model would need to be trained on every engine time less than the turn length. To maximise variation for a fixed training dataset size, and given that engine time is the primary limiting factor computationally, a single engine time less than the turn length was randomly chosen for each position and saved as a single training example, such that every training example was a different position (and game). Distribution of engine times was (engine time | count) 0.08 | 256120, 0.32 | 253793, 1.28 | 63003, 5.12 | 16514, 20.48 | 1811, 81.92 | 42.

Time Time remaining for the current player's turn needed to be adjusted for the chosen engine time. For example, if a turn began with 80 seconds remaining, ended with 73 seconds remaining, and the engine time was randomly chosen to be 5.12 seconds, the sample

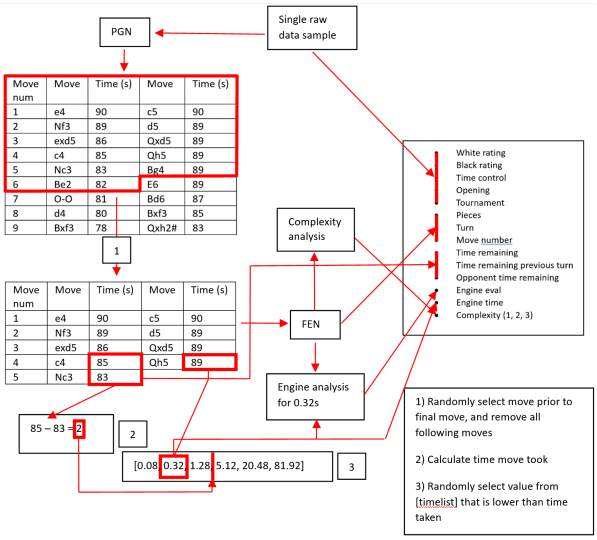

Figure 1: Overview of the data preparation process.

represents a snapshot of the game where the player has 80 minus 5.12 seconds remaining. As such, current time remaining was calculated by adding time remaining at beginning of their turn to engine time.

3.3 Models

As mentioned, we opted to use LR, RF and XGB models for outcome prediction. LR was implemented using scikit-learn, lbfgs for optimisation and max 1000 iterations [25]. RF and XGB were implemented using the XGBoost package, using softprob, and grid-search was used to select reasonable hyperparameters with training and validation datasets [26]. Results utilise holdout test dataset.

Given that the engine-only classifier was relying on a single feature, we used LR with the same configuration as above.

Evaluation metric While some papers have employed custom performance metrics for probability (which is what would be displayed when implemented), most, including both in-game chess studies, primarily utilise classification accuracy [27][4]. We

have also done so.

3.4 Experiments

Chess and Esport literature indicates that model behaviour varies greatly with game progression [4]. As demonstrated in Figure 2, early game outcome predictions are generally more difficult than late game predictions, often resulting in lower accuracy and differing decision-making process. To examine this, we have split performanced-based results arbitrarily pre- and post- move 30. Future studies should consider doing the same for SHAP values.

Model comparison We compare accuracy of the all-feature LR, RF and XGB, and engine-only LR models to provide a preliminary indication of which model(s) are best for this relatively unknown domain. We also compare how accuracies of the overall best-performing all-feature model (XGB) and the engine-only model vary over move number, to investigate the difference between early and late-game performance.

Feature importance SHAP values provide insight into the relative importance of features used by a model, outlining which features are more and less important in the model's decision-making process [12]. A feature with high importance will influence the outcome more than a feature with low importance. For the highest performing model we compute SHAP values for each feature for each outcome to determine their relative importances.

Performance over subsets We examine the highest performing model's performance across streaming-relevant actionable subsets of the data - time control and rating. Results are split into pre- and post- (inclusive) move 30.

4 Results

4.1 Model comparison

Performance of each model on the entire dataset, positions prior to move 26, and positions from move 26 onward, are outlined in table 1 . In every case, XGB outperformed RF which outperformed LR and EO LR. LR outperformed EO LR overall and prior to move 26, but performed worse from move 26 onward. Across all models, mean performance prior to move 26 was lower than mean performance from move 26 onward.

	Accuracy (%)		
Model	All data	<= move 30	> move 30
EO	60.59	58.33	68.64
LR	61.90	60.08	68.40
RF	63.32	60.58	73.12
XGB	65.34	62.56	75.28

Table 1: Classification accuracy (%) of engine-only model (EO) and all-feature LR, RF and XGB models over the whole dataset ("All data" column), on samples where move number is 30 or lower ("<= move 30" column) and on samples where move number is above 30 ("> move 30" column).

As shown in Figure 2, the majority of positions occur prior to m̃ove 30. In this range, the all-features model outperforms the engine-only model, while accuracy increases for both as move number increases. Accuracy somewhat plateaus and becomes erratic at around move 30.

4.2 Feature importance

As shown in Figure 3, for white win and loss outcomes the plots are very similar. Engine evaluation, white rating, black rating, opponent time remaining and turn are the five most important features in each case. For the draw outcome, move number was most important, followed by opponent time remaining, time remaining, black rating and engine evaluation. Engine time and tournament were the bottom two features in win and loss outcomes, and second and third from the bottom for the draw outcome. A queen was the most important piece in all three outcomes.

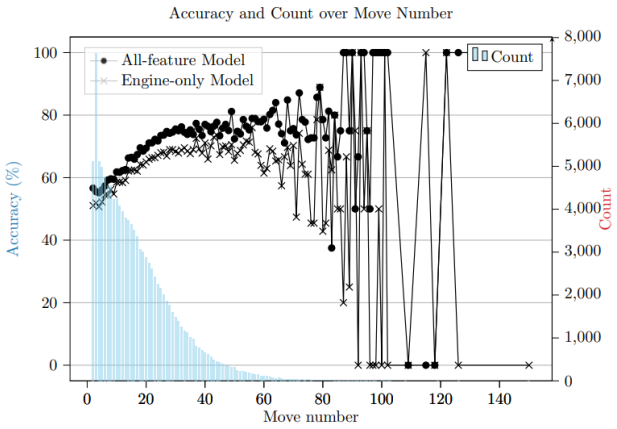

Figure 2: Line graphs showing classification accuracy (%) (left Y-axis) of the all-feature XGB (circles) and engine-only (crosses) models over move number (X-axis), and count of each move number (right Y-axis)

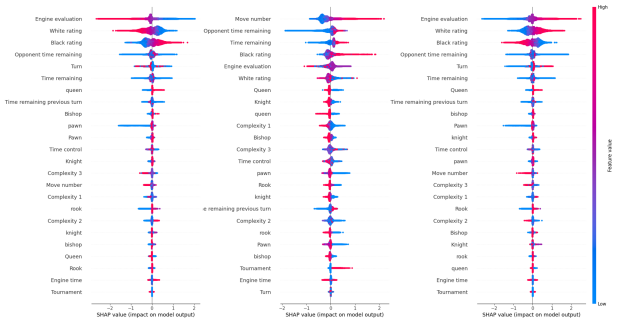

Figure 3: SHAP values for white win (left), draw (middle) and white loss (right). Values are sorted from most (top) to least (bottom) important. White piece names are capitalised.

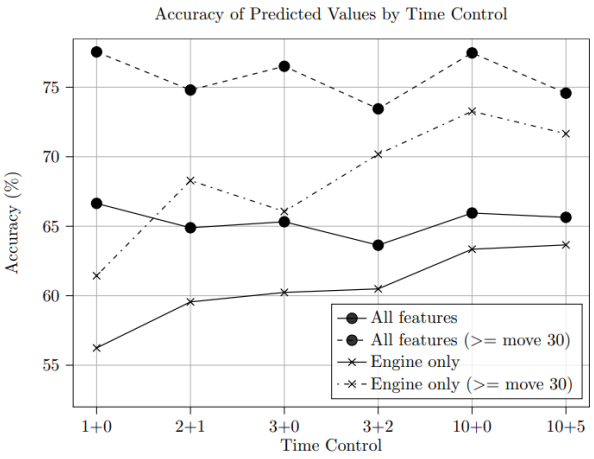

Figure 4: Line graphs showing classification accuracy (%) (Y-axis) of the all-feature XGB (circles) and engine-only (crosses) models at different time controls (X-axis) on all data (solid lines) and > move 30 (dotted lines)

4.3 Performance over subsets

As shown in 4, over the whole dataset, all-feature model accuracy was higher than engine-only accuracy for every time control. All-feature accuracy was similar across all time controls, ranging from 65.5% at 1+0 to 64% at 10+5, while engine-only accuracy trended generally with increasing time control, from a minimum of 56% in 1+0 to maximum of 63% in 10+5. For data where move number was >= 30, the all-feature model was more accurate than the engine-only model for every time control. Both models trended generally upward, all-feature and engine-only models ranging from minimums of 71% and 61% for 1+0 to maximums of 75% and 72% for 10+5 respectively.

As shown in Figure 5, on the whole dataset all-feature model accuracy was higher than engine-only accuracy for every white rating bucket. All-feature accuracy was similar throughout, with slight elevated accuracy at each end and lower accuracy for the middle buckets, while engine-only accuracy was similar throughout with a slight upward trend. After move 30, all-feature model accuracy was higher for every

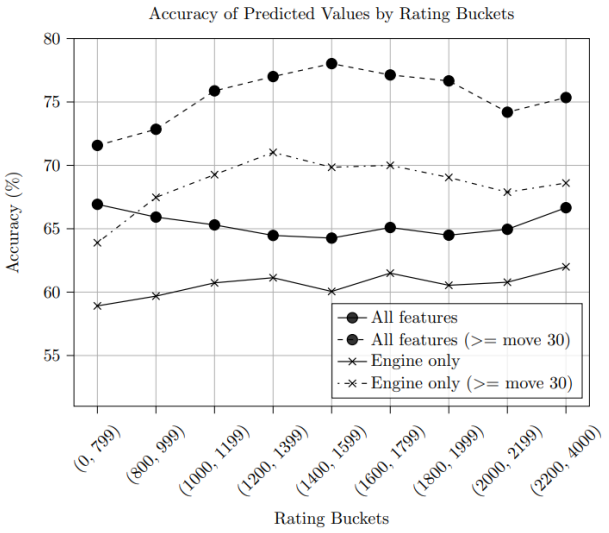

Figure 5: Line graphs showing classification accuracy (%) (Y-axis) of the all-feature XGB (circles) and engine-only (crosses) models in varying white rating buckets on all data (solid lines) and > move 30 (dotted lines)

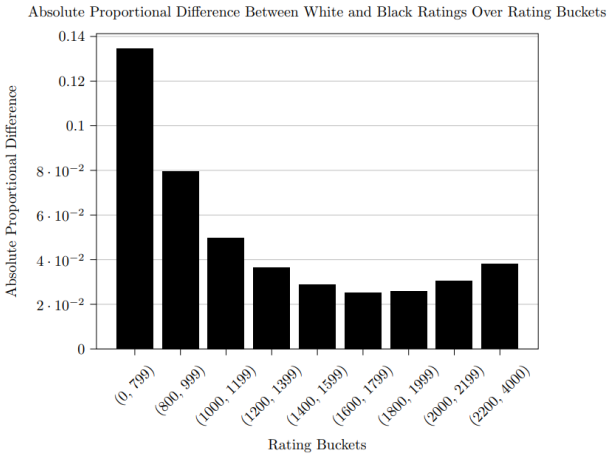

Figure 6: Mean absolute proportional difference between white and black ratings (Y-axis) for each rating bucket (X-axis)

elo bucket than engine-only accuracy. Accuracy for the engine-only model was lower at the ends than the middle, with a slight dip in the (1600 - 1799) bucket. The same was true with the all-feature model, but with a relatively lower low-end and relatively higher high-end.

As shown in Figure 6, the absolute relative difference between white and black ratings at different ratings buckets has a convex parabolic shape skewed toward lower ratings - largest at 0 - 799 rating ($\tilde{1}3\%$), greatly decreasing until 1600-1799 rating ($\tilde{2}.5\%$), then gently increasing until 2200-4000 ($\tilde{4}\%$).

5 Discussion

The aim of this study was to use engine evaluation and additional features including time for live speed chess outcome prediction and compare it with the current standard of engine-only outcome prediction, and to provide some insight into model performance with SHAP values and subset performance analysis. Our key findings were A) the model using additional features unanimously outperformed the engine-only model, and XGB performed best of the tested models, B) engine evaluation, player ratings and time were most important, and C) XGB all-feature model notably outperforms engine-only prediction in low time controls and games with large rating difference. As a result, we conclude that the use of additional features including time could be used to improve live speed chess outcome prediction.

As shown in Table 1, the XGB model outperformed all others in every regard. This is in line with expectations from the literature, where more complex models such as XGB tend to outperform simpler models where larger datasets are used [4][9].

As games progress, the outcome becomes nearer and predictive performance generally increases [10][11]. At the beginning of a game, no in-game advantage has been established. As the game progresses, imbalances generally increase, leading to better predictive power. This is shown in Figure 2 - there is a consistent increase in performance from move 0 to roughly move 30, after which performance plateaus and becomes increasingly erratic. The erratic behaviour can be explained by the decrease in sample count displayed in the same figure. To repli-

cate the move number distribution that will be found in the streaming domain, move count samples were not artificially manipulated to decrease bias toward earlier moves. As such, training samples exponentially decrease with move number, increasing performance erraticism.

Understanding how the model works can provide insights to direct future work. We did this using SHAP values, which display relative importance of features [12].

While no research has compared the relative importance of speed chess features in predicting game outcome, studies have demonstrated the predictive power of rating [15][16][17]. We anticipated that engine evaluation and time-based features would be relatively important as they directly impact game outcome.

As shown in Figure 3, for win and loss outcomes, engine evaluation, rating and time were found to be most important. Time control was relatively unimportant, suggesting that in-game time-based features can be considered independently of the game's time control, and that increment was not particularly important.

Some individual piece features (specifically the queen) were surprisingly important, indicating that engine evaluation does not entirely capture the positional aspect of speed chess necessary for outcome prediction, and the use of non-engine evaluation positional features can be beneficial [28]. The complexity features were surprisingly unimportant, suggesting that either they do not effectively capture position complexity, or position complexity isn't as important as anticipated in outcome prediction.

Engine time was notably unimportant, suggesting that the minimal engine depth used (0.08 seconds) adequately evaluates the position for outcome prediction, and additional calculation depth provides minimal benefit. Tournament was also very unimportant, suggesting that game characteristics don't differ much between tournament and non-tournament play.

Move number was the most important feature for draw outcome, but mid-way important for win and loss outcomes, suggesting that draws are mostly predicted at a subset of move number, but wins and losses are frequently predicted at any move number. Given that most games in the dataset had a win or loss outcome, the model likely conservatively predicts draw outcome, only doing so when it is disproportionately more likely than win/loss. Given that outcome is more obvious with higher move number (Figure 2), draws are likely mostly predicted when move number is high. This can be easily tested in future studies.

The background objective of this paper was to examine outcome prediction to facilitate streaming. Understanding how pre-game features impact model performance affords actionable insight for streaming services, affording better understanding of when to utilise outcome prediction. The two primary features we anticipated being useful are time control and rating, due to being both A) controllable prior to game onset and B) likely to influence model performance.

Time control Logic and literature suggest that engine-only model performance would correlate with time control length because longer time controls permit longer average time per move which increases move quality, reducing the gap between human and engine move quality [22][8]. The all-feature model incorporates time in to its predictions, suggesting it should perform disproportionately well at lower time controls where time is a more important factor.

As shown in Figure 4, the results back up these ideas. The engine-only model's performance correlates with time control length - bullet performance is lower than blitz, which is lower than rapid. In contrast, the all-feature model performs similarly across all time controls.

These findings indicate that the outcome prediction model outlined in this study could be deployed for any time control, and will notably outperform engine evaluation in shorter time controls due to usage of time-based features.

Rating When controlling for time, engine-only model performance may correlate somewhat with rating, because more skilled players will play at a level closer to engine performance, making outcome more likely to correlate with engine evaluation. Presumably all-feature performance should show a similar correlation, as move quality would be the primary

factor varying between rating buckets as other features would be mostly controlled for.

However, this is not perfectly reflected in our results. Prior to move 30, the engine-only model does exhibit a slight, vaguely linear increase in performance with rating, but the all-feature model has a slight convex parabolic shape - higher on the ends, lower in the middle. This elevated low-rating performance does not align with our expectations, as we anticipated only move quality to change between rating buckets, and all other variables remain fairly constant. As it turns out, as shown in Figure **??**, rating difference varied between rating buckets. The mean difference between player ratings was much higher at low ratings, likely due to reduced player count and the resulting reduced ability to find evenly-balanced matches. This larger mean rating difference provided more imbalanced games, likely improving the all-feature model's predictive ability at low ratings, resulting in the observed convex parabolic performance curve.

The reasoning behind performance variation across ratings after move 30 is less clear. Given that engine-only and all-feature models have similar trend suggests that the variation in performance stems from engine evaluation, with the all-feature model's other features providing a general increase in performance, but minimally influencing the trend across rating buckets. The erroneous trend may be a result of low representation of moves after 30 in the training set (see Figure 2) - values close the middle are more similar to better-represented data points and therefore perform better, while more extreme values are worse represented and perform worse. This combined with the expected correlation between rating bucket and performance could explain the observed trend - low-rated games being both extreme and low rating observe lowest performance, mid-rated games observe highest performance and high-rated games being extreme but high rated perform better than low-rated games but worse that mid-rated games. However, this is speculation and further investigation should be conducted.

These findings indicate that the outcome prediction model outlined in this study could be deployed at any rating level, and will notably outperform engine evaluation alone in games where there is a large difference in rating.

6 Conclusion

Speed chess streaming is gaining popularity [2]. Outcome prediction can augment viewer experience by improving understandability [4]. Chess streaming benefits disproportionately because understanding who is winning is inherently difficult. For this reason, engine evaluation is currently displayed on most streams [5]. However, engine evaluation considers only one aspect of the game, and can be improved upon using other easily available data such as time and positional complexity.

However, no studies have examined speed chess outcome prediction. This study provides a baseline for outcome prediction performance and demonstrates that even rudimentary modelling and feature engineering can greatly outperform engine analysis alone.

Specifically, we found that for outcome prediction, use of additional features unanimously improves performance compared with sole engine usage. Also, engine evaluation, player ratings and time were the most important contributing factors to decision making. Finally, all-feature usage notably outperforms engine-only prediction in low time controls and games with large rating difference. As a result, we conclude that the use of additional features including time could be used to improve live speed chess outcome prediction.

This study is a proof of concept rather than an exhaustive optimisation, so many improvements are possible in future studies. For example, performance optimisation can likely be done easily with basic methodology e.g. engineering additional features, and with use of historical features as was done in the other two chess in-game outcome prediction studies [11][10].

Disclosure statement

No conflicts of interest.

Funding

No funding.

Notes on contributor(s)

OP was the sole contributor.

Notes

This work is clearly unfinished. It is current available as a preprint to disseminate preliminary findings but will be updated prior to publication.

References

[1] *Chess Is Booming! And Our Servers Are Struggling.* https://www.chess.com/blog/CHESScom/chess-is-booming-and-our-servers-are-struggling. Accessed: 2023-10-10.

[2] Alexei Scerbakov, Johanna Pirker, and Frank Kappe. "When a Pandemic Enters the Game: The Initial and Prolonged Impact of the COVID-19 Pandemic on Live-Stream Broadcasters on Twitch." In: *HICSS*. 2022, pp. 1–10.

[3] Ming Tang and Jianwei Huang. "How Do You Earn Money on Live Streaming Platforms?—A Study of Donation-Based Markets". In: *IEEE/ACM Transactions on Networking* 29.4 (2021), pp. 1813–1826.

[4] Victoria J Hodge et al. "Win prediction in multiplayer esports: Live professional match prediction". In: *IEEE Transactions on Games* 13.4 (2019), pp. 368–379.

[5] *xQc Gets Checkmated by MoistCr1tikal in 6 Moves!* https://www.youtube.com/watch?v=e91M0XLX7Jw. Accessed: 2023-10-10.

[6] Shiva Maharaj, Nick Polson, and Alex Turk. "Chess AI: Competing Paradigms for Machine Intelligence". In: *Entropy* 24.4 (2022). ISSN: 1099-4300. DOI: 10.3390/e24040550. URL: https://www.mdpi.com/1099-4300/24/4/550.

[7] Kenneth Regan and Guy Haworth. "Intrinsic Chess Ratings". In: *Proceedings of the AAAI Conference on Artificial Intelligence* 25.1 (Aug. 2011), pp. 834–839. DOI: 10.1609/aaai.v25i1.7951. URL: https://ojs.aaai.org/index.php/AAAI/article/view/7951.

[8] Mariano Sigman et al. "Response time distributions in rapid chess: a large-scale decision making experiment". In: *Frontiers in Decision Neuroscience* 4 (2010), p. 60.

[9] Rory Bunker and Teo Susnjak. "The application of machine learning techniques for predicting match results in team sport: A review". In: *Journal of Artificial Intelligence Research* 73 (2022), pp. 1285–1322.

[10] Paras Lehana et al. "Statistical Analysis on Result Prediction in Chess". In: (2018).

[11] Mohammad M Masud et al. "Online prediction of chess match result". In: *Advances in Knowledge Discovery and Data Mining: 19th Pacific-Asia Conference, PAKDD 2015, Ho Chi Minh City, Vietnam, May 19-22, 2015, Proceedings, Part I 19*. Springer. 2015, pp. 525–537.

[12] Scott Lundberg and Su-In Lee. *A Unified Approach to Interpreting Model Predictions*. 2017. arXiv: 1705.07874 [cs.AI].

[13] Yeounoh Chung et al. "Slice finder: Automated data slicing for model validation". In: *2019 IEEE 35th International Conference on Data Engineering (ICDE)*. IEEE. 2019, pp. 1550–1553.

[14] Zheyuan Fan, Yuming Kuang, and Xiaolin Lin. "Chess game result prediction system". In: *Machine Learning Project Report CS* 229 (2013).

[15] Yannis Sismanis. "How I won the "Chess Ratings - Elo vs the Rest of the World" Competition". In: *CoRR* abs/1012.4571 (2010). arXiv: 1012.4571. URL: http://arxiv.org/abs/1012.4571.

[16] Diogo R Ferreira. *Predicting the outcome of chess games based on historical data*. Tech. rep. Technical report, IST–Technical University of Lisbon (November 2010), 2010.

[17] Arthur Berg. "Statistical analysis of the elo rating system in chess". In: *Chance* 33.3 (2020), pp. 31–38.

[18] *Chess ratings - Elo versus the Rest of the World*. https://www.kaggle.com/c/chess. Accessed: 2023-10-10.

[19] Mark E Glickman. "Example of the Glicko-2 system". In: *Boston University* 28 (2012).

[20] Matej Guid and Ivan Bratko. "Computer Analysis of World Chess Champions." In: *ICGA Journal* 29 (June 2006), pp. 65–73.

[21] C Goldammer. *Project Title*. https://github.com/cgoldammer/chess-analysis/blob/master/position_sharpness.ipynb. 2018.

[22] Frenk Van Harreveld, Eric-Jan Wagenmakers, and Han LJ Van Der Maas. "The effects of time pressure on chess skill: an investigation into fast and slow processes underlying expert performance". In: *Psychological research* 71 (2007), pp. 591–597.

[23] *lichess.org open database*. https://database.lichess.org. Accessed: 2023-10-10.

[24] *Stockfish 16*. https://stockfishchess.org/blog/2023/stockfish-16/. Accessed: 2023-10-10.

[25] Fabian Pedregosa et al. *Scikit-learn: Machine Learning in Python*. 2018. arXiv: 1201.0490 [cs.LG].

[26] Tianqi Chen and Carlos Guestrin. "XGBoost". In: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, Aug. 2016. DOI: 10.1145/2939672.2939785. URL: https://doi.org/10.1145%2F2939672.2939785.

[27] Konstantinos Pelechrinis. *iWinRNFL: A Simple, Interpretable Well-Calibrated In-Game Win Probability Model for NFL*. 2018. arXiv: 1704.00197 [stat.AP].

[28] Sacha Droste and Johannes Fürnkranz. "Learning the piece values for three chess variants". In: *ICGA Journal* 31.4 (2008), pp. 209–233.