Semester Project:

"How much influence does the opening move have on the game of chess?"

Group 10

Antoine Edelman <ae4fk@virginia.edu>, Xin Huang <xh2jg@virginia.edu>, Robert Knuuti <uqq5zz@virginia.edu>

School of Data Science, University of Virginia

DS 5110: Big Data Systems

Dr. Adam Tashman

November 30, 2021

Abstract

In this paper, we attempt to see if information at or before the first move sequence can be used to predict the winner of a chess game. Through statistical analysis we've concluded using a model-driven approach that ELO is the strongest influencer of a player winning or losing a match, regardless of skills or modeling method. Additionally we've concluded that when removing the ELO score that there were no strongly correlated parameters, showing that all other parameters are significant. No one opening appears to be stronger than another, although the King's pawn game appears to be fairly popular among the chess players. However this influencer only impacts 1-3% of the overall result, meaning that a model-driven assessment is not statistically significant. Overall, this shows that from our collected predictors that one cannot reliably predict the outcome of a chess match when training a model based on moves, skill level, and match types.

Introduction

"Chess is a game of strategy." This phrase is commonly spoken in the chess world as to demonstrate the complexity of the game theory that the board game represents. Since its creation in the 15th century, it has been widely played and various techniques and methods have been applied to how a game is played and many players continue to climb the ladder to mastery. What if it were possible to predict the outcome of a Chess game based on the first few moves taken? Perhaps there's an inherent advantage to playing as the white player? Are there other significant factors that may influence the outcome of the game that is significant?

In this report, we'll be seeking to answer these questions by reviewing a LiChess dataset published by Kaggle user *arevel*, which summarizes 6.3 million games of chess covering one month of chess play in 2016.

Data and Methods

Acquisition and Definitions

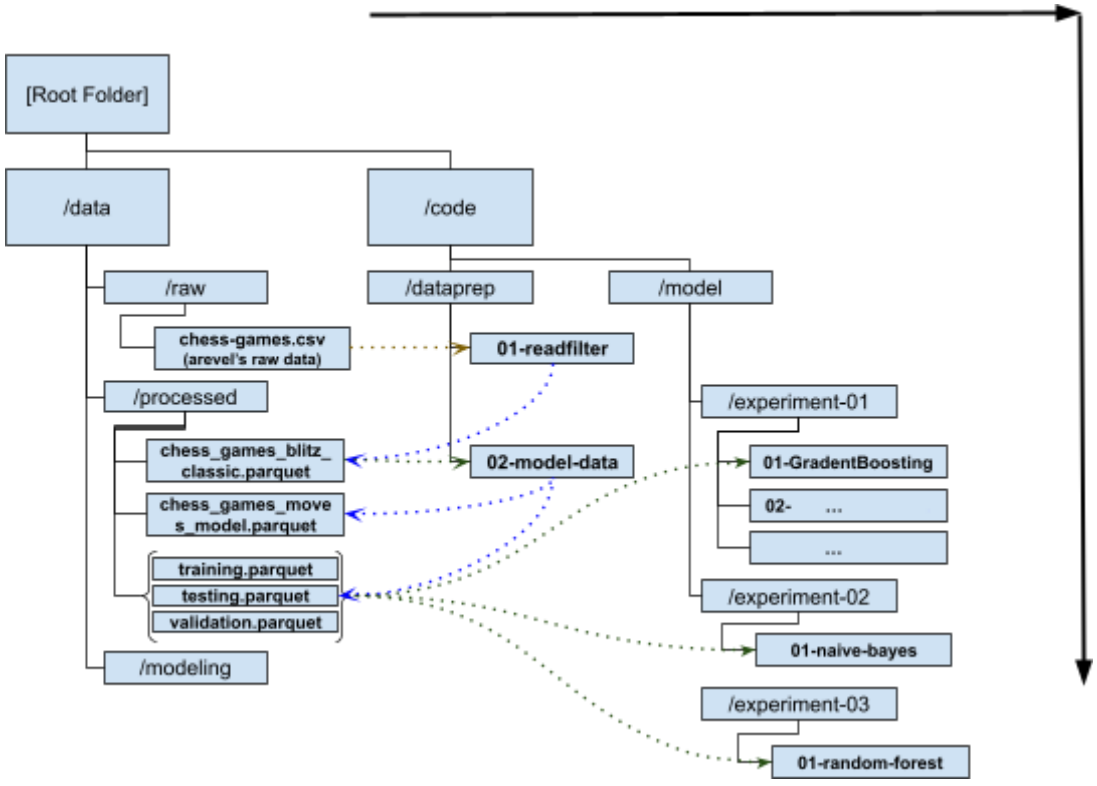
First, let's review the source data, we find multiple columns of data covering each of these matches.

Column	Definition
Event (categorical)	The type of chess game being played, including Classic, Blitz, Bullet
White, Black (categorical)	The associated ID for the white chess piece player
Result (categorical)	An encoded game result, where 1-0 indicates that White won, 0-1 indicates that Black won, and ½-½ indicating that the game was a tie, and "*" where a win/loss is not determined.
UTCDate, UTCTime (datetime)	The date and time of when the chess game started.
WhiteElo, BlackElo (integer)	The rated skill level of the player on the LiChess server, higher numbers indicate a stronger chess player.
WhiteRatingDiff, BlackRatingDiff (integer)	The difference in skill level with the opposite player. Positive values indicate the opposite player is weaker, and negative indicates the opposite player is stronger.
ECO (categorical)	The encoded opening is determined from the chess moves of the game. See https://www.chessprogramming.org/ECO for more details. Normally a Letter indicating a class of the opening, and a number for unique identifier.
Opening (categorical)	A common language name of the opening.

TimeControl (complex)	The total time of the game, divided by a + sign. The first number is the game's total clock time in minutes, where the number after the plus sign is the increment (Fischer Delay) that's added when changing turns.
Termination (categorical)	A categorical string indicating how the game ended, one of Normal, Time Forfeit, or Abandoned.
AN (complex)	Represents the Algebraic notation of the whole chess game as a single string. See https://en.wikipedia.org/wiki/Algebraic_notation_(chess) for more details.

Structure of Analysis

While constructing analysis, we closely aligned our techniques and notebook design leveraging Microsoft's Team Data Science Process (TDSP), dividing the acquisition, data wrangling, experiments, and persistence into separate domains to maximize reusability of the products from each phase of the project, and providing a workflow aligned structure to help manage the overall construction. This structure divides things into Code and Data, where Code contains a hierarchical structure for dataprep and models; and Data is used to store the various persisted data formats used throughout analysis.



As there are several notebooks used throughout this project, they are named with numeric prefixes to illustrate the order of execution. Note that dataprep should be fully executed prior to running any experiment, as all experiments depend on the data wrangling performed by the dataprep notebooks.

Feature Engineering, Data Exploration, and Approach

Reviewing our data, we made a few determinations based upon our prior knowledge of chess and based on the distribution of data.

Feature Engineering

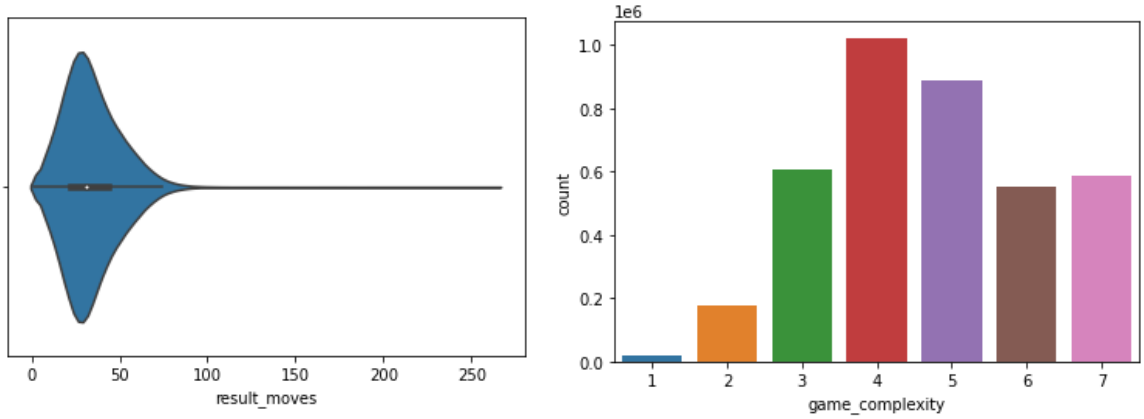
There were several features that we will be constructing on our data that appear to be useful.

Feature	Feature Source	Details
moves	AN	The AN field's raw format is great for compression and records keeping, but not good for modeling. Instead, we will transform this into an array of pairs that can be used to determine the length of the game as well as the index of each move taken in a match. This will be saved as a dense matrix so that future analysis can use this field with some more utility.
result_moves	moves	Given a set of moves, we'd like to understand the overall size of a match by move count. This feature takes our moves (generated from the AN list) and reports its length.
complexity	result_moves	Given our result_moves size, we create buckets overall length of the game (bucketing a match based on one move; or a "rage quit", and then incrementing by 10 up to 50, and then grouping all games greater than 50 as its own grouping)
EloDiff	WhiteElo, BlackElo	Given a game's White ELO and Black ELO rating, subtract the difference to determine the overall rating difference between players (positive indicates the white player is stronger, negative indicates black is stronger).
white_result	result	Indicator variable showing if white won. If marked as 1 then white was the winner of the match. If 0 then the match was either a tie or a loss.
Class_Type	ECO	Five categories were created based on the one of the five first initials for the ECO. (A => Flack Openings, B => Semi-Open Games, C => Open Games, D => Semi-Closed Games, and E => Indian Defenses)

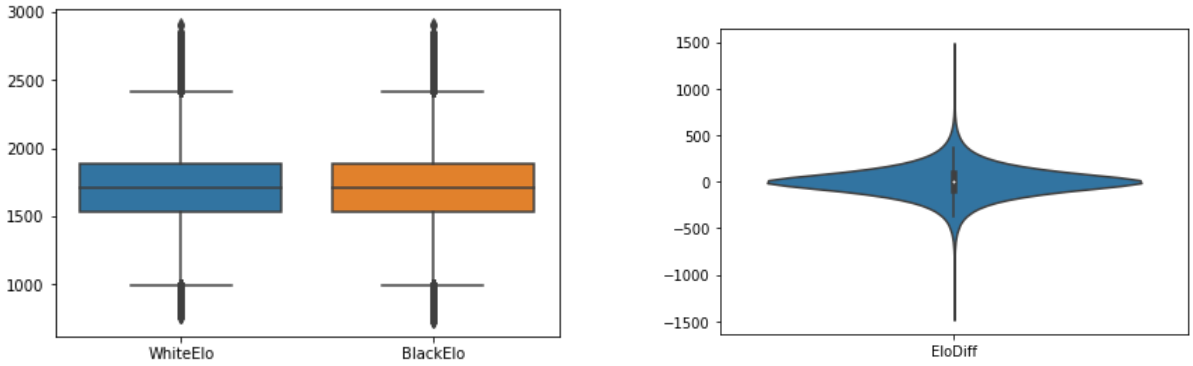
Exploration

We first investigated our existing metrics, to determine some of our dimensions of information, starting with our moves.

We see that with our move counts, there's a large right tail that we need to address to center our data. Reviewing the measures, we see that most games are completed around 30 moves, and this heavily tapers off around 60 moves. As such, we'll create a new feature to bin the move counts that will help better handle the outliers and normalize the distribution:



Next, let's review the ELO measures across our games and how well players are matched against each other.



Looking at our metrics, we find that the chess rating of players, on average, are all fairly close together, with the average score being around 1750 for both white and black. Similarly, we see that the matches that occurred were generally well met with equal rating levels (centering the mean on zero and tapering down, and rarely were there matches with greater than a 500 point difference.

While we investigated time control, it doesn't particularly have much to say about the game besides how the matches were categorized, although it does show that games played were typically initiated with the total length to not exceed 180 and 300 minutes or 3 to 5 hours. We also see that our match data shows that our distribution of winning games is fairly balanced, with white having a slightly higher count compared to black; and very few ties (intuitively this makes sense as ties in chess are quite hard to do).

Approach

Critically reviewing all our data, we find that most values are categorical in nature, and a few have complex representations that could be expanded upon in analysis. We also see that some data elements appear redundant such as Opening vs ECO, TimeControl vs Event, and WhiteRatingDiff vs BlackRatingDiff. However, the columns Opening and TimeControl are compressed representations of information.

For this analysis, we decided that we should focus on the two most popular categories of chess play: Bullet and Classic. These have a larger audience and record set that can help prevent selection bias, and also tends to map more effectively to traditional chess game play vs modern styles. Additionally, we decided to restrict games to those where the difference in chess ratings are within 500 of each other. This makes it so that we do not include chess matches that do not have a good matchup based on prior skill levels.

Results

Experiment 1 - Gradient Boosted Trees

For the first experiment, Gradient Boosted Trees are used due to their cutting edge abilities and recent prevalence. The goal is to see how certain variables created regarding the game can be changed to features for predictive capabilities. They are looked at individually, in pairs, and all together.

A filter on the data regarded ELO difference to see if it would change the accuracy of the model. As the table below shows, it didn't provide any insight or model improvement so was hence not used for the final model.

GBT Predictive ability by feature

Variable Type	AUC	AUC filtered
EloDiff	68.77%	57.06%
ECO_Type	53.46%	51.62%
Class_Type	51.47%	50.85%
event_vector	50.14%	50.10%

Here, the table shows that the ELO difference is the most effective at prediction while the event type has essentially no predictive ability. All variables across the board decreased when the ELO difference was filtered. This is expected for EloDiff but less expected for the others. The idea was by taking the skill level out of the equation it will reduce noise and increase the performance.

It's possible that the combination of features would have some type of impact on the accuracy. The table below shows those results. Event was removed from the following since it was essentially random.

GBT Predictive ability by feature

Variable Type	AUC	AUC filtered
EloDiff, ECO_Type, & Class_Type	68.83%	57.19%
EloDiff & ECO_Type	68.82%	57.16%

EloDiff & Class_Type	68.79%	57.09%
Class_Type & ECO_Type	53.74%	51.81%

Ordered by AUC, it is clear that the combination of features do increase the effectiveness of the model. This slight increase in accuracy means that each variable has insight in itself. However, it is very slight and will need better model tuning to increase accuracy.

With the best AUC using three features, a final model was created with cross validation using the following features and the corresponding results. The following variables (final value) were looked at in the tuning of the model: stepSize (1), minInstancesPerNode (25), subsamplingRate (1), and maxDepth (8). The metrics shown below are based on the validation data results instead of the performance of the test data shown above.

GBT final model metrics	
Metric	**Measure**
AUC	68.79%
Accuracy	63.09%
Precision	63.16%
Recall	61.66%
F1 Score	62.24%

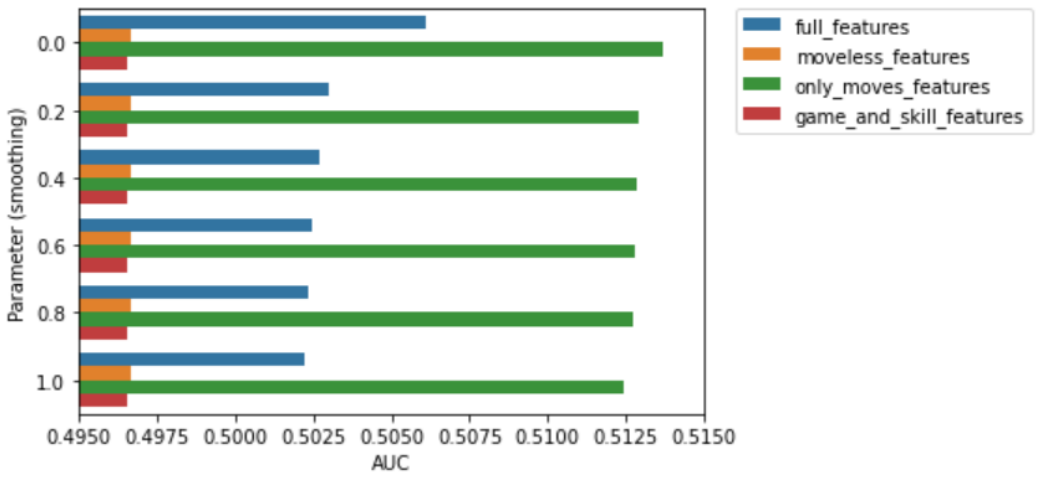
	Predicted Lose	**Predicted Win**
Actual Lose	375,226	206,536
Actual Win	220,215	354,145

Experiment 2 - Naive Bayes

For our second experiment, we use naive bayes as it appears that our predictors seem to be mutually exclusive (which is a trait of this model). Following the guise of our other models, we perform a categorical analysis to build a model to determine if our data can illustrate what it takes to make white win. Unlike the other experiments, we will build several different models using different groupings of predictors to help identify importance.

full_features	ECO, EloDiff, Event, and First two actions
moveless_features	EloDiff and Event
only_moves_features	First two actions
game_and_skill_features	ECO and EloDiff

Running our model against our training data, we resulted in the following performance measures from running cross validation with different smoothing values.



We see that tuning the hyperparameters for this model doesn't provide much impact to its performance on our training data (with the first significant change being ± 0.004; see the notebook for the explicit measures), however we do find that using a smoothing of 0 is superior to other options across all models and will be the tuned value for building our models.

From here we train our model against our set of predictors and evaluate their performance,

Metric	full	game_and_skill	moveless	only_moves
AUROC	0.506093	0.49655	0.496712	**0.513682**
Accuracy	**0.535799**	0.531128	0.503656	0.533177
F1	**0.525169**	0.517244	0.029259	0.520575
Precision	**0.533898**	0.529431	0.515683	0.531427
Recall	**0.516721**	0.505604	*0.015057*	0.510157

Overall, this model performs poorly, with the AUROC indicating that our model is only slightly better than random guessing with the full and only_moves feature sets. An interesting point is the moveless recall rating at 0.015, which is drastically different from all the other measures, meaning that this model has been overfit. Additionally we find that the difference in performance of predictors only varies by roughly 1-1.5% difference, which isn't a significant finding. However this model does demonstrate that by including the first two moves that there's a performance improvement in our prediction, meaning that our moves array does influence our response.

Experiment 3 - Random Forest

We first ran a random forest experiment against the dataset. Random forest has been chosen for the following reasons:

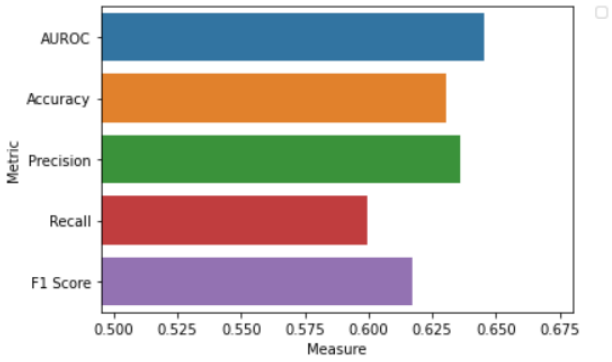
1. Random forest is scale invariant
2. Robust to irrelevant features;
3. Better interpretability.

We first split the dataset into training (60%) and testing (40%) data. The testing data is used to evaluate the final performance. We adopt the area under the curve AUC as the metric.

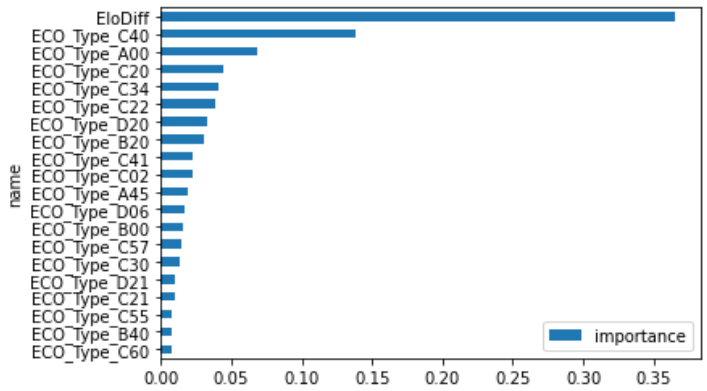
During cross validation, the following hyperparameters have been searched to obtain optimal performance

- Number of trees in random forest: num_trees = [20, 30]
- Maximum number of levels in tree: max_depth = [5, 8]
- Criterion used for information gain calculation criterion = ["entropy", "gini"]
- max_bins = [28, 32]
- samping_rate = [0.8, 1]

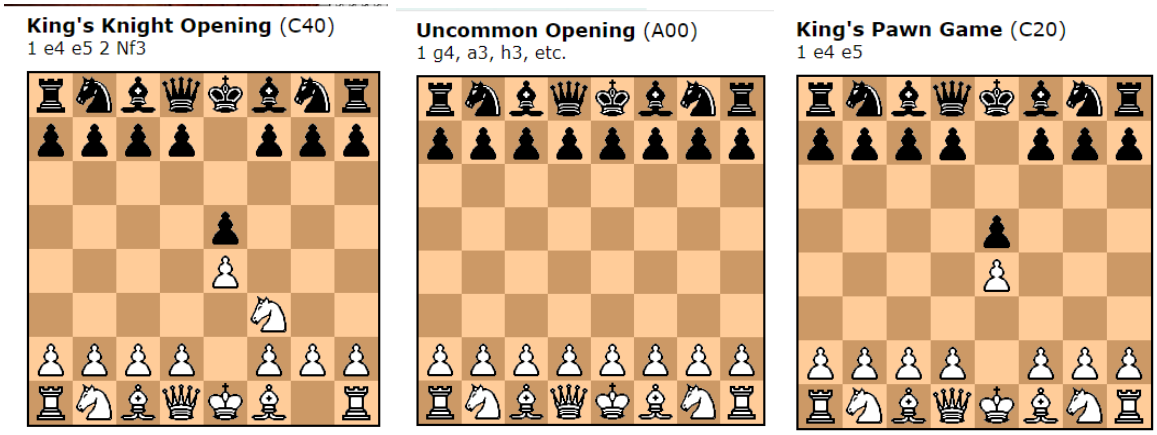
After intensive tuning using the tuning function from the spark ml library, we find the best combination of hyperparameters is numTrees = 30; maxDepth = 8; impurity = "entropy"; maxBins = 28; subsamplingRate = 1, which generates an AUC of 0.645.



Based on the optimal hyperparameter, we obtain the top 3 influential variables. As shown in tables below, we can see that EloDiff is the most influential variable. In the next section, we ran a second experiment on Gradient Boosted Trees to see if the performance can be further improved and whether the variable EloDiff stays the top feature as well.



name	importance
EloDiff	0.365457
ECO_Type_C40	0.138784
ECO_Type_A00	0.068527
ECO_Type_C20	0.044151
ECO_Type_C34	0.040995
ECO_Type_C22	0.038568
ECO_Type_D20	0.032479
ECO_Type_B20	0.031086
ECO_Type_C41	0.023108
ECO_Type_C02	0.022483
ECO_Type_A45	0.019380



Conclusion

We've now explored a snapshot into the world of chess and we can propose that most games are a 50-50 chance of winning or losing. Our models have demonstrated that predicting the outcome of a match given the move data is really only influenced by the ELO rating applied to the player. Knowing that ELO is a measure of a player's skill, this demonstrates that the strongest indicator of a chess match is really based upon the player's skill. All other predictors host only minor influences on the predicted outcome. Even so, knowing these points shows that the determination of a game win is not simply predicted based on some static information. Claude Shannon originally worked through this in his "Programming a Computer for Chess", explaining that it's impractical to program a computer to play perfect chess through enumeration (1950). He also explains that simply playing by the rules without strategy doesn't produce a good match either, although Deep Blue has proven that a brute-force AI can play chess well (Gonslaves 2017). However, this is only a brief look into chess. Given a larger dataset, or more extensive feature engineering, one could potentially find insights into play styles and potentially improve these models to something more significant.

References

Gonsalves, Tad (2017). "The Summers and Winters of Artificial Intelligence". In Khosrow-Pour, Mehdi (ed.). Encyclopedia of Information Science and Technology. 1. IGI Global. pp. 229–238. ISBN 978-1-5225-2256-0. https://www.google.com/books/edition/Encyclopedia_of_Information_Science_and/kvIoDwAAQBAJ?hl=en&gbpv=1&pg=PA229&printsec=frontcover

Microsoft. (2021, November 01). What is the Team Data Science Process?. https://www.who.int/news-room/fact-sheets/detail/the-top-10-causes-of-death

Revel, A. (2021, February 15). *Chess Games* (Kaggle, Version 1). [Data set]. Kaggle. https://www.kaggle.com/arevel/chess-games.

Shannon, Claude (1950). "Programming a Computer for Playing Chess". Philosophical Magazine. 41 (314). http://archive.computerhistory.org/projects/chess/related_materials/text/2-0%20and%202-1.Programming_a_computer_for_playing_chess.shannon/2-0%20and%202-1.Programming_a_computer_for_playing_chess.shannon.062303002.pdf

Wikipedia contributors. (2021, October 13). Algebraic notation (chess). In Wikipedia, The Free Encyclopedia. Retrieved 20:22, November 7, 2021, from https://en.wikipedia.org/w/index.php?title=Algebraic_notation_(chess)&oldid=1049790101

Wikipedia contributors. (2021, October 6). Encyclopaedia of Chess Openings. In Wikipedia, The Free Encyclopedia. Retrieved 20:25, November 7, 2021, from https://en.wikipedia.org/w/index.php?title=Encyclopaedia_of_Chess_Openings&oldid=1048580461