Project Report on
Player Strength Classification from Chess Game Notations

Aw Thura, Dhruv Patel

July 6, 2023

1 Introduction

Chess is a complex game that has attracted players for centuries and there is plenty of game data available for analysis. One crucial aspect of assessing a player's skills and strength is their Elo rating, a measure of one's relative skill level compared to other players. This rating system allows a standardized assessment of skills relative to other players and facilitates the uniform pairing of matches. However, the traditional calculation of the Elo is primarily based on game outcomes. Thus, if two random unrated players are playing chess in a park, there is no procedure to estimate or calculate their chess rating. This limitation motivated the need for a more granular approach to chess rating system. By introducing machine learning techniques, it could be possible to analyze the moves made in a chess game and predict a player's Elo rating based on the game's progression. This research aims to explore the potential of machine learning algorithms in accurately predicting a player's Elo rating and generate the features which influence a player's performance.

The inspiration for this study came from personal interest in Chess and the desire to explore how machine learning techniques can be used in this domain. Elo has been used for some time to measure a player's skill level, but the ability to forecast Elo based on moves made in the game opens up new study opportunities into the underlying aspects that affect a player's success. By analyzing the move sequences, it might be possible to identify patterns and strategic choices that indicate a player's degree of experience. A multitude of chess-related applications, including matching algorithms, player progress tracking, and adaptive coaching systems, may also benefit from accurate Elo prediction models.

The objectives of this research are: first, to analyze chess games and extract features that convey key characteristics of the game and player performance; and second, to develop a machine learning model that can accurately predict a player's Elo based on the extracted features. By achieving these goals, we aim to provide insights into the elements that affect a player's Elo rating and demonstrate the benefit of machine learning techniques in chess analysis.

2 Data

2.1 Dataset

For the dataset, we went through several candidates and eventually chosen the 60,000+ Chess Games Dataset[AdityaJha1504, 2021] from kaggle. The following table (see Table 1) briefly explain the contents in the datatset.

Element	Description
White username	Username of the white player
Black username	Username of the black player
White id	Link to other details of white player
Black id	Link to other details of black player
White rating	White's ELO rating
Black rating	Black's ELO rating
White result	White's win or the loss condition
Black result	Black's win or the loss condition
Time class	Type of game (blitz, bullet, rapid or daily)
Time control	Total time + Time increment
Rules	Either normal chess or other variants
Rated	Whether ELO points are at stake
FEN	Standard notation for describing a particular board position of a chess game
PGN	Standard plain text format for recording chess game

Table 1: Dataset Overview

FEN (Forsyth–Edwards Notation) and PGN (Portable Game Notation) are particularly significant elements in this dataset since they include crucial information on the game.

2.2 Sampling

The acquired dataset has 66,879 samples. To extract analysis from game FEN and game PGN, every move of every single game in the dataset will later be analyzed through a Chess Engine, which could be computationally expensive. Thus, it was decided that the data to be sampled down to approximately 3,000 data points, which could be feasibly processed by the local computer. First, a degree of data cleaning was performed. Specifically, chess variant games that deviate from traditional chess rules were eliminated, and games with handicapped scenarios where one player had a reduced number of chess pieces were also excluded.

To ensure a representative sample, data analysis is also carried out on the original dataset. It was observed that games under "blitz" time control comprises a large portion of our dataset (see Figure 1). Additionally, a majority of the games in the dataset were

played in 2021(see Figure 2). Thus, after thorough analysis , the decision was made to deviate from the traditional approach of random sampling. Specifically, games played under "blitz" time control within a specific five-day period, ranging from May 20th to May 24th, 2021, were identified as the preferred subset for sampling. Blitz games are selected due to their fast pace, and the frequency in the dataset. The chosen five-day period provided a time frame that balanced a sufficient volume of games with a manageable scope for analysis. This resulted in 3,196 rows of data in the sample dataset.

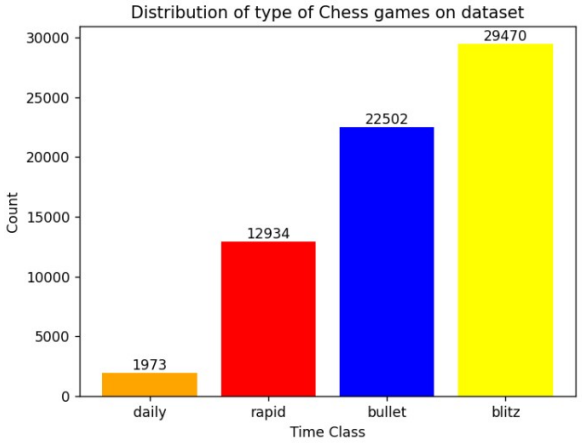


Figure 1: Chess Games in the dataset under different time control formats

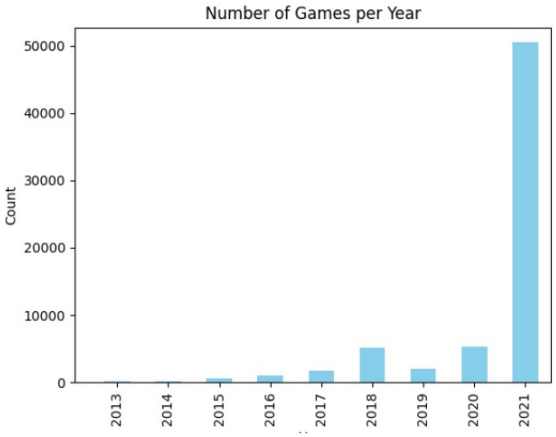


Figure 2: Chess Games in the dataset based on the year played

2.3 Feature Engineering

Currently, a challenging and intricate problem arises: how to convert several lines of chess notation, as illustrated (see Figure 3), into informative suitable for the model's input? Initially, the chess game PGN includes various complex and unnecessary information for our approach. Therefore, regular expressions (RegEx) were utilized to extract the moves in the Short Algebraic Notation (SAN) format (see Figure 4).

"pgn": "[Event \"Live Chess\"]\n[Site \"Chess.com\"]\.....**(Other data)**.....[Link\"https://www.chess.com/game/live/15211330359\"]\n\n1. e4 {[%clk 0:05:00]} 1... c5 {[%clk 0:05:00]} 2. Nf3 {[%clk 0:04:58.8]} 2... e6 {[%clk 0:04:59.4]} .....**(The Rest of Moves)** ..... 40... Kf8 {[%clk 0:00:43.9]} 41. Rh8# {[%clk 0:02:20.6]} 1-0\n",

Figure 3: Initial game PGN

"moves_san": "e4 e5 Qe2 Nf6 d3 Nc6 c3 Bc5 Nf3 h6 Be3 Bxe3 Qxe3 O-O Na3 Re8 O-O-O d5 d4 Bg4 h3 Bxf3 gxf3 exd4 cxd4 dxe4 f4 Nb4 Kb1 b6 Nc4 Qd5 b3 b5 Ne5 a5 Bxb5 Qxb5 a4 Qd5 Rhg1 Rac8 Rxg7+ Kxg7 Rg1+ Kf8 Rg6 fxg6 Nxg6+ Kf7 Ne5+ Ke7 Ng6+ Kd7 Ne5+ Kd8 Nf7+"

Figure 4: Extracted moves from game PGN

After obtaining the moves in SAN format, the moves are then passed into the chess engine Stockfish[Wikipedia contributors, 2023b]. Stockfish outputs the evaluation in a standard format.

```
"PovScore(Cp(-541), BLACK)",
"PovScore(Cp(+540), WHITE)",
"PovScore(Cp(-535), BLACK)",
"PovScore(Cp(+546), WHITE)",
"PovScore(Cp(-522), BLACK)",
"PovScore(Cp(+533), WHITE)",
"PovScore(Cp(-535), BLACK)",
"PovScore(Cp(+555), WHITE)",
"PovScore(Cp(-463), BLACK)",
"PovScore(Mate(+1), WHITE)",
"PovScore(Mate(-0), BLACK)"
```

The analysis generated was then filterd out for white player's perspective. Furthermore, to align the evaluation score with the format utilized by chess.com, the extracted values were standardized. This standardization involved scaling down the evaluation values by multiplying them with 0.01.Notably, checkmates or guaranteed checkmates were assigned the highest evaluation scores in the chess.com format, with +20 for the white player and -20 for the black player. This resulted in evaluation vectors, capturing evaluations after each and every move for every game, represented by:

$$E_m = [e_1, e_2, \ldots, e_n] \tag{1}$$

where,
E_m = Evaluation vector for game m
e_n = Evaluation after n-th move
n = Total number of moves for game m

From this extracted feature, the evaluation vector of any game in the dataset can be visualized to gain more insights on the game.(see Figure 5)

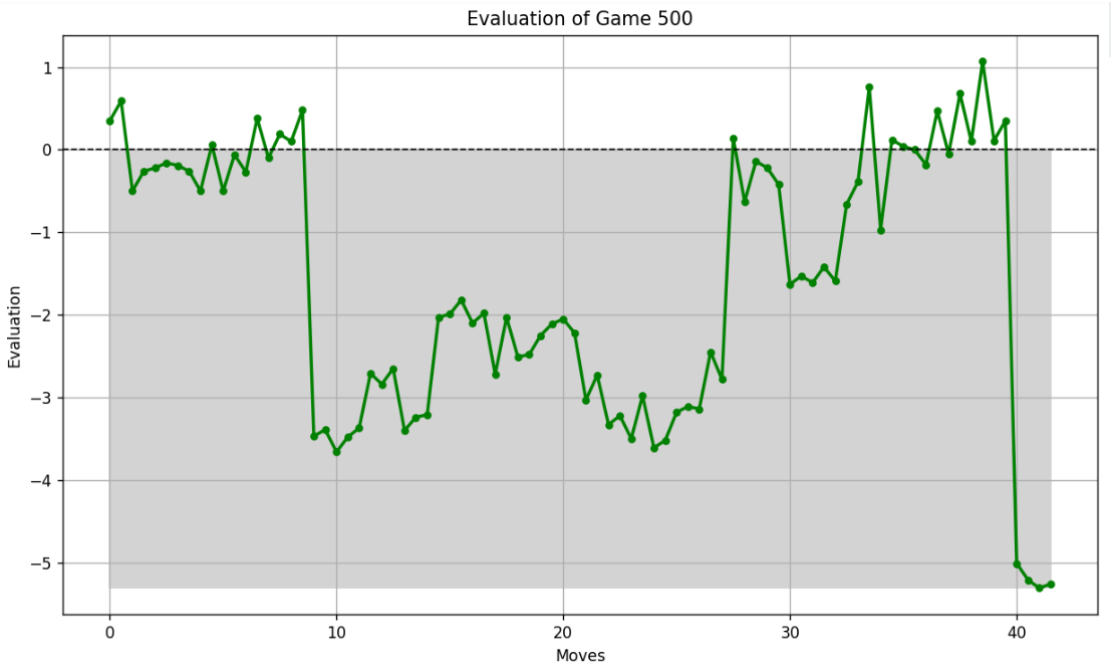


Figure 5: Evaluation Graph of Game 500

Next, the material imbalance was computed using the game FEN, which represents the board in an 8x8 format. Presented below is a sample of the board FEN:

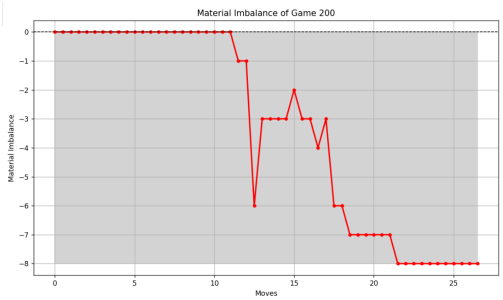
```
"4Q1k1/2p2p1p/p3p1pB/3rP3/1P3P1q/2P5/6PP/5RK1 b - -"
```

where, the letters correspond to different chess pieces, while the numbers indicate the count of empty squares between the pieces. To extract the "piece value count" [Wikipedia contributors, 2023a] for both the white and black players, regular expressions (Regex) were used. By calculating the difference between the two vectors, material imbalance was obtained which reflects the discrepancy in material throughout the game. This vector is represented as follows:

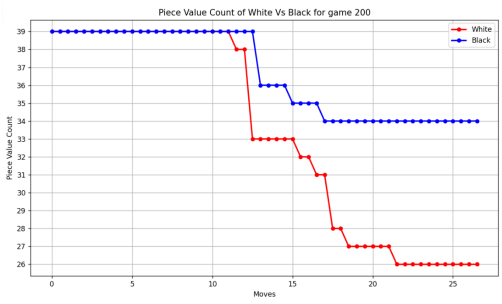
$$I_m = W_m - B_m \tag{2}$$
$$W_m = [wv_1, wv_2, \ldots, wv_n] \tag{3}$$
$$B_m = [bv_1, bv_2, \ldots, bv_n] \tag{4}$$

where,
I_m = Imbalance vector for game m
W_m = White player's piece value count for game m
B_m = Black player's piece value count for game m
wv_n = White player's piece value count after move n
bv_n = Black player's piece value count after move n

To gain more insights, the material imbalance vector (see Figure 6a), and the piece value count vectors (see Figure 6b) could also be visualized by graphs.



(a) Material Imbalance Graph



(b) Piece Value Count Comparison Graph

Through an analysis of the visualization of evaluation values throughout the game, a method was devised to compute a scalar value that potentially holds significance for the model. The approach involved calculating the area under the curve above zero (white player's advantage) and the area under the curve below zero (black player's advantage), which signifies the extent of dominance exhibited by each player over the course of the game. To standardize these values, they are divided by the total number of moves in the game. By comparing the difference and the ratio between these standardized values, scalar values can be generated to indicate the level of dominance of one player over the other. Therefore, the relative game advantage of the player controlling the white pieces is given by:

$$RA_m = \frac{1}{n}\left(\int_{x_1}^{x_n} max(y_i, 0)\, dx + \int_{x_1}^{x_n} min(y_i, 0)\, dx\right) \tag{5}$$

where,
RA_m = Relative game advantage of white player during game m
n = number of move

In the above equation, it is important to note that the + sign is used because the

evaluation of the black player from white's perspective is negative by default. Similarly, The ratio between the two areas is given by:

$$AR_m = \frac{\int_{x_1}^{x_n} max(y_i, 0)\, dx}{\int_{x_1}^{x_n} min(y_i, 0)\, dx} \tag{6}$$

where,
AR_m = Area ratio of game m
n = number of moves for game m

Furthermore, the amount of advantage swings in each game and the number of blunders made by each player were calculated. The advantage swings were determined by counting the instances when the game advantage shifted between the players. This count represents the frequency of advantage swings throughout the game and is expressed as:

$$S_m = \sum_{i=1}^{n} a(i) \cdot b(i) \tag{7}$$

where,
S_m = Amount of advantage swings during game m
$a(i)$ = Boolean function that represents evaluation change from negative to positive or vice versa on i-th move
$b(i)$ = Boolean function that represent evaluation change with a margin of 1 on i-th move
n = number of moves for game m

For the number of blunders, each time a player's evaluation loss exceeded a margin of 3 was counted. Each instance of such an evaluation loss can be considered a blunder. This count represents the number of blunders and can be represented as:

$$S_m = \sum_{i=1}^{n} c(i) \tag{8}$$

where,
WB_m = Amount of blunders by white player during game m
$c(i)$ = Boolean function that represents evaluation change with a margin of 3 on i-th move
n = number of moves for game m

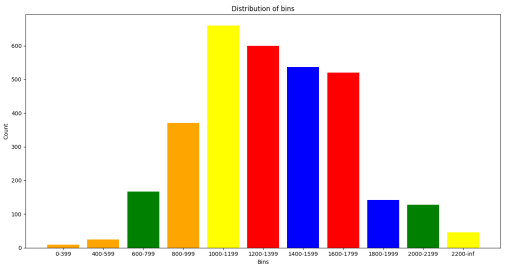
Following the whole feature engineering process, the resulting features are presented in the table below:

Table 2: Final Feature Table

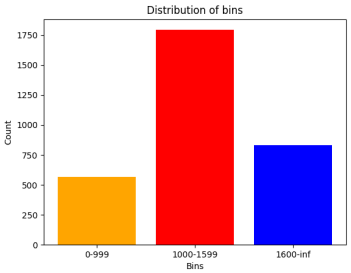
Element	Description
White result	White's win or loss condition
Black result	Black's win or loss condition
White bin	White's ELO range
Black bin	Black's ELO range
Material Imbalance	Vector of the material difference after each move (Equation 2)
White material count	White's piece value after each move
Black material count	Black's piece value after each move
Opening name	Name of played chess opening
Eval white perspective	Vector of the evaluation of the game from white's perspective (Equation 1)
Area ratio	Ratio of the amount of advantage between the players (Equation 6)
Relative game advantage	Relative game advantage (Equation 5)
Area ratio rev	Reverse of Area ratio
Advantage swings	Count of the changes in evaluation lead throughout the game (Equation 7)
White blunders	Count of blunders by white player (Equation 8)
Black blunders	Count of blunders by black player
Relative area white	Dominance area of white player
Relative area black	Dominance area of black player

2.4 Preprocessing

To treat the problem as a classification task per the project requirements, bins with intervals of ratings are generated for this step. This step holds critical importance as the output of the model will be defined by the created bins. Two approaches were implemented. Firstly, a more specific 11 bins approach (see Figure 6a) and a more generalized 3 bins approach (see Figure 6b).



(a) Bins separated to 11 classes



(b) Bins separated to 3 classes

For the preprocessing of features, normalization techniques were applied to account for negative values in the dataset. Two methods, namely standard scalarization and min-max scalarization, were employed. Standard scalarization transforms the features to have zero mean and unit variance, while min-max scalarization scales the features to a specific range, typically [-1, 1]. The selection of the optimal scalarization method was based on the performance of the trained model, ensuring the best results in terms of model accuracy.

3 Method

Following the feature generation, a correlation matrix was plotted to gain deeper insights into the features (see Figure 6). The analysis revealed that all of the numerical features exhibited correlation scores of less than 0.2. Consequently, it became evident that training the models using traditional machine learning methods would pose a challenge.

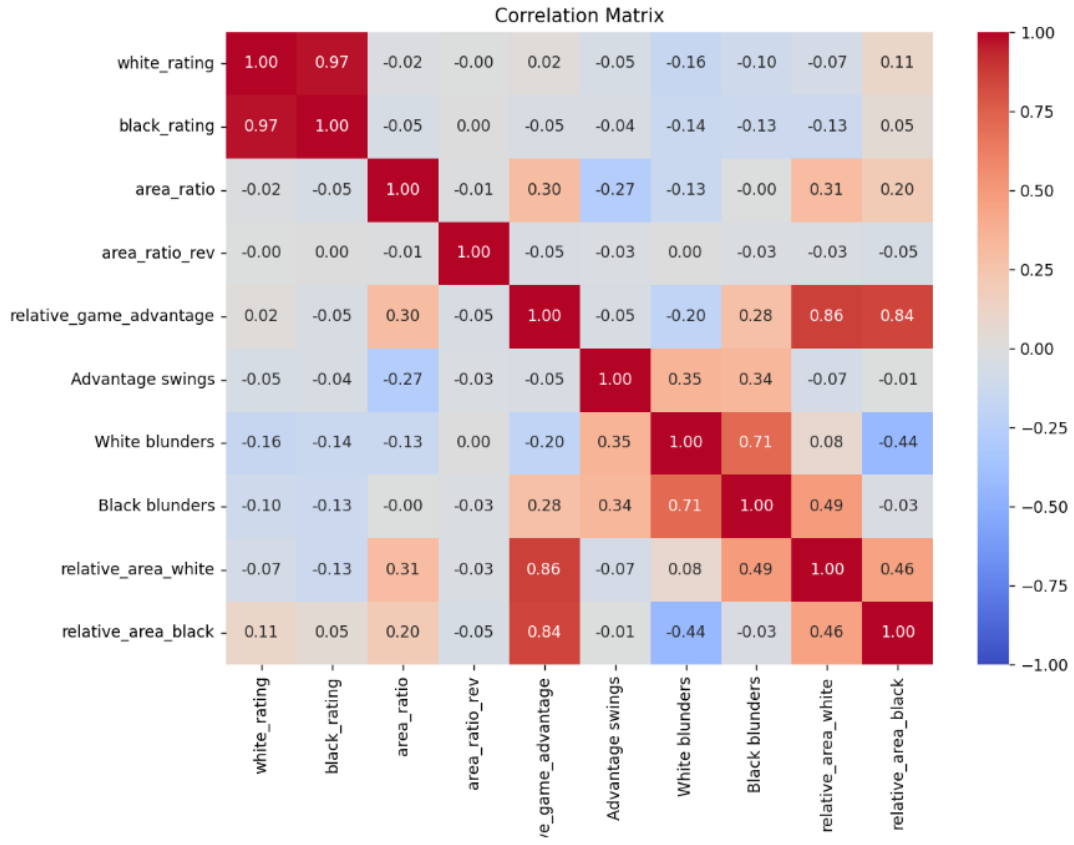


Figure 6: Correlation Matrix

3.1 kNN model

As defined, two different approaches were adopted to define the output of the problem: a specific approach with 11 classes and a generalized approach with 3 classes. In the initial phase of our research, a K-Nearest Neighbors (KNN) machine learning algorithm with an optimized k value was trained. However, this model was limited in its ability to incorporate sequential features, which meant that only numerical and categorical features could be utilized. As a result, the crucial sequential features remained unused in this approach.

Hyperparameter	Values
Test size	0.25
Neighbors	optimized
Cv	5
Feature Range for Minmax scalar	-1 - 1

Table 3: Range of Hyperparameters experimented for the kNN model

3.2 Random Forest Classifier

Furthermore, the effectiveness of a Random Forest classifier was explored by testing a range of estimators from 100 to 500. Despite the higher complexity and increased number of estimators, this approach also yielded suboptimal results for the research objectives.

Hyperparameter	Values
Test size	0.25
Estimators	100 - 500
OOb scores	True

Table 4: Range of Hyperparameters experimented for the Random Forest Classifier

3.3 Long-Short Term Memory (LSTM)

The initial experiments with KNN and Random Forest classifiers highlighted the challenges of incorporating sequential features and achieving satisfactory performance. Consequently, the need for a more suitable approach to handle such features is recognized. After research, a viable solution arises in Recurrent Neural Networks (RNNs). RNNs have the capacity to capture temporal dependencies in the data, providing improved performance and deeper insights. Consequently, it was decided to explore Long short-term memory (LSTM), a specialized type of RNN that addresses the vanishing gradient problem, as a promising solution to enhance the outcomes of the project.

With the implementation LSTM, the sequential features can be eventually utilised. The LSTM model architecture consists of a Sequential model with an initial LSTM layer of 32 units, followed by a Dense layer of 32 units. A Dropout layer was then included to prevent overfitting, followed by the final softmax prediction layer. Three activation functions, sigmoid, ReLu and Tanh are tried out and Hyperparameters such as learning rate and dropout rate are then tuned to obtain the optimal results.

Hyperparameter	Values
Test size	0.2
Learning rate	0.1 - 0.001
Epoches	0.1 - 0.001
Batch size	32,64
Activation Functions	ReLu, Sigmoid, Tanh
Class weights (for 3-class classifier)	0: 2.0, 1: 1.0, 2: 1.8

Table 5: Range of Hyperparameters experimented for the LSTM model

4 Results and discussion

4.1 Specific 11-class classifier approach

Average Type	Precision	Recall	F1 score
Macro Average	0.09	0.09	0.08
Weighted Average	0.15	0.18	0.16

Table 6: Results of kNN model with optimized k and Standard Scalar

Average Type	Precision	Recall	F1 score
Macro Average	0.12	0.11	0.11
Weighted Average	0.17	0.18	0.17

Table 7: Results of Random Forest model with 500 estimators and Standard Scalar

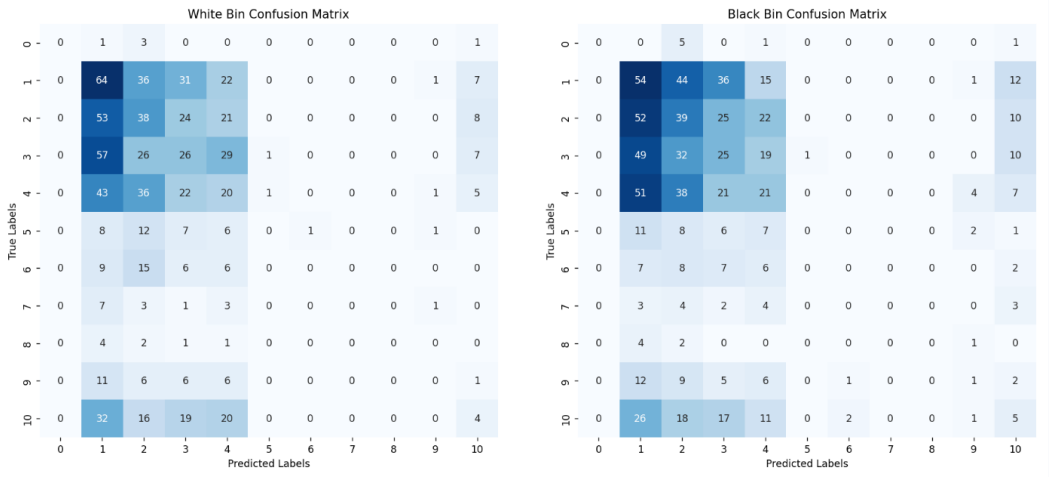


Figure 7: Confusion Matrix for 11-class kNN Classifier

Average Type	Precision	Recall	F1 score	Precision(train)	Recall(train)	F1 score(train)
Macro Average	0.09	0.11	0.10	0.42	0.21	0.23
Weighted Average	0.18	0.21	0.19	0.37	0.33	0.31

Table 8: Results of LSTM trained 20 epoches

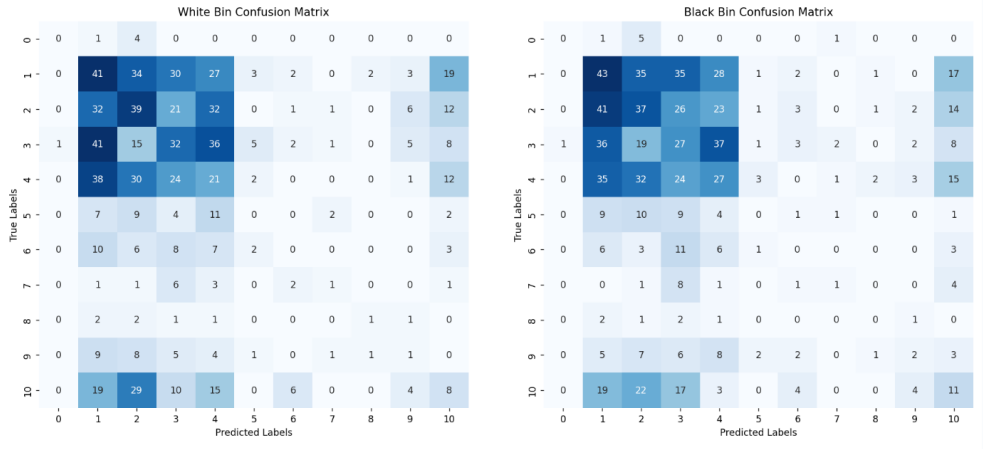


Figure 8: Confusion Matrix for 11-class Random Forest Classifier

Average Type	Precision	Recall	F1 score	Precision(train)	Recall(train)	F1 score(train)
Macro Average	0.09	0.09	0.08	0.74	0.59	0.64
Weighted Average	0.15	0.16	0.15	0.69	0.65	0.65

Table 9: Results of LSTM trained 1000 epoches

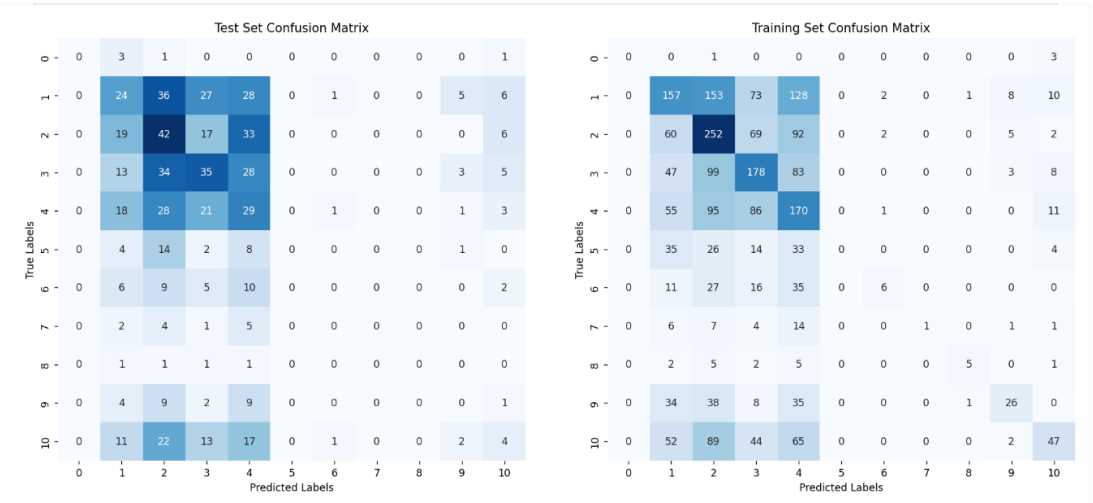


Figure 9: Confusion Matrix for 11-class LSTM (20 epoches)

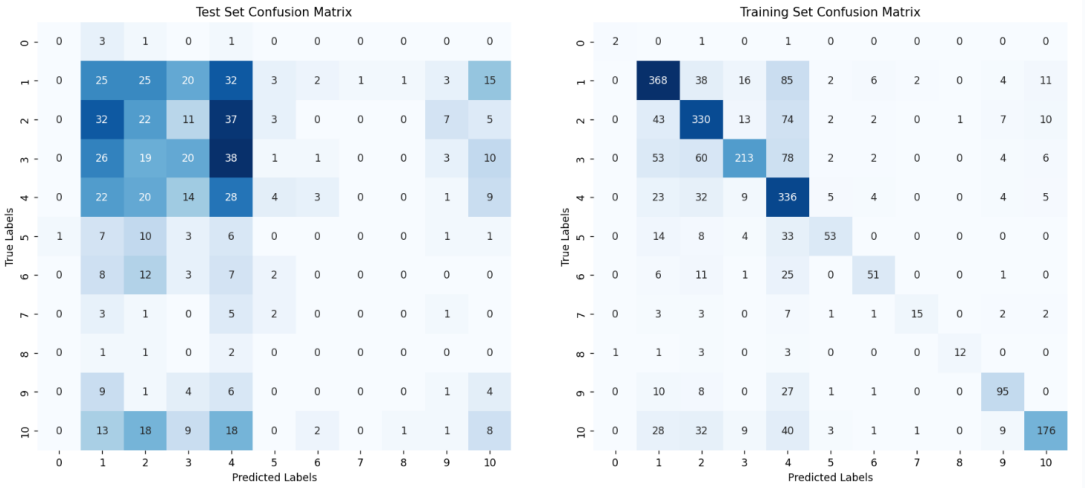


Figure 10: Confusion Matrix for 11-class LSTM (1000 epoches)

Method	Accuracy
Random choice	0.091
kNN	0.18
Random Forest Classifier	0.18
LSTM (20 epoches)	0.21
LSTM (1000 epoches)	0.18
LSTM (20 epoches) - Train	0.33
LSTM (1000 epoches) - Train	0.65

Table 10: Selected model accuracies compared to random choice score

4.2 Generalized 3-class classifier approach

Average Type	Precision	Recall	F1 score
Macro Average	0.38	0.35	0.32
Weighted Average	0.45	0.53	0.45

Table 11: Results of 3-class kNN model with optimized k and MinMax Scalar

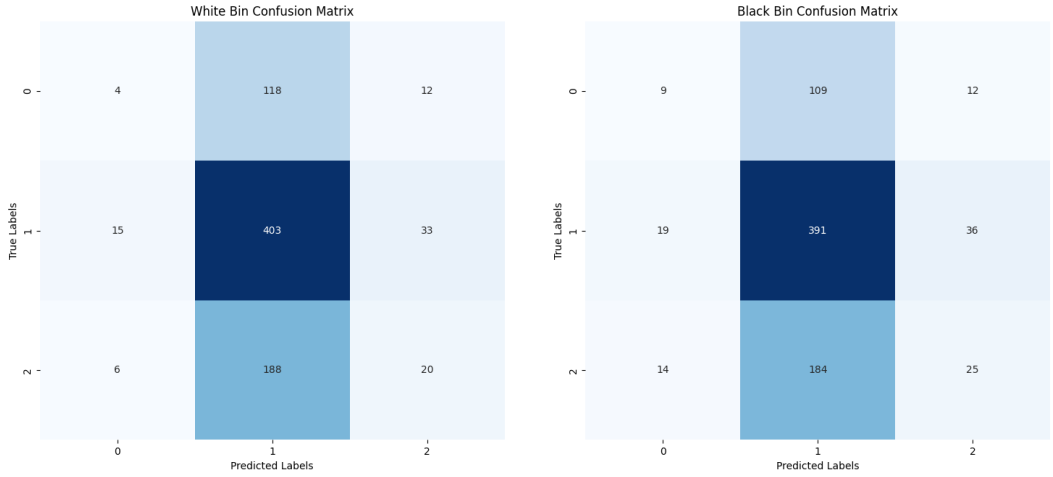


Figure 11: Confusion Matrix KNN 3-class classifier

Average Type	Precision	Recall	F1 score
Macro Average	0.39	0.36	0.34
Weighted Average	0.45	0.51	0.45

Table 12: Random Forest 3-class classifier results with 500 estimators and Standard Scalar

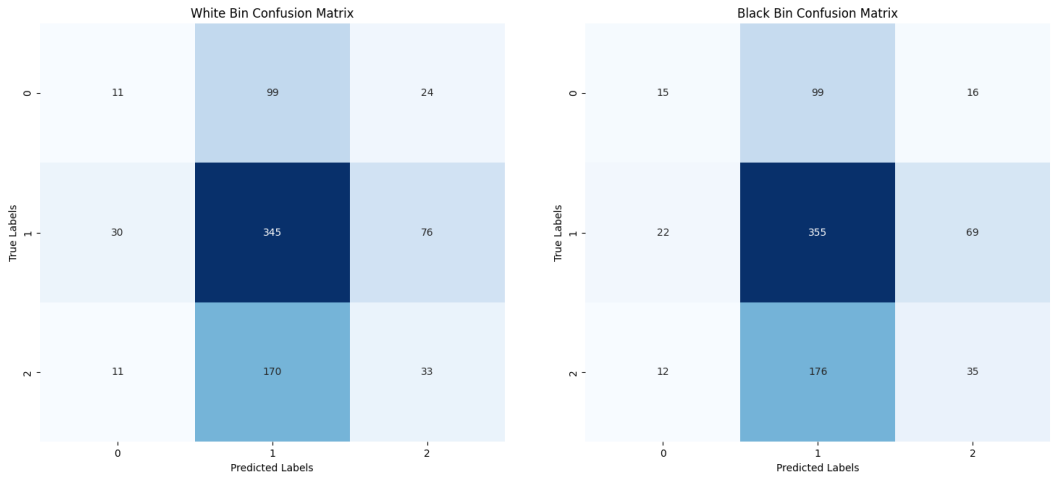


Figure 12: Confusion Matrix Random Forest 3-class Classifier

Average Type	Precision	Recall	F1 score	Precision(train)	Recall(train)	F1 score(train)
Macro Average	0.42	0.36	0.34	0.48	0.39	0.38
Weighted Average	0.47	0.54	0.47	0.51	0.55	0.49

Table 13: Results of 3-class classifier LSTM trained 10 epoches

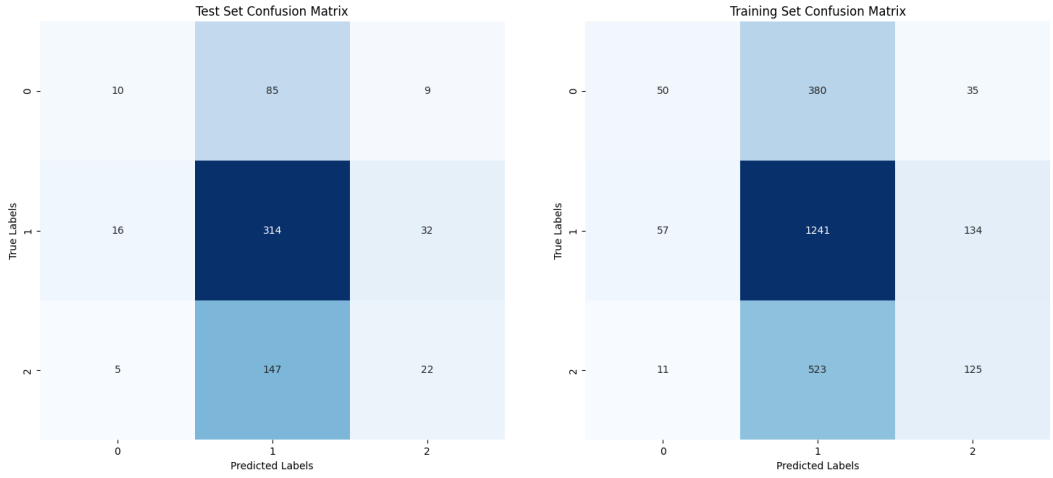


Figure 13: Confusion Matrix LSTM 3-class Classifier, 10 epoches

Method	Accuracy
Random choice	0.33
kNN	0.53
Random Forest Classifier	0.51
LSTM (10 epoches)	0.54
LSTM (10 epoches) - Train	0.55

Table 14: Model accuracies for 3-class classifier compared to random choice score

4.3 Discussion

Initially, the implementation was focused on 11-class classifier solely. However, the model evaluation results were sub-optimal, with the accuracy scores only slightly surpassing the random choice. The expectation was that incorporating LSTM would lead to improved results; however, this proved not to be the case. Furthermore, it was observed that for the 11-class LSTM classifier, the model evaluation values decreased on the training set after training 1000 epoches, indicating a sign of overfitting.

This motivated a change in perspective of the problem, leading to the implementation of 3-class classifier. However, even for this classifier, the LSTM model began to overfit after approximately 10-15 epochs. Despite this, a slight improvement in results was observed compared to the previous iterations. For both 3-class and 11-class classifiers, it could be observed from the accuracy tables that LSTM performs marginally better.

In conclusion, it can be determined that LSTM remains a more favorable approach compared to traditional classifiers. Additionally, the model evaluation scores on the generalized 3-class classifier are superior to those of specific 11-class classifier approach, as the random choice score is already higher. It is also observed from the behavior of confusion matrices that the models tends to predict the classes with higher occurrence more than the low-occurrence classes. To counter this, Weighted features were introduced to the LSTM and the weights were tuned accordingly. This improved the performance on classes with smaller samples but decreased the model accuracy as a whole.

To improve this model, further methods which can utilize the extracted sequential features could be explored. One promising candidate is the use of Transformers, which excel in processing sequential data through their attention mechanism. More data samples could be added to the training data since the neural network based models tend to generally perform better on a larger datasets. Furthermore, seeking advice from chess experts, such as chess grandmasters, can greatly contribute to improving feature engineering as some could already guess the Elo of their opponents just by playing a game with them. Overall, features other than the chess-engine evaluation could be extracted based on the level of understanding on the game by the researchers.

5 Summary

This project report focuses on player strength classification from chess game notations using machine learning techniques. The objective is to explore the potential of machine learning algorithms in accurately predicting a player's Elo rating and identify the features that influence a player's performance.

The report starts with an introduction to the significance of Elo ratings in assessing a player's skill level and the limitations of traditional Elo calculation methods. The motivation behind this research is to develop a more granular approach to chess rating by analyzing moves made in a game and predicting Elo ratings based on the game's progression.

The dataset used in the project is the 60,000+ Chess Games Dataset, obtained from Kaggle. The dataset includes various features such as player usernames, Elo ratings, game outcomes, time controls, and game notations in FEN and PGN formats. To extract meaningful insights from the dataset, data sampling and cleaning are performed. The dataset is sampled down to approximately 3,000 data points, considering computational limitations. Games with chess variants and handicapped scenarios are excluded. The sampling process focuses on blitz games played within a specific five-day period to ensure a representative sample.

Feature engineering is a crucial step in converting chess notations into suitable inputs for the machine learning model. The challenge lies in transforming the complex lines of chess notation into informative features. Further analysis and preprocessing techniques are employed to extract key characteristics from the game notations.

The report discusses the machine learning models and algorithms employed for player strength classification, including k-nearest neighbors, random forests and LSTM. The performance of these models is evaluated using evaluation metrics such as accuracy, confusion matrices, and classification reports.

The findings reveal that machine learning models, particularly the LSTM-based model, marginally outperform traditional classifiers in predicting player Elo ratings. The models showcase the ability to capture patterns and strategic choices that indicate a player's experience and performance. The project highlights the potential applications of accurate Elo prediction models in various chess-related domains, such as matchmaking, player progress tracking, and adaptive coaching systems.

In conclusion, this project demonstrates the potential of machine learning techniques in analyzing chess game notations and predicting player Elo ratings. The research provides valuable insights into the factors influencing a player's performance and highlights the benefits of incorporating machine learning in chess analysis.

References

[AdityaJha1504, 2021] AdityaJha1504, C. (2021). 60,000+ chess game dataset (chess.com). Available online at: https://www.kaggle.com/datasets/adityajha1504/chesscom-user-games-60000-games.

[Wikipedia contributors, 2023a] Wikipedia contributors (2023a). Chess piece relative value — Wikipedia, the free encyclopedia. [Online; accessed 6-July-2023].

[Wikipedia contributors, 2023b] Wikipedia contributors (2023b). Stockfish (chess) — Wikipedia, the free encyclopedia. [Online; accessed 6-July-2023].