Chess Game Result Predictor

Hüseyin Alper Koyuncu
Department of Computer Engineering
TOBB ETÜ
Ankara, Turkey
h.koyuncu@etu.edu.tr

Pelin Buçukoğlu
Department of Computer Engineering
TOBB ETÜ
Ankara, Turkey
pbucukoglu@etu.edu.tr

***Abstract*—The goal of this project is to predict the result of a chess game (white win, black win, or draw) using machine learning classification techniques. The analysis is based on a Kaggle chess game dataset, which includes features such as player ratings, game type (rated or unrated), number of moves, and opening codes. Multiple models, including logistic regression, k-Nearest Neighbors, Random Forest, and Support Vector Machines, achieved similar accuracies of approximately 61%. Observations align with the literature, as all models struggled to predict draws accurately, highlighting the inherent challenge of identifying draw outcomes in chess. This model provides insights into the factors influencing game results.**

***Index Terms*—Chess, Machine Learning, Prediction, Game Analysis, Classification**

I. Introduction

Chess has been a key area of research in artificial intelligence (AI) for many years. It has often been used to test decision-making algorithms because of its complexity and the need for strategic thinking. Over time, AI systems have made significant progress in chess, evolving from rule-based engines to deep learning systems that can beat world champions. While much of the focus has been on AI systems analyzing moves or recommending strategies, predicting the outcome of a chess game based on basic features has not been studied as much.

This project aims to predict the result of a chess game—classified as a white win, black win, or draw—using machine learning classification techniques. The dataset used for this analysis comes from Kaggle and originally contained 20,000 games, which was reduced to 17,700 after cleaning and preprocessing. The features include player ratings, whether the game was rated or not, the number of moves, and the opening code. Unlike deep learning methods that examine moves or positions, this project uses high-level features to understand what factors influence game outcomes.

Our work builds on studies like the one by Drezewski and Wator, who used deep learning and sequential data to predict chess outcomes [1]. Their approach achieved 69% accuracy by using detailed chessboard representations. In comparison, our project focused on simpler features and achieved up to 61% accuracy using models like k-Nearest Neighbors, Logistic Regression, Random Forest, and Support Vector Machines. Attempts to oversample draws, which are inherently harder to predict, lowered the overall accuracy, highlighting the difficulty of identifying draw outcomes with this approach.

This project shows that simpler models can still provide useful insights into what influences chess game results. However, it also demonstrates the limits of using high-level features compared to methods that analyze moves or board positions. By applying machine learning, this project contributes to understanding how game results are affected by key factors and paves the way for more detailed future studies.

II. Background

Before diving into the problem, it is essential to explain some chess-specific terms to help readers understand the concepts used in this project. If you are unfamiliar with the basic chess rules, such as the pieces and how they move, we recommend researching those first, as they are simple and quick to learn.

A. Castling

Castling is a special move in chess where the king and rook change positions. It is generally used to place the king in a safer position.

B. Resign

Resigning means a player accepts the loss and ends the game before a checkmate.

C. Mate

Mate, short for checkmate, occurs when the opponent's king is trapped and under threat. It ends the game immediately, with the mated player losing.

D. Blundering

Blundering refers to a significant mistake made by a player during a game. Such errors often result in the loss of a major piece, substantial material disadvantage, or even an immediate end to the game. Blunders are typically game-changing and are avoided by more experienced players through careful calculation and strategic planning.

E. Player Rating (ELO)

Player rating, often referred to as ELO, is a numerical value that represents a player's skill level. Matchmaking systems typically pair players with similar ratings to ensure fair games. This means the ratings of white and black players are often close. However, the higher-rated player is generally assumed to be the stronger one. The ELO system becomes more accurate

as players compete more, as their ratings adjust to reflect their true skill. Occasionally, a player might be underrated—for example, someone who has studied extensively or plays over-the-board (OTB) chess instead of online. In such cases, the lower-rated player might be the better one until they play enough games to reach their deserved ELO.

F. Increment Code (Time Control)

Time control specifies the time constraints for each player in a game. For instance, "10+5" means each player starts with 10 minutes and gains an additional 5 seconds for every move. This feature was removed during preprocessing because both players play under the same time control, making it irrelevant to determining which player performs better.

G. Opening Name and Opening ECO

Chess openings refer to the initial moves of a game, which are often theorized and named. These openings can have distinct characteristics. For example, games starting with the Sicilian Defense are typically sharp and tactical, often leading to opposite-side castling. Italian Game openings, in contrast, are common and tend to result in more drawish positions. Opening names provide such information, while the Opening ECO (Encyclopaedia of Chess Openings) is an encoded version of the opening name.

H. Moves

Chess moves are stored using standard notation, with columns labeled A through H and rows numbered 1 through 8. For example, "Nf3" indicates a knight moving to the f3 square. This feature was not used in the project because analyzing board positions requires advanced AI engines like Stockfish, which goes beyond the scope of this work.

III. Related Work

Machine learning techniques have been increasingly applied to predict chess game outcomes in recent years, utilizing diverse approaches and datasets. Studies in this area often focus on different methodologies, ranging from high-level metadata analysis to low-level, sequential data modeling.

One group of studies emphasizes sequential data and deep learning techniques. Drezewski and Wator [1] used long-short-term memory (LSTM) networks to analyze chess moves as sequential data, achieving 69% accuracy by experimenting with various chessboard representations, such as bitmap and algebraic notation. Similarly, Reddy et al. [2] applied recurrent neural networks (RNNs) to model temporal patterns in chess games, focusing on dynamic strategies and move sequences. These deep learning approaches highlight the potential of low-level data, such as individual moves and board positions, to improve accuracy. In contrast, our project takes a higher-level approach, analyzing metadata such as player ratings, game type, and opening codes, offering a lightweight alternative that avoids complex sequence modeling.

Another group of works incorporates move sequences and dynamic game analysis. Masud et al. [3] proposed an ensemble-based framework to predict chess outcomes while the game is in progress. By segmenting training data based on player profiles and dynamically predicting outcomes after each move, their system achieved 66% accuracy, often making correct predictions nine or more moves before the game's conclusion. Similarly, Rosales Pulido [4] utilized the Gorgo Base dataset of three million games, analyzing board positions after twenty moves to predict outcomes with models like SVM and Random Forest. Lehana et al. [5] applied regression and Naïve Bayes classifiers on a dataset of 10,000 games, using move sequences and evaluation scores from the Stockfish engine. These granular approaches offer precise insights into game progression, while our work prioritizes simplicity by leveraging metadata to identify broader trends without analyzing individual moves.

Several studies have focused on chess metadata and related factors influencing game results. Mujagić et al. [6] explored a range of features affecting chess performance, such as player skill level, current form, and psychological factors. Their study demonstrated that ensemble techniques, particularly Random Forest classifiers, outperformed simple classification methods, achieving an accuracy of 68.67% in predicting game outcomes. Similarly, Thabtah et al. [7] investigated Elo ratings as predictors of chess outcomes, using machine learning models like decision trees and neural networks. Their findings emphasize the significance of Elo scores, which align with our project's use of player ratings. DeCredico [8] also examined metadata, focusing on win rates and draw rates for specific openings. While their dataset contained over 2.5 million games, our project uses a smaller dataset of 20,000 games (17,700 after cleaning) and focuses on a minimal feature set to explore broader game trends.

Other works explore unique prediction frameworks and models. Tijhuis [9] developed a framework to predict player ratings based on features extracted from individual games, such as move quality and player behavior. Their system achieved 64.26% accuracy for rating predictions based on broad classes and 79.29% accuracy when distinguishing between extreme rating ranges. This approach focuses on rating prediction rather than game outcomes, but it shares similarities with our project in using game metadata for classification tasks. Saaty and Vargas [10] proposed a hierarchical analysis framework that combines technical and behavioral characteristics of chess players to predict championship match outcomes. Their methodology considers psychological and environmental factors, offering insights into the complexity of game prediction. These studies emphasize broader decision-making frameworks and advanced segmentation techniques that can complement our lightweight approach to predicting game results.

In summary, this project differs from low-level deep learning approaches by focusing on high-level metadata and avoiding detailed sequence analysis. It also contrasts with metadata-focused studies by combining features like player ratings, game type, and opening codes in a lightweight framework. This balance between simplicity and feature diversity allows

us to explore general trends in chess game outcomes, achieving up to 61% accuracy across multiple machine learning models.

IV. Data

The dataset used for this project was obtained from Kaggle, specifically from the following source: https://www.kaggle.com/datasets/datasnaek/chess. It contains detailed information about chess games, including a variety of features such as player ratings, game metadata, and move sequences.

A. Features Before Cleaning

The original dataset consisted of the following features:

- **Game ID**: Unique identifier for each game.
- **Rated**: Indicates whether the game affects player ratings (True/False).
- **Start Time / End Time**: The start and end timestamps of the game.
- **Number of Turns**: Total number of moves in the game.
- **Game Status**: Victory status (checkmate, draw, resign, etc.).
- **Winner**: The winner of the game (white, black, or draw).
- **Time Increment**: Time control for the game (e.g., "10+5" means 10 minutes per player with 5 seconds added per move).
- **White Player ID / Rating**: The unique ID and Elo rating of the white player.
- **Black Player ID / Rating**: The unique ID and Elo rating of the black player.
- **All Moves**: The sequence of moves in Standard Chess Notation.
- **Opening Eco**: Standardized code for the opening played (e.g., "C50").
- **Opening Name**: The name of the opening (e.g., "Italian Game").
- **Opening Ply**: The number of moves in the opening phase.

B. Features After Cleaning

After the data cleaning process, the following features were retained for the analysis:

- **White Player Rating / Black Player Rating**: Numerical representation of player skill.
- **Rating Difference**: Feature added to represent the difference between white and black player ratings.
- **Number of Turns**: Total moves played in the game.
- **Opening Eco**: Encoded representation of the opening used.

The following features were removed:

- **Victory Status**: This feature was removed because it directly indicated draws in the dataset, allowing models to predict draws with 100% accuracy. This would undermine the objective of assessing the models' ability to predict outcomes based on gameplay characteristics.
- **Opening Ply**: This feature was deemed irrelevant for determining the winner and could confuse the model.
- **All Moves**: Analyzing move sequences requires a complex chess engine, which was beyond the scope of this project.
- **Duplicate Entries**: Removed to avoid biases in training.

C. Data Cleaning and Preprocessing

Several steps were taken to clean and preprocess the dataset:

- Removed the `opening_name` column, as it was redundant with the `opening_eco` column, which provides a more concise representation.
- Filtered out games with fewer than 20 moves unless they ended in a checkmate. Such games often result from player errors or external decisions and do not reflect strategic or tactical gameplay.
- Encoded categorical variables, such as `opening_eco`, into numerical values suitable for machine learning models.
- Removed games with incomplete or corrupted data entries.

D. Dataset Statistics

To provide a clear view of the dataset before and after cleaning:

- **Initial Dataset**: 20,000 games with 14 features.
- **After Cleaning**: 17,700 games with 4 primary features.

E. Data Splits

The cleaned dataset was divided into three subsets:

- **Training Set (80%)**: Used to train the machine learning models. The training set contains 14,169 samples, distributed as follows:
  - White Win: 7,056
  - Black Win: 6,475
  - Draw: 638
- **Validation Set (10%)**: Used to fine-tune hyperparameters and evaluate model performance during development. The validation set contains 1,771 samples, distributed as follows:
  - White Win: 883
  - Black Win: 800
  - Draw: 88
- **Test Set (10%)**: Used for the final evaluation of model performance. The test set contains 1,772 samples, distributed as follows:
  - White Win: 876
  - Black Win: 809
  - Draw: 87

V. System Design

This section outlines the system designed to predict chess game outcomes using machine learning. It includes the features used, the models implemented, and the strategies explored to address class imbalance.

A. Input Features

The input features for the models were selected to balance simplicity and relevance to the prediction task:

- **Rated (boolean)**: Indicates whether the game is rated or unrated.
- **Number of Moves (numeric)**: Total moves made in the game.
- **Opening ECO (categorical)**: Encoded representation of the chess opening used.
- **Rating Difference (numeric)**: The difference between the ratings of the white and black players.
- **White Player Rating / Black Player Rating (numeric)**: The Elo ratings of the players, used in all models except k-Nearest Neighbors (kNN), where they were excluded to avoid the curse of dimensionality.

B. Model Selection and Parameters

Several machine learning models were implemented to predict the game outcome as one of three classes: *white win*, *black win*, or *draw*. Each model was trained and evaluated using the same training, validation, and test splits. Below are the models and their configurations:

- **k-Nearest Neighbors (kNN)**:
  - Parameters: Number of neighbors (`k`) was varied from 1 to 60 to identify the optimal value.
  - Distance metric: Euclidean distance was used to compute neighbors.
  - Feature Selection: `White Rating` and `Black Rating` were excluded, with `Rating Difference` used instead.
- **Logistic Regression**:
  - Parameters: Regularization strength (`C`) was optimized.
  - Solver: Multi-class classification was handled using the one-vs-rest (OvR) strategy.
- **Random Forest**:
  - Parameters: Number of trees (`n_estimators`) and maximum tree depth (`max_depth`) were adjusted.
  - Class Weights: Adjusted during training to give more importance to the minority class (draws). However, this strategy reduced overall accuracy, as the model struggled to generalize effectively when prioritizing draws.
- **Support Vector Machines (SVM)**:
  - Kernel types: Linear, polynomial, and radial basis function (RBF) kernels were tested.
  - Parameter: Regularization parameter (`C`) was tuned.

C. Handling Class Imbalance

The dataset had a significant class imbalance, with far fewer draw outcomes compared to white and black wins. While we explored several strategies to address this imbalance, each attempt to prioritize draws ultimately reduced the overall performance. As a result, we decided to leave the draw class underrepresented, as it was the best way to maintain accuracy for the more common outcomes (white and black wins). Below are the strategies we tried and their effects:

- **Oversampling**: Artificially increasing the number of draw samples in the training set was attempted. However, this led to models overemphasizing draws, which are inherently harder to predict. As a result, the overall accuracy dropped significantly.
- **Class Weights**: For models like Logistic Regression, SVM, and Random Forest, class weights were adjusted to give more importance to draws. While this slightly improved draw predictions, it also reduced overall accuracy, as the models struggled to balance predictions for white and black wins.
- **Random Forest Specific Adjustments**: Additional parameters in the Random Forest model were tuned to prioritize draws. However, these changes backfired, making the model less effective at predicting white and black wins and further decreasing overall performance.

In the end, leaving the draw class underrepresented yielded the most balanced results, as any attempts to emphasize draws came at the cost of significant drops in accuracy for the other classes. This highlights the challenge of working with imbalanced datasets, especially when the underrepresented class is inherently difficult to predict.

D. System Workflow

The models followed a consistent workflow:

1) **Feature Selection**: Input features (`Rated`, `Number of Moves`, `Opening ECO`, `Rating Difference`, and player ratings) were selected based on relevance to game outcomes and simplicity.
2) **Model Training and Validation**: Each model was trained on the training set and hyperparameters were fine-tuned using the validation set.
3) **Final Testing**: The best-performing configuration of each model was evaluated on the test set.

E. Evaluation Metrics

The following metrics were used to evaluate model performance:

- **Accuracy**: The proportion of correctly predicted outcomes.
- **Confusion Matrix**: Provided insights into the distribution of correct and incorrect predictions across all classes.
- **Precision, Recall, and F1 Score (Macro-Averaged)**: Used to assess the model's ability to predict all classes, particularly the minority class (draws).

VI. Results

This section provides a detailed analysis of the performance of different machine learning models used to predict chess game outcomes. Metrics such as accuracy, precision, recall, and F1 score (macro-averaged) were used to evaluate the models. A summary of each model's performance is provided below, with corresponding tables of metrics and confusion matrices included at the end.

A. Model Performance Overview

a) k-Nearest Neighbors (kNN):: The kNN model, using the original dataset, achieved an accuracy of 59% and a macro-averaged F1 score of 0.40. The model performed poorly in predicting draws, with zero precision and recall for this class. This result highlights the difficulty of predicting underrepresented classes in imbalanced datasets.

b) kNN with Oversampling:: To address the class imbalance, the draw class was oversampled, leading to a reduced accuracy of 52%. While predictions for draws slightly improved, the model's overall performance suffered, as it misclassified a larger proportion of white and black wins.

c) Logistic Regression:: Logistic Regression performed slightly better, achieving an accuracy of 61% and a macro-averaged F1 score of 0.42. However, the model struggled with draws, with only minimal improvement in recall for this class. The class imbalance remained a significant challenge, even with weighted adjustments.

d) Random Forest:: The Random Forest model achieved an accuracy of 61%, similar to Logistic Regression. Attempts to address class imbalance by adjusting class weights during training negatively affected the model's ability to predict white and black wins. Draws remained the hardest class to predict, with a macro-averaged F1 score of 0.43.

e) Support Vector Machines (SVM):: The SVM model also achieved an accuracy of 61%, with a macro-averaged F1 score of 0.41. Like the other models, it failed to predict draws effectively, as evidenced by zero precision and recall for this class. Adjustments to kernels (linear, polynomial, RBF) yielded no significant performance gains.

B. Tables of Results

The tables below summarize the metrics and confusion matrices for each model:

Class	Precision	Recall	F1-score
Black Win	0.57	0.62	0.59
Draw	0.00	0.00	0.00
White Win	0.60	0.61	0.61
Macro Avg.	0.39	0.41	0.40
Accuracy		0.59	

TABLE I
PERFORMANCE METRICS FOR KNN ON THE TEST SET

	Predicted Black	Predicted Draw	Predicted White
Actual Black	503	0	306
Actual Draw	42	0	45
Actual White	341	0	535

TABLE II
CONFUSION MATRIX FOR KNN ON THE TEST SET

VII. DISCUSSION

One of the biggest challenges in this project was predicting games that ended in a draw. Unlike white or black wins, draws don't have clear features in the high-level data we used. They often share similar characteristics with the other two outcomes, like player ratings, number of moves, and opening choices.

Class	Precision	Recall	F1-score
Black Win	0.54	0.55	0.54
Draw	0.07	0.11	0.09
White Win	0.57	0.53	0.55
Macro Avg.	0.39	0.40	0.39
Accuracy		0.52	

TABLE III
PERFORMANCE METRICS FOR KNN WITH OVERSAMPLING ON THE TEST SET

	Predicted Black	Predicted Draw	Predicted White
Actual Black	442	58	309
Actual Draw	36	10	41
Actual White	345	70	461

TABLE IV
CONFUSION MATRIX FOR KNN WITH OVERSAMPLING ON THE TEST SET

Class	Precision	Recall	F1-score
Black Win	0.60	0.59	0.60
Draw	0.50	0.01	0.02
White Win	0.62	0.69	0.65
Macro Avg.	0.57	0.43	0.42
Accuracy		0.61	

TABLE V
PERFORMANCE METRICS FOR LOGISTIC REGRESSION ON THE TEST SET

	Predicted Black	Predicted Draw	Predicted White
Actual Black	481	1	327
Actual Draw	42	1	44
Actual White	273	0	603

TABLE VI
CONFUSION MATRIX FOR LOGISTIC REGRESSION ON THE TEST SET

Class	Precision	Recall	F1-score
Black Win	0.60	0.62	0.61
Draw	0.18	0.02	0.04
White Win	0.63	0.67	0.65
Macro Avg.	0.47	0.44	0.43
Accuracy		0.61	

TABLE VII
PERFORMANCE METRICS FOR RANDOM FOREST ON THE TEST SET

	Predicted Black	Predicted Draw	Predicted White
Actual Black	501	5	303
Actual Draw	46	2	39
Actual White	289	4	583

TABLE VIII
CONFUSION MATRIX FOR RANDOM FOREST ON THE TEST SET

Class	Precision	Recall	F1-score
Black Win	0.62	0.54	0.58
Draw	0.00	0.00	0.00
White Win	0.60	0.73	0.66
Macro Avg.	0.41	0.42	0.41
Accuracy		0.61	

TABLE IX
PERFORMANCE METRICS FOR SVM ON THE TEST SET

	Predicted Black	Predicted Draw	Predicted White
Actual Black	437	0	372
Actual Draw	35	0	52
Actual White	237	0	639

TABLE X
CONFUSION MATRIX FOR SVM ON THE TEST SET

Without detailed information about how the game played out, like move sequences or board states, it's tough to find patterns that point to draws.

Draws in chess usually happen in specific situations during the game. For example, balanced endgames or positions with opposite-colored bishops are common signs of a draw. These kinds of details are hard to capture with high-level features like ratings or move counts. A more detailed analysis using move sequences or board positions might help models recognize these patterns and predict draws better. But doing this would require a deeper and more computationally heavy approach, similar to how advanced chess engines like Stockfish analyze games.

The struggle to predict draws isn't unique to our project. Other studies that used similar high-level approaches, without diving into move-by-move data or deep learning, reported similar results. For example, Drezewski and Wator [1] achieved 69% accuracy with a deep learning approach using sequential data, but high-level models like ours typically hit a limit around 60–61% accuracy. This shows that using only high-level features has its limits, and better performance would likely need more detailed data.

Even with these challenges, our models managed to achieve 61% accuracy. This shows that they're pretty good at predicting white and black wins, which are easier to distinguish in the data. Since draws are rare and share features with the other outcomes, it's expected that their F1 scores are low. For our high-level approach, accuracy remains the best way to evaluate how well the models performed.

That 61% accuracy also suggests we've hit the ceiling for what our current features can achieve. To go beyond this, we'd need to use more detailed features like move sequences or board positions. Our approach was designed to be simple and easy to interpret, focusing on general game characteristics. While this has its limitations, it also shows the trade-offs between keeping things simple and aiming for better performance.

VIII. Conclusion

In summary, predicting draws is tough because high-level features just don't give enough information to set them apart in chess game analysis. Future work could look into using move sequences or board positions to get a better understanding of what leads to draws, which might help improve accuracy and balance across classes. Similar studies that used high-level approaches got results close to ours, showing that our findings make sense, but going further would need more detailed data and advanced methods.

References

[1] R. Drężewski and G. Wator, "Chess as sequential data in a chess match outcome prediction using deep learning with various chessboard representations," *Procedia Computer Science*, vol. 192, pp. 1760–1769, 2021.

[2] K. Reddy, B. Kumar, N. P. Kumar, T. Parasuraman, C. Balasubramanian, and R. Ramakrishnan, "Chess match outcome prediction via sequential data analysis with deep learning," in *2023 International Conference on Sustainable Communication Networks and Application (ICSCNA)*, 2023, pp. 1442–1447.

[3] M. M. Masud, A. Al-Shehhi, E. Al-Shamsi, S. Al-Hassani, A. Al-Hamoudi, and L. Khan, "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.

[4] H. A. Rosales Pulido, "Predicting the outcome of a chess game by statistical and machine learning techniques," Master's thesis, Universitat Politècnica de Catalunya, 2016.

[5] P. Lehana, S. Kulshrestha, N. Thakur, and P. Asthana, "Statistical analysis on result prediction in chess," *International Journal of Information Engineering and Electronic Business*, vol. 11, no. 4, p. 25, 2018.

[6] A. Mujagić, A. Mujagić, and D. Mehanović, "Predictive analysis of chess player performance: An analysis of factors influencing competitive success using machine learning techniques," in *International Symposium on Innovative and Interdisciplinary Applications of Advanced Technologies*. Springer, 2024, pp. 392–408.

[7] F. Thabtah, A. J. Padmavathy, and A. Pritchard, "Chess results analysis using elo measure with machine learning," *Journal of Information & Knowledge Management*, vol. 19, no. 02, p. 2050006, 2020.

[8] S. DeCredico, "Using machine learning algorithms to predict outcomes of chess games using player data," Master's thesis, Rochester Institute of Technology, 2024.

[9] T. Tijhuis, "Predicting chess rating based on a single game," Ph.D. dissertation, TILBURG UNIVERSITY.

[10] T. L. Saaty and L. G. Vargas, "Hierarchical analysis of behavior in competition: Prediction in chess," *Behavioral science*, vol. 25, no. 3, pp. 180–191, 1980.