Predicting Chess Winners by the Opening Strategy

Ty Turner

May 5th, 2025

Introduction

Chess is a timeless game that has transcended empires, cultures, and even the boundaries between the physical and virtual worlds. For some, it is a casual pastime; for others, it can become a professional pursuit. A typical classical chess game lasts approximately 40 moves (or 79 half-moves) and can range from just a few minutes to over an hour, depending on the players' skill levels and strategic approaches. For newcomers to chess, playing the game often takes precedence over analyzing it, especially for those with limited time. This research aims to predict the winner of a chess game based on the opening strategy, specifically within the first 15 moves. To accomplish this, we collected a random sample of 10,000 games played on Lichess during July 2016.

Literature Review

DeCredico [1] proposed a similar model that incorporated each player's win rate, draw rate, rating, and the win/draw rate of the opening to predict the outcome of a chess game. The study concluded that including player ratings significantly improved model accuracy beyond random guessing, while opening strategies had minimal impact on predictive performance. Other researchers have explored more granular features derived from the specific moves made during games. Kalgara [3], for instance, engineered features such as "Queen Lifetime," "Number of Central Pawns," and "Castling," which led to a 20–30% improvement over random guessing. Additionally, Oshri [10] and Gupta [4] employed Convolutional Neural Networks (CNNs) to assign value to individual chess squares and pieces, enabling mid-game evaluation of optimal moves based on board state.

Mcllroy-Young and collaborators conducted research on detecting decision-making styles using behavioral stylometry [8]. Their model was capable of identifying whether a player was high-ranking based on data from entire games or from partial games starting after a specific number of moves. Levene and Fenner [7] employed temporal difference learning and other machine learning techniques to enhance their model's accuracy—an approach commonly used in prominent chess engines such as Stockfish and AlphaZero. Degni [5] also discusses the influence of such engines, noting that their dominance has contributed to a perceived decline in creativity within the game. According to Degni, the first 15–20 moves of many games often follow established, high-performing openings previously analyzed by these engines.

Exploratory Data Analysis

Our dataset was provided in the standard Portable Game Notation (PGN) format, a plain-text format commonly used to record chess games in a human-readable way. In addition to PGN, the dataset included several statistical features not originally part of the format. For this study, we primarily focused on a subset of features that could help us build an efficient predictive model for determining chess game outcomes. The key features used were: **Event**, **ECO** (Encyclopedia of Chess Openings code), **Opening**, **WhiteElo**, **BlackElo**, **TimeControl**, and **AN** (Algebraic Notation). Our target variable was **Result**, where a value of 0 indicates a win for the Black player and a value of 1 indicates a win for the White player. We also engineered additional features to capture variance and potential correlations not explicitly available in the original data. One such feature was **EloDiff**, which represents the difference in Elo ratings between the two players—a positive value indicates White had the higher rating, while a negative value indicates Black did. Another feature, **AN_cluster**, grouped games by similarity in their opening sequences based on algebraic notation, allowing the model to capture relationships between openings and outcomes.

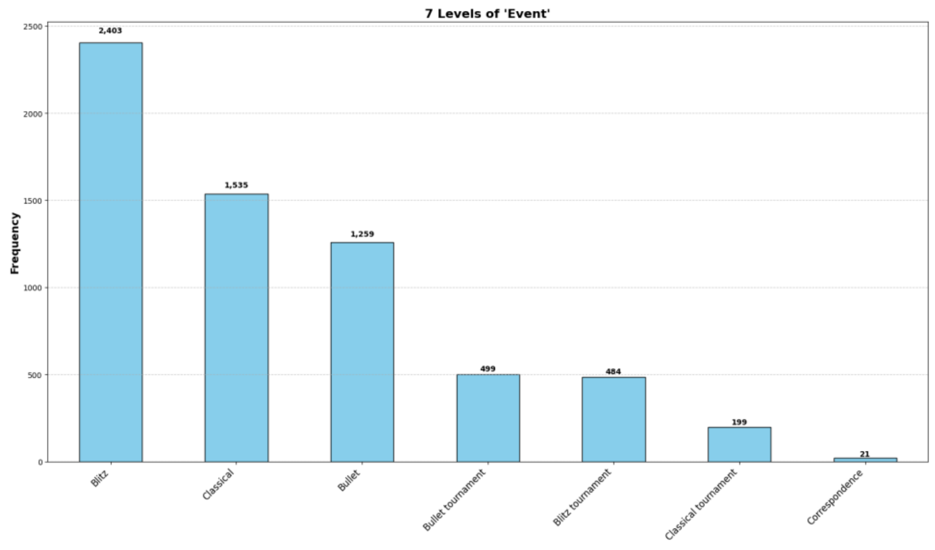


Figure 1 – Event Feature Histogram

We begin our analysis with the categorical features. The **Event** feature specifies the type of game being played. The most common types observed in the dataset are **Blitz**, **Classical**, and **Bullet**. Blitz and Bullet refer to fast-paced formats that are ideal for players with limited time, while Classical denotes the standard format of chess, typically played at a slower pace with longer time controls. The next two categorical features pertain to the game's opening: **ECO** (short for *Encyclopedia of Chess Openings*) and **Opening**. ECO represents a standardized code that categorizes chess openings, while the Opening field provides the corresponding descriptive name.

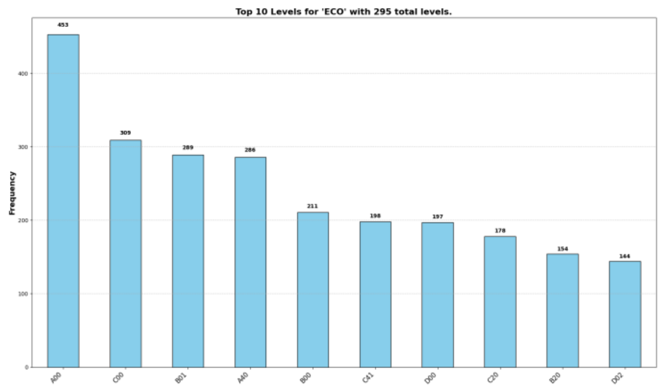


Figure 2 – ECO Feature Histogram

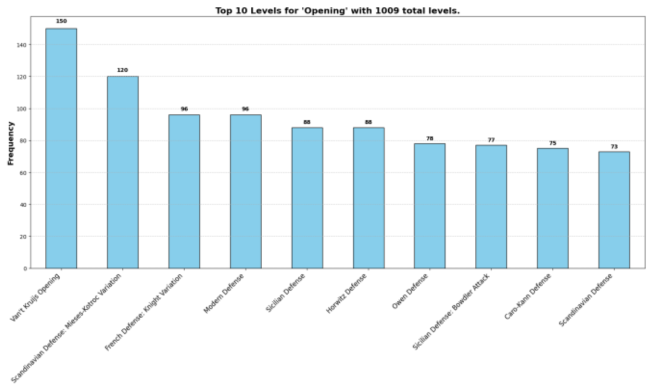


Figure 3 – Opening Feature Histogram

The **ECO** feature contains 500 possible values, corresponding to a standardized three-character code used to classify chess openings. The code begins with a letter from **A to E**, followed by a two-digit number ranging from **00 to 99**. In contrast, the **Opening** feature provides the name of the specific opening or sequence of opening strategies used in a given game. Due to the wide variety of possible move sequences, the number of distinct Opening values is considerably larger and effectively unbounded.

Next, we consider our quantitative features. **WhiteElo** and **BlackElo** represent the matchmaking ratings of the White and Black players, respectively. **TimeControl** denotes the total duration of the game in seconds, which reflects the format and pacing of each match. Finally, **AN_cluster** is a feature we engineered to group similar games based on the first 15 moves, as recorded in the original **AN** (Algebraic Notation) field.

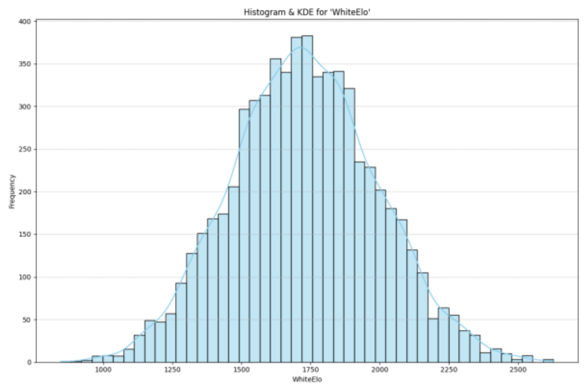


Figure 4.1 – WhiteElo Histogram & KDE Plot

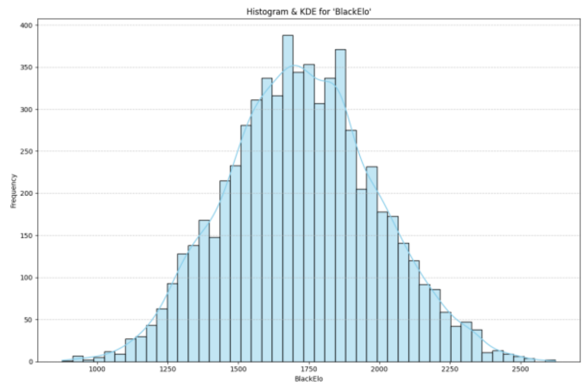


Figure 4.2 – BlackElo Histogram & KDE Plot

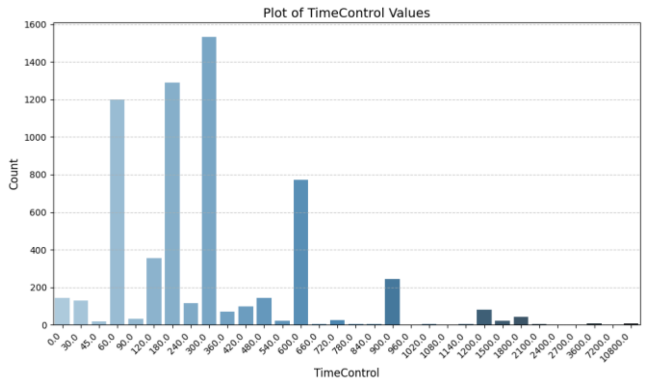


Figure 5.1 – TimeControl Histogram Plot

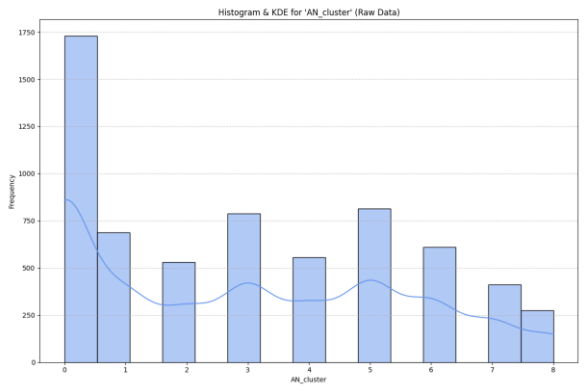


Figure 5.2 – AN_cluster Histogram & KDE Plot

We also have a table to give relevant stats on our quantitative variables.

Six-Number Summary	WhiteElo [Figure 4.1]	BlackElo [Figure 4.2]	TimeControl [Figure 5.1]	AN_cluster [Figure 5.2]
Min	846.00	877.00	0.00	0.00
Q1	1547.75	1544.75	120.00	0.00
Q2	1724.50	1722.00	240.00	3.00
Q3	1901.00	1896.00	300.00	5.00
Max	2628.00	2623.00	10800.00	8.00
Mean	1727.80	1725.90	326.28	2.99

Table 1 – Quantitative Features Statistics

CatBoost

CatBoost is a gradient boosting algorithm, similar to **XGBoost**, that is particularly effective at handling categorical variables without requiring preprocessing techniques such as one-hot encoding, which can contribute to the curse of dimensionality. CatBoost can automatically process categorical features such as **ECO** and **Opening**, allowing us to concentrate our feature engineering efforts on the quantitative variables.

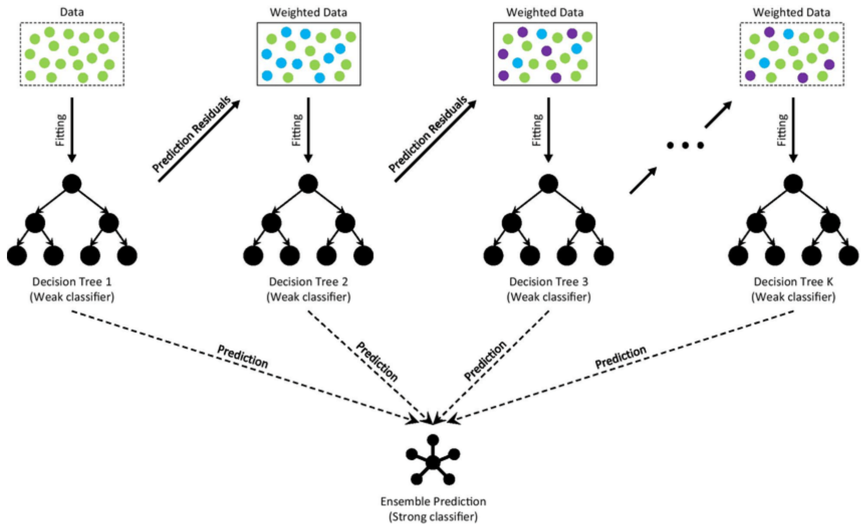


Figure 6 - Diagram of Gradient Boosting Procedure. (Al-Abdaly, 2024)

CatBoost allows you to specify the number of iterations used to build decision trees, as illustrated in Figure 6. Each tree is trained on the residuals of the previous one, gradually improving the model. After completing the specified number of iterations, the algorithm combines the trees—typically by averaging—to produce a strong ensemble prediction. Given CatBoost's ability to handle categorical features effectively while avoiding overfitting and minimizing training time, it is an excellent choice for training our model.

K-Means Clustering

The primary feature engineering challenge in our dataset was developing a method to allow the model to effectively compute and interpret the **AN** column. Since our research goal was to predict the winner by the 15th move, we limited our analysis to the first 15 moves, discarding any moves beyond that point. We then applied a distance metric to compare each game in the dataset, generating a distance matrix. The distance metric compares two games by evaluating their **AN** values and determining where the divergence occurs in the opening sequence. Below is an example of comparing the opening plays of two separate games.

$$AN1 = 1.e4, e5\ 2.nf3, nc6$$

$$AN2 = 1.e4, e5\ 2.nf3, d6$$

The divergence in the two example games above would determine that the divergence happens at **n = 4**. Below would be the resulting distance metric calculation.

$$distance(AN1, AN2) = \frac{1}{n\text{^}2} = \frac{1}{16}$$

The closer the games were in terms of opening moves, the smaller the distance metric value, approaching 0. Conversely, the more divergent the games were in their opening moves, the distance metric would approach 1. Once the distance matrix is populated, we then apply a clustering algorithm to group similar distance metrics, assigning each game to an appropriate cluster in the **AN_cluster** column.

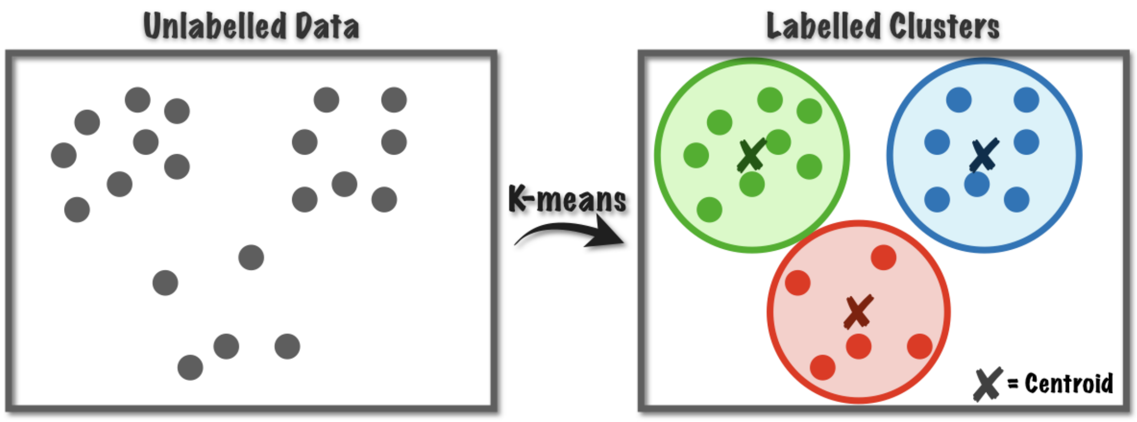


Figure 7 - K-Means Clustering Example

We selected the **K-Means Clustering** algorithm for our clustering analysis, as it is well-suited for categorizing clusters based on similarity. To determine the optimal number of clusters (**k**), we will use both the **Elbow Method** and the **Silhouette Method** to guide our decision. Figure 8 and Figure 9 illustrate the efficiency of different **k**-values, highlighting the point at which the efficiency drops for higher **k**-values. The Elbow Method identifies the optimal number of clusters by evaluating the within-cluster sum of squares for each **k**-value. This is done using the following equation:

$$WCSS = \sum_{i=1}^{k} \sum_{x \in C_i} ||x - \mu_i||^2$$

Where k is the number of clusters, C_i represents the i-th cluster, and μ_i is the sample mean of the i-th cluster. We plot the WCSS values for each k and identify the "elbow point" on the graph, which helps us determine the optimal k-value.

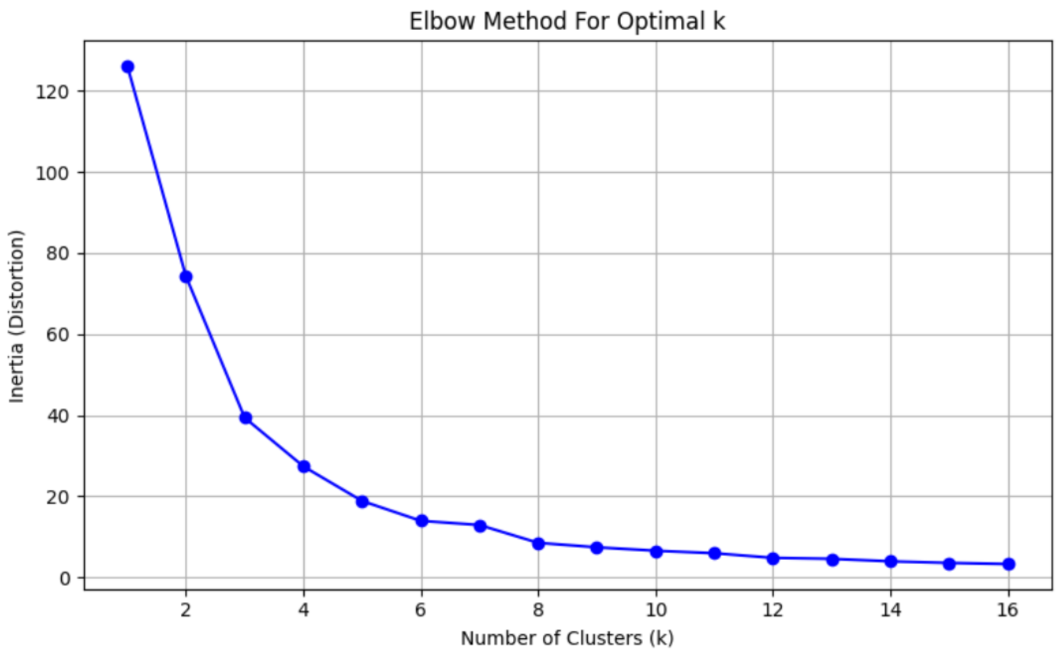


Figure 8 – Elbow Method Graph

Analyzing the Elbow Method in Figure 8, we observe that the optimal **k**-values lie between 4 and 9, as indicated by the curved elbow shape rather than a sharp, rigid drop. To further validate our choice of the optimal **k**, we then apply the **Silhouette Method**. This method evaluates the quality of clustering using the following equation:

$$s(i) = \frac{b(i) - a(i)}{\max\{a(i), b(i)\}}$$

Where a(i) is the average distance between point i and all other points in the same cluster, and b(i) is the minimum distance between point i and all points in the nearest cluster to which it does not belong. For each **k**-value, the silhouette score is averaged over all points, and the results are plotted in a graph, as shown in Figure 9.

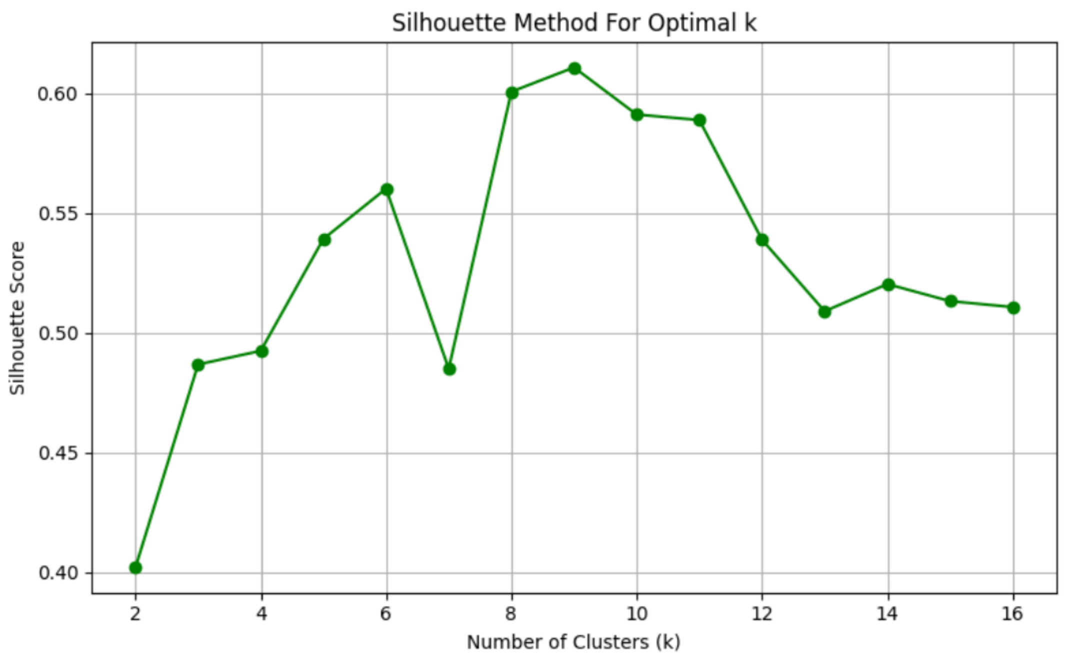


Figure 9 – Silhouette Method Graph

As shown above, we focus on **k**-values 8 and 9, as they fall within the optimal range identified by the Elbow Method in Figure 8. Additionally, these values exhibit silhouette scores closest to 1, with **k=8** yielding a score of 0.6 and **k=9** yielding a score of 0.62.

Model Optimization

As illustrated in Figure 8 and Figure 9, we optimized the **k**-value in K-Means Clustering to **k=8**, as shown in Figure 5.2. For the CatBoost model, we focused on hyperparameter tuning, selecting the following values: **iterations = 5,000**, **depth = 8**, and **learning rate = 0.001**. The choice of 5,000 iterations was based on empirical testing, as values beyond this threshold did not significantly improve the model's performance and only increased computational time. We selected a depth of 8 since the model had 8 features to learn from, aligning the model's depth with the number of input features.

Before performing feature engineering, we split our data into two groups: 80% for training and 20% for validation. Afterward, we applied the feature engineering techniques described in the Exploratory Data Analysis and K-Means Clustering sections to the training dataset. We further divided the training dataset into two subsets: 80% for training the model and 20% for testing. We then trained the CatBoost model, and in the following section, we will present and discuss the results.

Model Results

In Table 2, we present the results of our model on the testing dataset.

Classification Report	Accuracy Score	F1-Score	Support
Black wins - 0	0.630	0.620	776
White wins - 1	0.650	0.650	824

Table 2 - Classification Report Table for Testing Dataset

The model achieves an accuracy of approximately 64% in predicting the correct winner. Additionally, the F1-Score is slightly lower at 0.63, indicating a balanced trade-off between precision and recall.

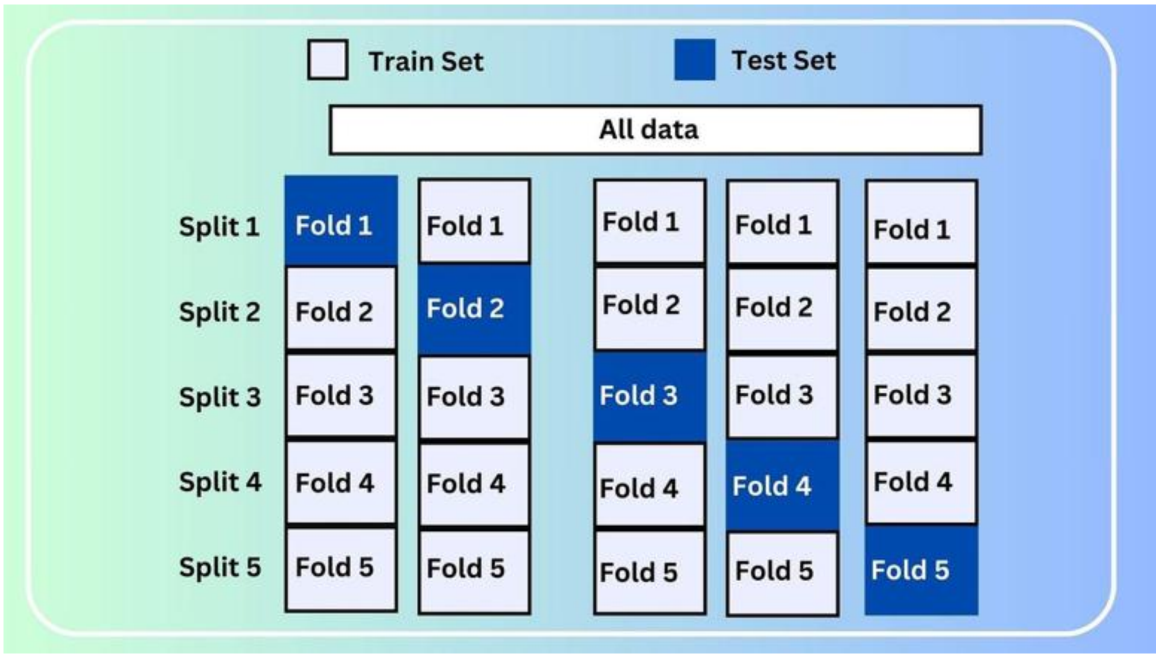


Figure 10 – K-folds Cross Validation Example

For our cross-validation, we employed the K-Folds cross-validation method on both our testing and holdout datasets. K-Folds divides the data into **k** subsets (folds), and for each fold, the model is trained on all but one fold and tested on the remaining fold. This process is repeated for each fold, and the scores are averaged to compute the overall cross-validation score. We applied K-Folds cross-validation to our model and dataset, and the resulting scores are shown below.

Metric	Value
Cross-Validation Accuracy	0.634687
Test Accuracy	0.631500
F1-Score	0.631380
AUC-ROC	0.695860

Table 3 - K-Folds Cross Validation Table

As shown in Table 3, the cross-validation accuracy aligns closely with the accuracy on the test dataset, demonstrating that our model performs equally well on the holdout set as it did on the test set. This indicates that the model generalizes well to unseen data. In Figure 11, we present the ROC (Receiver Operating Characteristic) curve, which corresponds to the values presented in Table 3.

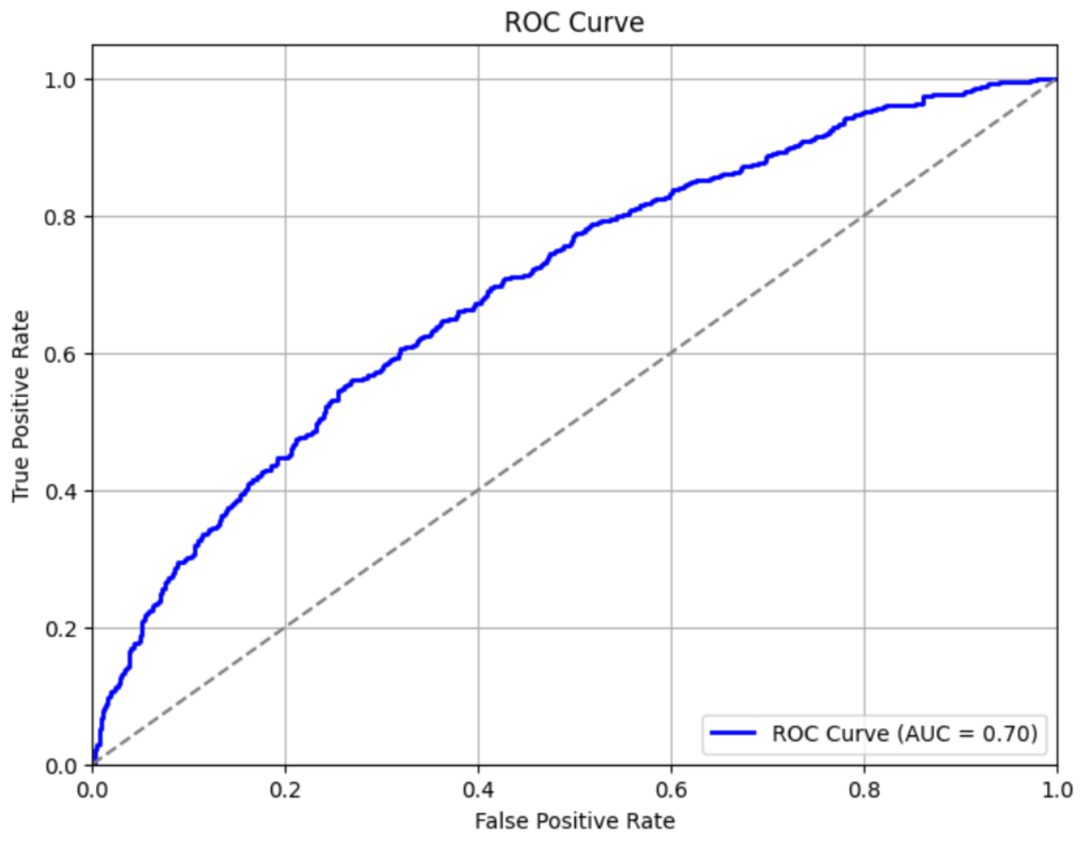


Figure 11 – AUC-ROC Curve Graph

With an ROC curve value of 0.7, our model demonstrates predictive power that is significantly better than random guessing, though there is still room for improvement. Additionally, as shown in Table 2, the model predicted 'White' to win more accurately than 'Black', which is consistent with the findings from the ROC curve.

In the following figures, we present our model's Shapley values, which indicate the importance of each feature in our model's predictions. In Figure 12, the Shapley beeswarm plot shows the value range for each feature. Negative values indicate lower feature values, while positive values correspond to higher feature values. The color scale on the right-hand side of the graph highlights the influence of each feature on the model's predictions: red indicates a high influence, while blue represents a lower influence.

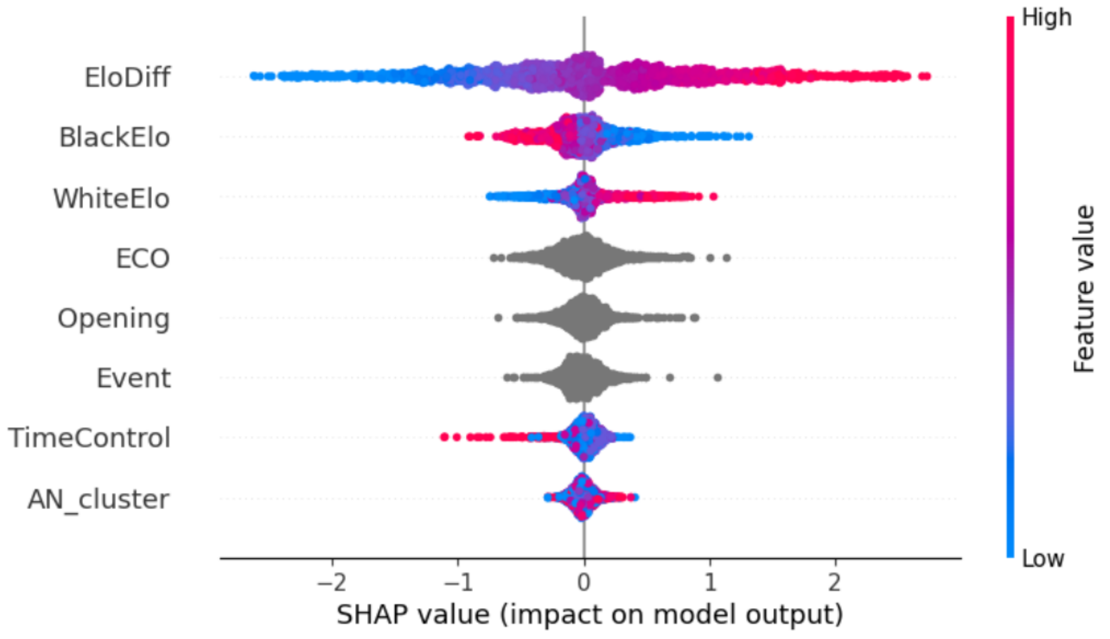


Figure 12 – Shapley Beeswarm Plot

As shown in the beeswarm plot above, it is logical that a higher 'EloDiff' value plays a significant role in predicting the winner, as it reflects the discrepancy between the players' matchmaking ratings. Our model performs better when predicting outcomes involving lower-rated 'Black' players and higher-rated 'White' players, which aligns with the high importance of 'EloDiff' as a predictor. In Figure 13, we also present a bar graph illustrating the model's feature importance, highlighting the relative significance of each feature in the prediction process.

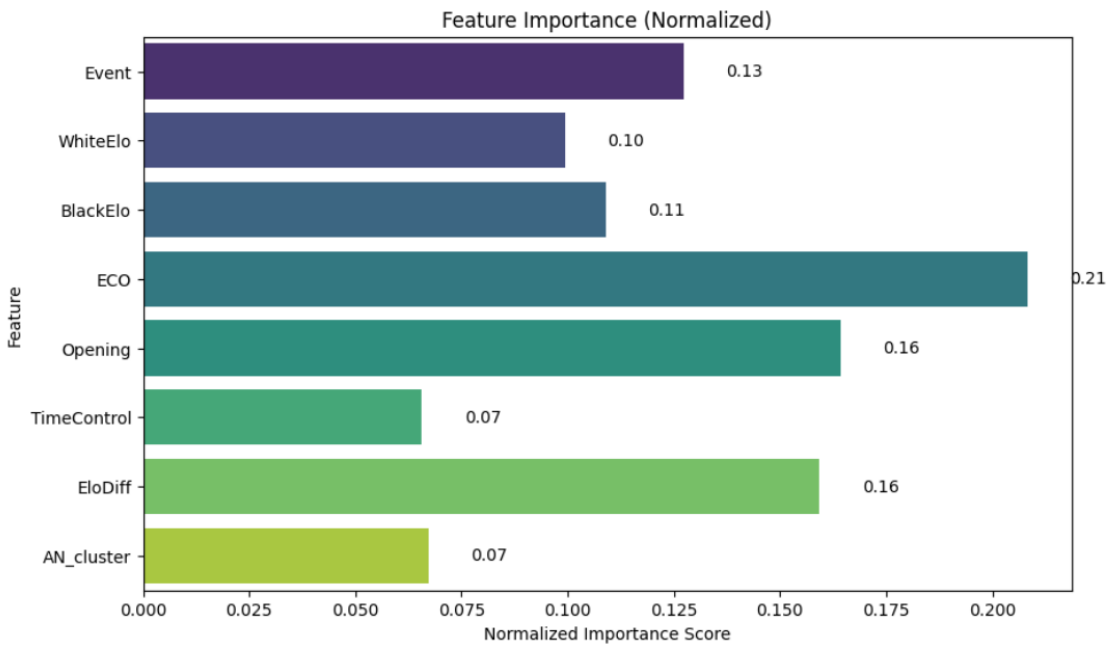
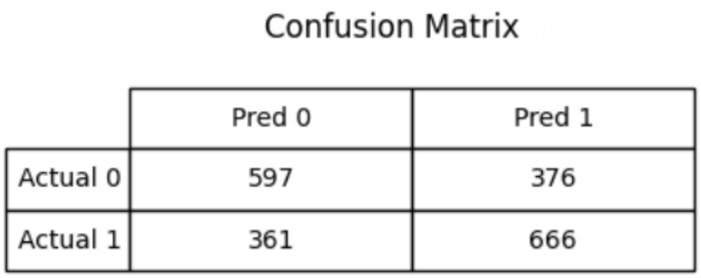


Figure 13 – Shapley Bar Plot

The bar graph in Figure 13 provides additional insight beyond what the beeswarm plot offers. While the beeswarm plot excels at displaying quantitative values, it struggles to effectively represent the importance of categorical features. In contrast, Figure 13 clearly highlights the relative importance of each feature, while Figure 12 emphasizes the significance of specific feature values. This analysis demonstrates that our model prioritizes categorical features but still places significant importance on 'EloDiff' as a valuable predictor. To further assess our model's predictive performance, we present a confusion matrix that illustrates how well the model's predictions align with actual outcomes on the holdout set.



Confusion Matrix

	Pred 0	Pred 1
Actual 0	597	376
Actual 1	361	666

Figure 14 – Confusion Matrix for Holdout Set

The confusion matrix presented above shows that our model performed well in predicting both 'Black' and 'White' wins. However, it more frequently predicted 'White' as the winner when 'Black' actually won, rather than predicting the opposite scenario.

Conclusions and Future Work

Our model demonstrates strong performance in predicting the winner of a chess game after the 15th round. However, it is not flawless and can be further refined to improve its predictive capabilities. One notable observation is that our model places greater weight on categorical features than on quantitative ones, as is typical with algorithms like CatBoost.

Future work could involve exploring methods to transform the 'ECO' and 'Opening' columns into quantitative variables. This would allow us to apply different algorithms that could better capture the nuances of these features and potentially improve the model's performance. By doing so, we might see increased importance given to features such as 'EloDiff' and 'AN_cluster,' which are currently underutilized with CatBoost.

Additionally, we believe that the distance metric played a crucial role in our model, as it enabled the grouping of similar game opening strategies, creating relationships between rows that may not have been apparent without this transformation.

References

[1] DeCredico, S. (2024). *Using Machine Learning Algorithms to Predict Outcomes of Chess Games using Player Data*. *Rochester Institute of Technology*. Retrieved from https://repository.rit.edu/cgi/viewcontent.cgi?article=13036&context=theses

[2] Arsalan, M.F. (2024). *Deep Learning Approaches to Predicting the Optimal Chess Moves from Board Positions*. International Journal of Engineering Trends and Technology, 72(4), 430–438. Retrieved from https://ijettjournal.org/Volume-72/Issue-4/IJETT-V72I4P105.pdf

[3] Kalagara, A. (2024). *Predictive Modelling of a Chess Player's Style using Machine Learning*. *National High School Journal of Science*. Retrieved from https://nhsjs.com/2024/predictive-modelling-of-a-chess-players-style-using-machine-learning/

[4] Gupta, A. (2023). *On the Value of Chess Squares*. Booth School of Business, University of Chicago. Retrieved from https://www.mdpi.com/1099-4300/25/10/1374

[5] Degni, R. (2023). *The Ultimate Checkmate: AI and Chess Engines. Codemotion Magazine*. Retrieved from https://www.codemotion.com/magazine/ai-ml/the-ultimate-checkmate-ai-and-chess-engines/

[6] Ferreira, D. R. (2010). *Predicting the Outcome of Chess Games Based on Historical Data* (Technical Report). IST-Technical University of Lisbon. Retrieved from https://web.ist.utl.pt/diogo.ferreira/papers/ferreira10report.pdf

[7] Levene, M., & Fenner, T. (2009). *A methodology for Learning Players' Styles from Game Records*. arXiv preprint arXiv:0904.2595. Retrieved from https://arxiv.org/pdf/0904.2595.pdf

[8] McIlroy-Young, R., Wang, R., Sen, S., Kleinberg, J., & Anderson, A. (2021). *Detecting Individual Decision-Making Style: Exploring Behavioral Stylometry in Chess*. In *Advances in Neural Information Processing Systems* (NeurIPS). Retrieved from https://papers.nips.cc/paper/2021/file/ccf8111910291ba472b385e9c5f59099-Paper.pdf

[9] Panchal, H., Mishra, S., & Shrivastava, V. (2022). *Chess Game Result Prediction Using Machine Learning*. In *2022 2nd International Conference on Advance Computing and Innovative Technologies in Engineering* (ICACITE), 2022, 970–975. IEEE. Retrieved from https://ieeexplore.ieee.org/document/9708405

[10] Oshri B., & Khandwala, N. (2015). *Predicting Moves in Chess Using Convolutional Neural Networks*. Stanford University. Retrieved from https://cs231n.stanford.edu/reports/2015/pdfs/ConvChess.pdf

[11] Al-Abdaly, N. M., Seno, M. E., Thwaini, M. A., Imran, H., Ostrowski, K. A., & Furtak, K.

(2024). *Advanced Ensemble Machine-Learning Models for Predicting Splitting Tensile Strength in Silica Fume-Modified Concrete*. *Buildings*, *14*(12), 4054. https://doi.org/10.3390/buildings14124054