Proceedings of CONF-SEML 2026 Symposium: Computational Analysis and Modeling in Complex Intelligent Syste (CONF-SEML 2026)
DOI: 10.54254/2755-2721/2026.GU34787

Chess Game Outcome Prediction Based on Machine Learning Methods

Xiaonan Zeng

Electronic and Computer Engineering, Shenzhen MSU-BIT University, Shenzhen, China
outlook_498E94FBD5E9E145@outlook.com

Abstract. As a classic strategy confrontation board game, chess features both complex spatial layout characteristics and long-term sequential dependency, making it an important research scenario in the fields of artificial intelligence and machine learning. This study is based on 3,196 endgame match data and 35 board state features, constructing a multi-model comparison framework including decision trees, logistic regression, Long Short-Term Memory (LSTM), and Convolutional Neural Networks (CNN), to verify the performance differences of different algorithms in the task of predicting the outcome of chess moves. Through feature encoding, dataset division, and multi-model training and evaluation, the strengths and weaknesses of each model in terms of accuracy, generalization ability, and inference efficiency are analyzed. Preliminary experiments show that logistic regression as a baseline model can effectively learn simple chess game patterns. Subsequently, the introduction of LSTM and CNN revealed that the dataset is relatively weak in sequence modeling and spatial feature extraction, resulting in low accuracy. Further improvement in prediction accuracy is needed. This research provides a reference basis for the model selection of the intelligent analysis system for board games, and also offers practical ideas for the prediction task that integrates temporal and spatial features.

Keywords: Decision tree, logistic regression, Long Short-Term Memory, Convolutional Neural Networks, Chess move prediction

1. Introduction

As a typical representative of human intellectual games, the decision-making process in chess involves complex spatial layout analysis and long-term strategic reasoning. In recent years, with the breakthroughs in deep learning technology, chess AI systems like AlphaZero have achieved performance that surpasses the top human level in the field of game decision-making [1]. However, such systems mostly focus on generating "optimal decisions", while their research on the task of "predicting the outcomes of real players' actions" is relatively limited. The prediction of real player actions and outcomes not only helps chess players conduct tactical analysis and game review, but can also be applied in scenarios such as intelligent chess-playing systems and novice teaching assistance, and holds significant application value. The prediction of the outcome of chess endgames is a typical sequential decision-making and classification problem, which holds significant value for the study of intelligent game algorithms.

© 2026 The Authors. This is an open access article distributed under the terms of the Creative Commons Attribution License 4.0 (https://creativecommons.org/licenses/by/4.0/).

DOI: 10.54254/2755-2721/2026.GU34787

Early research on chess move prediction mostly relied on expert systems and heuristic search, such as the classic alpha-beta pruning algorithm, which achieved the selection of optimal chess moves through evaluation functions and search strategies [2-4]. The above ways are only appropriate for definite games and cannot be applied to the problem of predicting how an irrational person will behave. With the progress of machine learning technology, some scholars have started to apply supervised learning to the problem of predicting moves in chess. Panchal and others employed deep neural networks for chess move prediction and showed the advantages of deep learning models in automatic feature extraction [5]. In addition, sequence models such as LSTMs are employed to capture the long-term time dependencies of the game, and CNN is used to extract spatial structure features of the chessboard [6, 7]. However, the previous studies also have some defects. For example, most studies focus on generating the optimal moves but pay little attention to predicting the actions of real players; many studies use very different data formats and feature encoding methods, and there is no unified comparison system for them; and systematic evaluation of the generalisation ability and feature fusion mechanism of the models is relatively scarce. Based on the above reasons, a single comparative experiment will be conducted to determine how well various machine learning models perform in predicting chess moves systematically. A combined comparative framework for decision trees, logistic regression, LSTM and CNN models has been constructed to evaluate the different strengths of these models in addressing spatial and temporal features of chess games, providing empirical support for the selection of models in the intelligent analysis system for chess games [6-8]. Theoretically, this paper conducts a search for the optimal direction of time-series prediction and spatial feature extraction; at the same time, the results of this research can be directly applied in practice to recommend chess moves and review game replays.

2. Dataset

2.1. Data sources

The study takes "Chess (King-Rook vs. King-Pawn)" dataset publicly available from the UCI Machine Learning Repository. It is a multi-variable categorical game dataset with a total of 3196 instances [9]. As each line of data is separated by a comma, and the last field is either the character "won" or "nowin", it can be divided into two types. The total number of data in the "won" category is 1669, and these indicate that the white side (with king and rook) has won; the total number of data in the "nowin" category is 1527, and these indicate that the white side has not won, meaning either the black side has won or the game has ended in a draw. Each item is a full end game, and there are a total of 35 characteristics of the end game, including board position and piece status. There are no missing values, and thus the data is complete and usable.

2.2. Feature encoding scheme

The original data is in character form (f/t/n for false/true/neutral), and it needs to be converted into numerical features for the model to accept them. Define the feature encoding mapping function:

$$\phi(c) = \begin{cases} 0, & c = f \\ 1, & c = t \\ 2, & c = n \end{cases} \tag{1}$$

DOI: 10.54254/2755-2721/2026.GU34787

Apply the above mapping to all 35 feature columns, convert the original character-based chess game features into a numerical vector x∈ {0, 1, 2}35, and the column "won" is a binary classification problem. Set the LabelEncoding function.

$$\psi(s) = \begin{cases} 1, & s = won \\ 0, & s = nowin \end{cases} \tag{2}$$

The result of the game is converted into a binary classification label y ∈ {0, 1}, where 1 indicates that the King-rook has won, and 0 indicates that it has not won.

A training set and a test set were divided from the encoded data in an 8:2 ratio. A fixed random seed of random_state=42 was set to ensure the reproducibility of the experiment, and the training set and test set, respectively, contained 2556 and 639 game data.

3. Methodology

The library for decision trees is sklearn.tree.DecisionTreeClassifier. The structure is a binary tree. It splits based on features and by default uses the CART algorithm (Classification and Regression Tree) [8]. Splitting criterion: gini (Gini coefficient), random_state = 42 (fixed random seed), the rest are default values (such as max_depth = None, the tree will grow completely until the purity of the leaf nodes is 0 or the number of samples is less than min_samples_split).The loss function (optimization objective) is to minimize the node impurity (Gini or information gain), and the accuracy calculation directly calls the accuracy_score function.

The library for Logistic Regression is sklearn.linear_model.LogisticRegression. Its structure is a linear binary classification model, and the output is mapped to probabilities through the Sigmoid function [10].

$$P(y = 1x) = \frac{1}{1+e^{-(\omega^T x+b)}} \tag{3}$$

max_iter=1000 (maximum number of iterations), random_state=42, and the rest are set to their default values (solver='lbfgs', penalty='12')

Loss function: Logarithmic loss (Cross entropy):

$$J(\omega) = -\frac{1}{N}\sum_{i=1}^{N}[y_i \log(p_i) + (1-y_i)\log(1-p_i)] \tag{4}$$

The framework of LSTM is TensorFlow/Keras, and its structure is a Sequential model [6]. The LSTM model adopts a sequential model structure. The specific layers and parameter settings are shown in Table 1.

Table 1. Illustration of LSTM's specific layers and parameter settings

Layer	Parameter	Explanation
LSTM (64)	Units=64	Input shape (12, 3), which means 12 time steps, with 3 features per step (rearrangement of the original 36 features)
Dense (32, activation='relu')	32 neurons,ReLU activation	Fully connected layer
Dense (1, activation='sigmoid')	One neuron, Sigmoid	Output binary classification probability

DOI: 10.54254/2755-2721/2026.GU34787

Training parameters: Optimizer Adam (learning rate = 0.001), loss function is binary_crossentropy (the same as logistic regression), metric is accuracy rate, epochs = 10, batch size = 32, validation split = 0.1.The accuracy rate is obtained by using model.evaluate to return the accuracy on the test set after the training is completed.

The framework of CNN is TensorFlow/Keras, and the structure employs the Sequential model [7]. The CNN model also adopts a sequential model structure, consisting of two convolutional pooling modules and a fully connected classification layer. The detailed hierarchy and parameter settings are shown in Table 2.

Table 2. Detailed structure and parameter settings illustration of CNN

Layer	Parameter	Explanation
Conv1D (32, kernel_size=3, activation='relu')	32 convolutional kernels, each with a size of 3	Input shape: (36, 1) (The 36 features are regarded as a 1D sequence, with a single channel)
MaxPooling1D (pool_size=2)	Pooling window 2	Downsampling
Conv1D (64, kernel_size=3, activation='relu')	64 convolutional kernels, each with a size of 3	Second layer convolution
MaxPooling1D (pool_size=2)	Pooling window 2	Downsampling
Flatten ()	Flatten	Convert the feature map into a one-dimensional vector
Dense (64, activation='relu')	64 neurons	Fully connected layer
Dense (1, activation='sigmoid')	Output layer	Second classification

The training parameters are exactly the same as those of LSTM (optimizer, loss function, epochs, batch size, validation_split), and the accuracy is also obtained through evaluate.

4. Experiments and results

Table 3. Comparison of experimental results of the four models

Model	Accuracy of the test set	Precision rateExplanation	Recall rate	F1
Decision Tree (Benchmark)	99.37%	99.43%	99.43%	99.43%
Logistic regression	95.93%	94.97%g	97.70%	96.32%
LSTM	85.45%	89.47%	83.05%	86.14%
CNN	94.05%	90.79%	99.14%	94.78%

The comparison of the accuracy, precision, recall and F1 value of each model on the test set is shown in Table 3. The decision tree fits this dataset almost perfectly because the data features are highly correlated with the target variable, and the decision boundaries are clear. Taking the decision tree model as the initial benchmark, the accuracy rate on the test set was 0.9937. The subsequent effects of LSTM and CNN were significantly worse than those of logistic regression and decision trees. This was because the design intention of LSTM is to handle data with real-time sequential dependencies (such as text, time series), while in this experiment, 35 independent features were forcibly regarded as "time steps", but there was no sequential dependency among these features. The gating mechanism of LSTM attempts to learn non-existent temporal patterns, but by forcibly

DOI: 10.54254/2755-2721/2026.GU34787

reshaping it into 12×3, the structural information is lost. As a result, its temporal modeling ability fails to show its advantages, and it can only memorize the noise in the training set. In this tabular classification task, it performs poorly and increases the model complexity, leading to overfitting.

CNN is designed to extract local spatial correlations (such as adjacent pixels in an image). After reshaping the table data into a 5×7 grid, the originally non-adjacent features were forcibly assigned to adjacent positions. The convolution operation, when performed on the randomly arranged grid, was unable to capture meaningful local patterns and instead amplified the noise. However, it has a high recall rate, indicating that it almost never misses any positive cases ("winning" situations), but the low precision rate also suggests that it may mistakenly classify some negative cases as positive ones.

Therefore, the model selection must be in line with the actual structure of the data. For tabular data that does not contain temporal or spatial dependencies, traditional linear models or tree models tend to be more effective and stable.

5. Conclusion

This study was based on the endgame dataset of chess king-rook versus king-pawn battles, and conducted experiments for comparing multiple models. The decision tree model performed exceptionally well on this dataset, verifying the learnability of the dataset; logistic regression, as the benchmark model, has the advantage of efficient reasoning and is suitable for rapid deployment; LSTM and CNN models have greater potential in complex feature modeling, and further parameter tuning and optimization are required. For instance, LSTM is limited by the situation of limited data volume. If the number of units is set too high, the risk of overfitting is high, resulting in the lowest percentage in all four dimensions; the extremely high recall rate and relatively low precision rate of CNN can also be further balanced by adjusting the classification threshold.

The current experimental dataset is relatively small, which may limit the sufficient training of deep learning models and pose the risk of overfitting. The impact of the feature encoding method on the model performance has not been fully quantified, and integration learning or attention mechanisms can be introduced in the future. Moreover, the interpretability of the four types of models varies significantly, and a trade-off needs to be made based on specific scenarios. In the future, it is necessary to expand to more types of endgames and collect larger-scale data. If the data has real-time sequence characteristics (such as complete game records), then the potential of LSTM/Transformer needs to be re-evaluated. It is also possible to explore encoding domain knowledge of the chessboard (such as the spatial relationship of pieces) as effective features, and then combine with CNN, which may be helpful in improving performance.

References

[1] Sekar, E. G. H., & Jin, R. (2025, May). Human-aligned chess AI: A multitask transformer for humanlike decision-making. In 2025 IEEE Conference on Artificial Intelligence (CAI) (pp. 1230–1234). IEEE.

[2] Madake, J., Deotale, C., Charde, G., & Bhatlawande, S. (2023). CHESS AI: Machine learning and Minimax based chess engine. In 2023 International Conference for Advancement in Technology (ICONAT), Goa, India.

[3] Karia, P., Jain, V., Shah, M., & Rane, S. (2022). Digitization of chess board and prediction of next move. In 2022 IEEE 7th International Conference for Convergence in Technology (I2CT), Pune, India.

[4] Parmar, H., Panthy, S. Y. R., & Murari, U. K. (2025). A hybrid Minimax-MCTS chess engine enhanced by RLHF using Stockfish and AlphaZero. In IEEE International Conference on Advances in Computing Research on Science Engineering and Technology (ACROSET), Indore, India (pp. 1–8).

[5] Panchal, H., et al. (2021). Chess moves prediction using deep learning neural networks. In 2021 International Conference on Advances in Computing and Communications (ICACC), Kochi, Kakkanad, India (pp. 1–6).

DOI: 10.54254/2755-2721/2026.GU34787

[6] Zhu, J. (2024). Automatic digitization of chess scoresheets with RNNs. In 2024 IEEE MIT Undergraduate Research Technology Conference (URTC), Cambridge, MA, USA (pp. 1–5).
[7] Agarwal, S., Dash, S., Saini, A., Singh, N. K., Kumar, A., & Diwakar, M. (2024). NIRNAY: An AI chess engine based on convolutional neural network, Negamax and move ordering. In 2024 11th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions) (ICRITO), Noida, India.
[8] Yamada, H., et al. (2023). A method for estimating online chess game player ratings with decision tree. In 2023 IEEE International Conference on Big Data and Smart Computing (BigComp) (pp. 320–321). IEEE.
[9] UCI Machine Learning Repository. (1989). Chess (King-Rook vs. King-Pawn) data set. https://archive.ics.uci.edu/ml/datasets/Chess+(King-Rook+vs.+King-Pawn)
[10] Reyes, M. J. R.-a. D., Dicreto, E., Santos, E. G. D., Limbag, D. F. P., & Sampedro, G. A. (2025). EloMetrics: Advanced outcome prediction for chess matches with Elo ratings and logistic regression. In 2025 International Conference on Electronics, Information, and Communication (ICEIC), Osaka, Japan (pp. 1–4).