Machine Learning to Study Patterns in Chess Games

Student Number: 690065435

Academic Year 2022/2023

Abstract

Abstract here.

I certify that all material in this report which is not my own work has been identified.

Signature: _____

Contents

1 Project Specification, Motivation, and Aims

1.1 Project Specification

Chess is one of the oldest and most popular board games globally – it has a rich history and vast literature dating back centuries, with the first comprehensive book about chess called 'Book of the Chess' appearing in the year 840, discussing chess openings, endings, and hundreds of chess problems [1, 2]. However, the large-scale exploration of chess games has only caught on recently – the most well-known and highest-rated chess engine [3], Stockfish, has used over 9,400 years of CPU time to analyse over 5.6 billion self-play chess games as of April 2023 [4]. Websites such as Lichess and Chess.com have provided an online platform for people to play chess – users can create a free account and search for a game with other players with various time controls, where their matchmaking system will pair them with a player of a similar skill level. These platforms have made the game accessible to more people in standardised formats – this has enabled the collection of data on a large scale, which makes it a fascinating domain for data mining and machine learning. Furthermore, chess games involve cognition and human behaviour that we can apply more generally – there have been studies in areas like social learning theory on other games such as Go [5], but they also apply to chess.

In this dissertation, we explore data mining and machine learning techniques to analyse large-scale data from the Lichess Open Database, which contains monthly records on tens of millions of chess games played by users of various skill levels. Our goal is to discover patterns and insights into how people play chess, which could contribute to further studies in chess such as cheat detection algorithms to improve the integrity of chess tournaments and promote positive growth in the game.

I will investigate the following question: can we use data mining and machine learning techniques on historical data to discover patterns and insights into how people play chess? Our project has the following objectives:

- Create a data pipeline for downloading and processing the data efficiently
- Explore the data to find interesting patterns and insights
- Implement machine learning to create models based on patterns in chess games
- Analyse and evaluate the usefulness of the models

In the next section, we will explore previous research and how they relate to the motivations for our project, followed by the design section discussing how we will achieve these objectives.

1.2 Motivation and Aims

1.2.1 Early History of Computer Chess

Since the first digital computer switched on in 1945, programmers have been interested in using computers to play chess. In 1948, Alan Turing worked with David Champnowne to create the 'Turochamp', a machine routine for playing chess [6]. Claude Shannon wrote a paper in 1949 describing a computer that could play chess by analysing the seven most likely moves from the current positions and branching these into the seven most likely replies from the opposition continuously until the computer ran out of memory [7]. Dietrich Prinz eventually implemented the first chess program in November 1951, though it could not complete an entire chess game, and it used exhaustive search to play moves rather than a heuristic [6].

IBM made a significant breakthrough with their Deep Blue supercomputer starting with their initial prototype in 1995 [8]. It was released in 1996 [9] and was able to perform 200 million calculations per second [10]. Deep Blue was the first computer to use a heuristic search algorithm, searching for a solution to a problem by evaluating possible solutions and choosing the best one. In 1997, Deep Blue became the first computer to beat a world champion as it defeated Garry Kasparov over a six-game match [11], indicating that chess engines were better than human players. However, a lack of a deep understanding of the game was evident – Deep Blue accepted Kasparov's sacrifice of a rook for a bishop in the first game, only to lose 16 moves later [10].

1.2.2 Modern Developments in Computer Chess

Stockfish is the strongest chess engine in the world. Both Lichess and Chess.com use Stockfish to provide game analysis on their platforms, and it is also used by professional players to analyse their games. It was released as a free, open-source chess engine in November 2008 [12], with volunteers donating CPU time to test improvements to the engine using a distributed framework, Fishtest [13]. Stockfish uses the alpha-beta pruning search algorithm which improves minimax search [14] by avoiding variations that will never be reached in optimal play due to either player redirecting the game [15].

However, other chess engines have used different algorithms in recent history. Most notably, DeepMind released AlphaZero in 2017 [16] based solely on reinforcement learning – it had no knowledge of chess beyond the basic rules and learned within hours by playing millions of games [10]. AlphaZero used a general-purpose Monte Carlo tree search algorithm, a heuristic search algorithm [16]. It uses a tree to represent the game state, then randomly selects moves and uses the results of each game to weigh the tree nodes accordingly [17]. In December 2017, AlphaZero beat Stockfish in a 100-game match with 28 wins, 72 draws, and zero losses [18], showing the feasibility of alternative approaches, although Stockfish has since improved.

These developments in computer chess demonstrate the potential of big data in chess. While chess engines have gathered a lot of attention from researchers, there has been less attention paid to patterns and insights into how people play chess, which is what our project will focus on.

1.2.3 Growth in the Popularity of Chess

Chess has seen a resurgence in popularity over the last decade. The rise of chess platforms such as Lichess and Chess.com have enabled people to play chess games online with other people globally in seconds. Lichess recorded 121,332 standard-rated games in January 2013, which rose to 103,178,407 games a decade later in January 2023 [19].

More recently, popular culture has driven a chess boom. As the COVID-19 pandemic struck in late 2019, people turned to chess to fill their spare time. The release of The Queen's Gambit in October 2020 accelerated people's interest – Netflix announced that a then-record 62 million households watched the series in its first 28 days [20]. Chess.com saw its monthly active users double from around 8 million to nearly 17 million between October 2020 and April 2022 [20]. It also reported a record for the monthly number of chess games played, with 1 billion games played in February 2023 [21]. Chess was also featured in the most popular social media post in 2022, featuring Cristiano Ronaldo (the most followed user on Instagram) playing chess with Lionel Messi on an Instagram post that has received over 42,800,000 likes as of April 2023 [22].

The fast growth of chess has given us increasing amounts of and better quality data to analyse, which is critical to the success of our project. As previously mentioned, Lichess records tens of millions of games every month, and they make this accessible to the public through their Open Database [19], which is what our project will use as its data source.

1.2.4 Practical Uses of Chess Databases

Numerous chess databases have been created and made publicly available, such as the Lichess Open Database [19] and the FICS Games Database [23]. They store chess games in PGN (Portable Game Notation) format, containing metadata via headers such as the players involved and the result, plus the sequence of moves in the game. While PGN is the standard file format, the metadata headers are not standardised. For example, the FICS Games Database records the ply count (the total number of turns the game lasted), whereas Lichess does not.

The practical uses of chess databases have come under scrutiny lately with the Hans Niemann scandal in September 2022, when reigning five-time World Chess Champion Magnus Carlsen withdrew from a tournament and accused Niemann of cheating over the board [24]. Chess.com conducted an investigation using a variety of analytics tools designed for cheat detection, such as comparing Niemann's moves to the moves recommended by the top chess engines, leading to them finding Neimann guilty [25]. Similarly, Kenneth Regan, often paid by the International Chess Federation (FIDE) to monitor tournaments [26], concluded that Niemann was likely cheating according to his proprietary

cheat detection software that uses a statistical model to output a z-score representing the degree of similarity between the player and chess engines [26].

Our project will seek to provide insights into patterns in chess games, which could contribute to improvements in cheat detection algorithms and ultimately improve the integrity of chess tournaments and the broader chess landscape.

2 Design, Methods, and Implementation

2.1 Downloading the Data

Collecting the data correctly was paramount to the success of our project, and it was important to use a large sample size to ensure that our insights represent the general population of chess games. We used the Lichess Open Database [19] of standard rated games for our data source – they upload tens of millions of games every month in PGN format, and they are easily accessible to the public. We decided to focus on games in 2022, as this enables me to capture the latest trends in chess.

The time controls in Lichess are decided based on the estimated game duration, using the formula: $\textit{total time in seconds} = (\textit{initial clock time}) + 40 \times (\textit{clock increment})$. Games between 29 seconds and 179 seconds are Bullet; games between 179 seconds and 479 seconds are Blitz, games between 479 seconds and 1499 seconds are Rapid, and games over 1500 seconds are Classical. Furthermore, games are either Rated or Unrated – the former results in rating points changing based on the game outcome, whereas the latter does not. The Lichess Open Database only includes Rated games. We will only be analysing Rated Bullet, Rated Blitz, and Rated Rapid games – we will not be including Rated Classical games, as they have a significantly smaller sample size and the longer time controls result in a vastly different playstyle.

We started by downloading the PGN files for each month, which introduced difficulties with big data. Each month's file is around 30 GB to download and they are compressed, which means each month is around 210 GB when uncompressed, resulting in approximately 2.5 TB of data for the year. My laptop only had 1 TB of storage, and the sheer size of the data would make it unrealistic to process in a reasonable amount of time. Therefore, we collected a sample of 6 million chess games from each month which we would later filter down.

2.2 Data Processing

PGN (Portable Game Notation) files are a standard format for recording chess games – each game has headers containing information like the White player, Black player, their ratings, and the result of the game. In addition, it stores a list of all the moves in each game. While PGN files are well-structured, they are not easy to process. The pgn2data Python library [27] provides a simple way to convert PGN files into CSV files, but it exported both the moves and the metadata of each game in separate CSV files. We did not need the moves, so it was using up unnecessary space and taking a long time to process each game. It was also designed for generic PGN files so it did not include additional useful metadata provided by Lichess such as the opening of the game.

Thus, we wrote our own Python script to convert each PGN file into a single CSV file containing only the metadata of each game. We used the python-chess library to parse the PGN files and iterate over each game, skipping Rated Bullet games, Rated Classical games, games that involved a non-human player, and games that were abandoned. Our sample reduced from 60 million games to slightly over 40 million games after filtering it. To optimise performance, we split each month's PGN into six files each containing 1 million games with pgn-extract (a PGN manipulator written in C) [28], and we used Python's multiprocessing library to process each month in parallel – this reduced the processing time from 120 minutes to 20 minutes for each month on my Apple M1 Max chip.

Once we had a CSV for each month, we combined them into a single CSV file containing our final data set. We converted this CSV to a folder of Parquet files using Dask [29], which is a library for parallel computing in Python and extends common interfaces like pandas [30] to handle big data in a larger-than-memory environment. This significantly reduced the load time for the data, and it meant we could use the Dask library to perform parallel computations on the data.

For low-level, turn-based analysis of games, we created a .scout file using Scoutfish (a tool for querying chess databases written with C++) [31] to enable fast queries. For example, we can query the database to find all games where the board state is a certain position using the FEN (Forsyth-Edwards Notation) of the position, which is the standard string used in chess to represent the board state – this may be useful for finding occurrences of a specific variation of an opening. It provides the offset line number of each matching game in the PGN file, which we can use to find more information about the game. However, Scoutfish queries are not intuitive to write, so it can take a lot of effort to get insights using this tool. Therefore, we will primarily focus on high-level analysis of game metadata using Dask DataFrames.

Figure 1: Diagram of the Data Pipeline

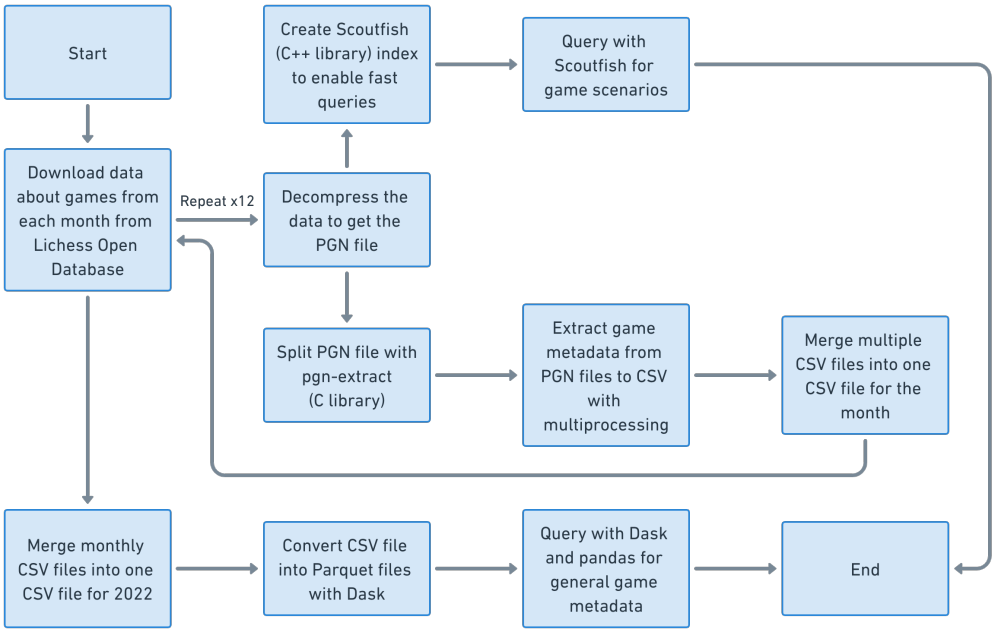


Figure 1 shows an overview of our data pipeline. It shows how we used the Lichess Open Database to collect the data over 12 months. To analyse game metadata, we aggregated the data into a single CSV file, and converted it to a folder of Parquet files to query with Dask. Meanwhile, to perform more detailed analysis of game scenarios, we used this data to create a Scout index to enable fast queries.

We also created a Python script to convert the CSV file to an SQLite3 database – this enables us to perform queries on the data over SQL for manual data exploration, which is easier and more interactive than using Dask functions.

2.3 Subsampling for Machine Learning Models

Machine learning is computationally expensive, so generating and testing machine learning models in a reasonable amount of time would be infeasible. Therefore, we will further subsample the data. We will use a random sample of 12.5% of the original sample size, which is approximately 5 million games. Since we are not trying to predict rare events and we are still using a large sample, our subsampling should have minimal impact on the representativeness of our insights – patterns that we identify for 5 million games should also be present in the complete data set.

3 Project Results and Evaluation

In this section, we will present the results of our data analysis and evaluate their insights on a cyclical basis – this enables us to scrutinise our findings and probe new directions accordingly. We will start by exploring the data to understand the data set better before implementing machine learning models to uncover patterns in how people play chess.

3.1 Data Exploration

3.1.1 Structure of Data

Our data consists of 40,121,728 Rated Blitz and Rated Rapid games that were played on Lichess in 2022. Each game is described by: the datetime of when the game started, what type of game (Rated Blitz or Rapid) it was, the specific time control, outcome of the game (1-0 for a White win, 1/2-1/2 for a draw, and 0-1 for a Black win), how the game ended, ECO category of the opening, the specific opening that was played, who played White and Black, their Lichess ratings, and a URL to the game on Lichess. Figure 2 shows the head of the DataFrame, which contains the first two games in the data sample – note that this has been transposed to maintain readability.

Figure 2: Structure of Data

	0	1
UTCDateTime	2022-01-01 00:00:11	2022-01-01 00:00:11
Event	Rated Rapid game	Rated Blitz game
TimeControl	600+0	300+3
Result	1-0	0-1
Termination	Normal	Normal
ECO	B00	D00
Opening	Nimzowitsch Defense: Scandinavian Variation, A...	Queen's Pawn Game: Mason Variation
White	shahzad97	nicolas_clos
Black	mpasha98	pangma
WhiteElo	1500	1430
BlackElo	1205	1412
Site	https://lichess.org/UMMGcaGz	https://lichess.org/yJI78ET0
BaseOpening	Nimzowitsch Defense	Queen's Pawn Game
EloDiff	295	18
RelativeEloDiff	10.91	0.63

3.1.2 Distribution of Player Ratings

We counted the ratings of players in our data set. Rating systems are designed to predict the outcome of games so that players can be matched fairly. There are various rating systems used in chess – the most common is the Elo rating system, which is used by various sports. Lichess uses the Glicko 2 rating system, which accounts for volatility amongst other factors to provide better prediction accuracy than the Glicko 1 and Elo rating systems [32, 33]. The Glicko 2 rating system starts at 1500, and it is artificially limited to a minimum of 600 in Lichess. We created bins of 200 rating point ranges starting from 600 – this handles the representation of new players well, as those who have played few games will not have changed their rating significantly and likely get placed in the 1400-1600 bin. Note that the last four bins on the right of the bar chart do exist, but they are difficult to see – they contain totals of 11,622 for 2800-3000, 382 for 3000-3200, 459 for 3200-3400, and 1 player for 3400-3600. In addition, the counts are based on each game played, so some players may be overrepresented in the data set – a player who plays 100 games within the year will have 100 entries in the data set, whereas a player who plays 1 game will have 1 entry. However, this is not a significant issue because the data set is large enough to represent the distribution of player ratings well.

Figure 3: Distribution of Player Ratings on Lichess in 2022

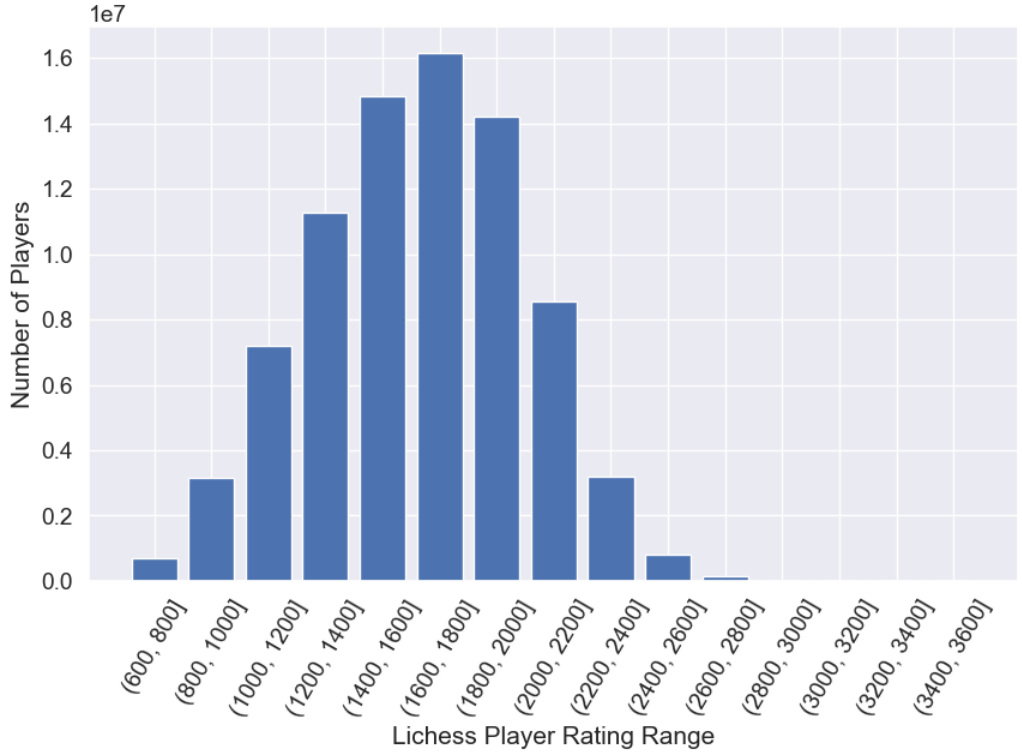


Figure 3 shows that the distribution of player ratings tends towards a normal distribution. We could explain this based on the concept that a player's performance is similar to a random walk, which states that an object moves randomly with equal probability in different directions. In Lichess ratings, a player's rating can be seen as the result of a series of random walks, where the player's rating constantly changes as they win and lose games. Over time, their rating will tend to stabilise to represent their true skill level. Moreover, the central limit theorem states that the sum of a large number of independent and identically distributed random variables tends towards a normal distribution [34]. In this case, the ratings of individual players can be seen as independent and identically distributed random variables, with each game representing a single trial. Therefore, the central limit theorem predicts that the distribution of ratings will tend towards a normal distribution as the number of games increases. This is supported by how largest bin is the 1400-1600 bin, which contains the starting rating of 1500. Lichess' matchmaking system could contribute to this – it matches players with similar skill, so games theoretically end up with a roughly equal number of wins and losses.

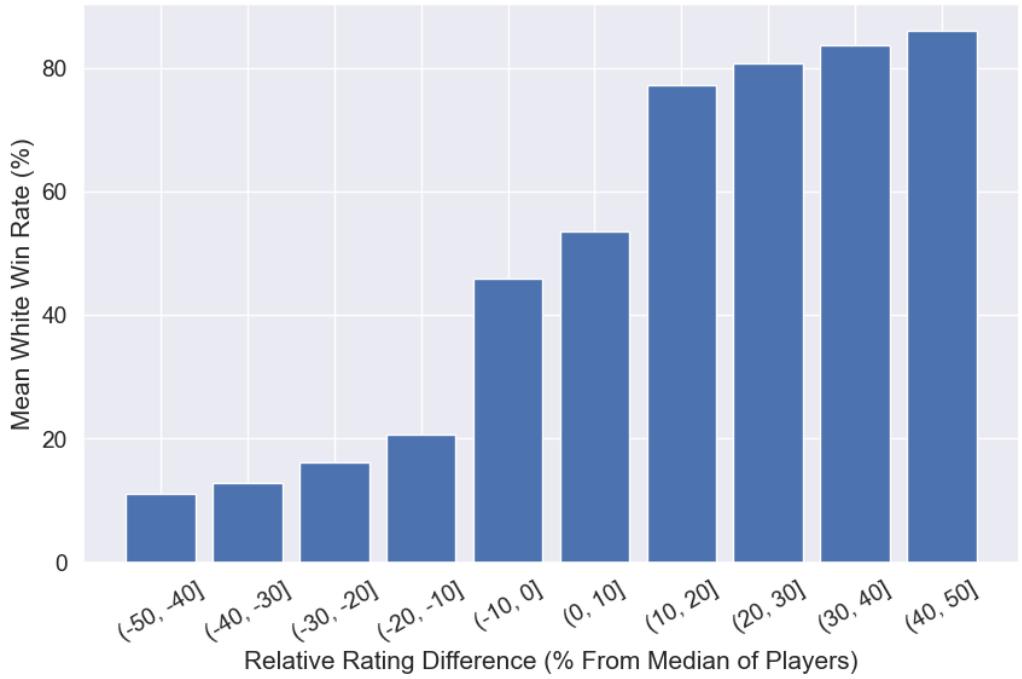
3.1.3 Effect of Rating Difference on Win Rate

Next, we investigated the impact of player rating differences on White's win rate. To begin with, we counted the outcomes of all games, and then counted the outcomes of games when White had a higher rating – White won 49.69% of all games, which rose to 53.98% of games when they had a higher rating than their opponent (as shown in figure 14 from the appendices). Then, we sought to quantify the impact of rating difference in both absolute and relative terms – the former was the difference in rating between the two players, and the latter was the difference in rating divided by the mean rating of the two players. Figure 15 and figure 16 from the appendices show the distribution of the absolute and relative rating differences. We grouped the data by rating difference and calculated the win rate for White. For example, when White is rated 1100 and Black is rated 900, the absolute rating difference is 200, whereas the relative rating difference is 10%. We focused on the win rate for White because the opening of the game is usually decided by White.

The results showed that there is a clear positive correlation between rating difference and White's win rate. Both the absolute and relative difference metrics corroborate this, but we saw that relative rating difference highlighted the impact of rating on White's win rate more clearly. Absolute rating

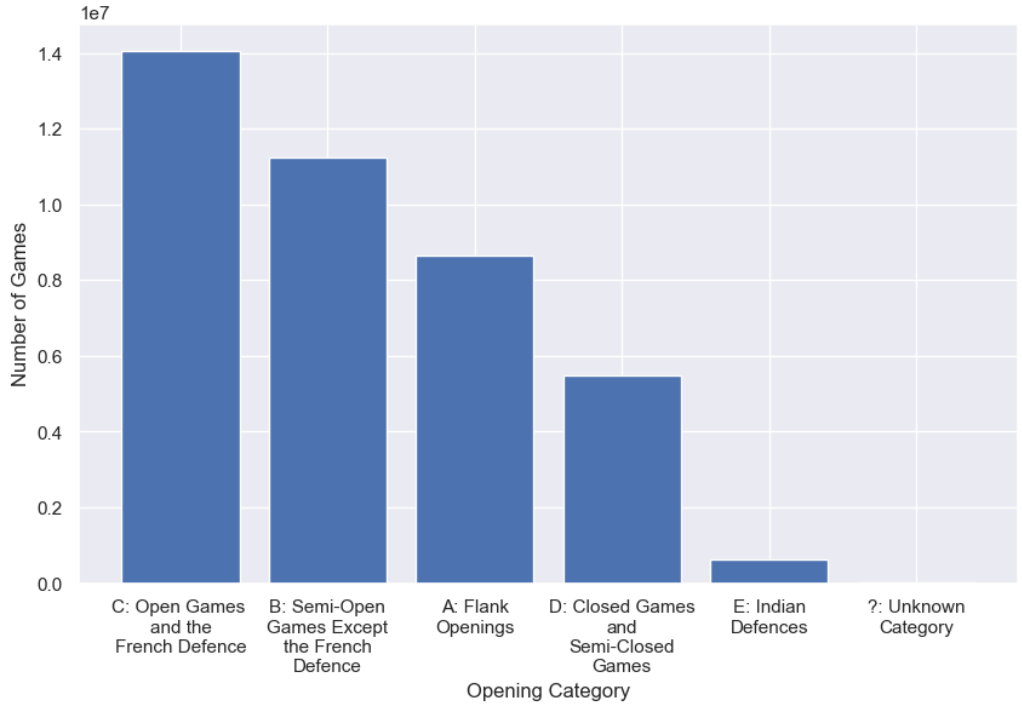
difference (shown in figure 17 from the appendices) is more intuitive to understand, but it is less accurate than relative rating difference (shown in figure 4) because it is not normalised.

Figure 4: Mean White Win Rate for Each RelativeEloDiff Bin



3.1.4 Most Popular Openings by Category

Figure 5: Most Popular Openings by Category on Lichess in 2022



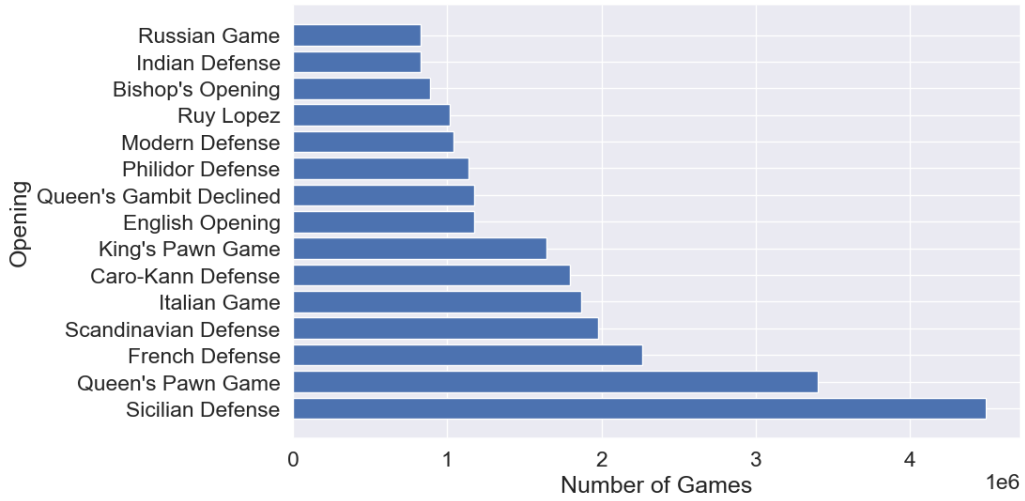
We also counted the most popular openings by category in figure 5, as this is represented by the ECO column provided in the Lichess PGN data. ECO (Encyclopedia of Chess Openings) is a classification of chess openings based on the first few moves of the game. Editors who mostly consist of chess grandmasters select critical opening lines and assign them a code [35]. The ECO system consists of

five categories, A to E, each of which represent a category of opening. Within each category, there are further subcategories to group openings more explicitly. For example, a game with an ECO code of A41 means that the opening is a flank opening as it is in the A category, and the 41 means that the opening was the Queen's Pawn Game. The ECO system is not perfect as it does not include all openings (as demonstrated by the 32,799 games with the unknown category in figure 5), but it is useful to analyse opening trends.

Figure 5 shows that the most popular category of openings are open games and semi-open games, whereas the Indian defences and closed games are the least popular. This makes sense for the general population of games – the ideas behind these openings are simple and straightforward, as they focus on controlling the centre and developing pieces quickly. However, we see that the pattern is different for players rated 2000 and above – semi-open games and flank openings are more popular than open games in this category (shown in figure 18 from the appendices). At higher rated games, players often prefer to play more complex and dynamic openings that enable opportunities to create imbalances in the position and therefore gain an advantage at an earlier stage of the game.

3.1.5 Most Popular Base Openings

Figure 6: Most Popular Base Openings on Lichess in 2022



The data set also includes a column for the specific opening detected in each game based on Lichess' opening detection. We grouped openings that were different variations of each other into base openings to get a more accurate representation – figure 21 from the appendices contains examples of base openings on the chess board. The top 15 most popular base openings account for 61.79% of all games in our data set. Figure 6 shows that the most popular openings are the Sicilian Defence and the Queen's Pawn Game by a significant margin. The Sicilian Defence is the most well-studied response to White moving the king's pawn forward by two, which could explain why it is the most popular opening. However, the Queen's Pawn Game is slightly misleading, as it describes any opening beginning with White moving the queen's pawn where they do not play the Queen's Gambit and therefore encompasses a large number of openings such as the London System which are not labelled separately.

Delving deeper into the data, we found that the most popular base openings varied by rating groups. The top players rated 2000 and above strongly preferred the Sicilian Defense compared to other openings (shown in figure 19 from the appendices). This may be because the Sicilian Defense is a complex opening that requires a lot of preparation due to the number of variations, and it is unanimously accepted as the best response to White opening with the king's pawn. Conversely, the Sicilian Defense becomes much less popular for players rated 1200 and below for the same reasons. Figure 20 from the appendices shows that the two most popular base openings for this group are the Queen's Pawn Game and the King's Pawn Game, which represent generic openings starting with the

queen's pawn and the king's pawn respectively – this is likely because they are simple and easy for beginners to play.

3.2 Predicting Game Results Using Decision Tree and Random Forest Classifiers

We sought to predict the result of a game using both decision tree classifiers and random forests with various features. Based on the previous section, we focused on features related to player rating and opening. Figure 7 summarises the results we obtained from our models, grouped by type of classifier and sorted by F1 score. We used F1 score as our evaluation metric because it is a good measure of accuracy for imbalanced data sets such as ours, where there are fewer draws compared to wins and losses for White, as it accounts for a balance between precision and recall. We also used accuracy as a secondary metric to compare the performance of our models.

Figure 7: Results of Models to Predict the Result of a Game

Classifier	**Features**	**F1 Score**	**Accuracy**
Decision Tree	BaseOpening, RelativeEloDiffBin (5% Increments)	0.510	0.521
Decision Tree	ECO, RelativeEloDiffBin (5% Increments)	0.508	0.521
Decision Tree	BaseOpening, EloDiffBin (10 Increments)	0.498	0.525
Decision Tree	BaseOpening, EloDiff	0.495	0.519
Decision Tree	BaseOpening, RelativeEloDiff	0.492	0.511
Decision Tree	BaseOpening, RelativeEloDiff, TimeControl	0.486	0.496
Decision Tree	BaseOpening, WhiteElo, BlackElo	0.474	0.476
Random Forest	BaseOpening, RelativeEloDiffBin (5% Increments)	0.509	0.521
Random Forest	BaseOpening, RelativeEloDiffBin (10% Increments)	0.509	0.521
Random Forest	ECO, RelativeEloDiffBin (5% Increments)	0.508	0.521
Random Forest	BaseOpening, RelativeEloDiff	0.490	0.511
Random Forest	BaseOpening, RelativeEloDiff, TimeControl	0.484	0.497

In the results table, you can see that we denoted the RelativeEloDiffBin and EloDiffBin features with different increments. We previously binned these features to explore the distribution of values, but we wanted to see whether using finer binning would improve the performance of our models. Hence, we tested the same model but with different binning increments, which we did not include in the table for brevity – the increments included in the table represent those that produced the best F1 score and accuracy.

There was little difference in the performance of our decision tree classifiers and our random forest classifiers – both of them achieved similar F1 scores and accuracy when using the same features. However, we saw some variation in performance when using different features. We saw an improvement by binning the RelativeEloDiff values instead of using the raw values – this is likely because it helps to smooth the noise of tiny differences and help extract the general information that varying RelativeEloDiff provides. Using 5% increments was the sweet spot between binning too finely and thus introducing noise, and binning too broadly and losing the value of information. Meanwhile, using ECO values rather than BaseOpening values resulted in worse performance, likely due to the ECO categories being too vague, with openings in the same category being vastly different in some cases.

While there are some differences in results between the variations of our models, our classification models generally produced poor F1 scores and accuracy. The best F1 score was 0.510 while the best accuracy was 0.525 – this indicates that our features were not good predictors of the result of a game. Lichess' matchmaking system pairs opponents of similar skill levels. Thus, the difference in rating between players is often minimal, resulting in a small sample size for the rating difference bins, which may have contributed to the poor performance. Furthermore, the outcome of a chess game is highly dependent on how the players perform in the game, which is subject to variation, especially in lower-rated games due to human nature – these are not features that we have in our data set, nor are they features that we can feasibly obtain.

3.3 Clustering Base Openings by Game Results

Following our earlier exploration of the most popular base openings, we investigated whether we could cluster them by game results. Accurate clusters would allow us to identify base openings that are more likely to result in a certain outcome – this could be useful in determining someone's style of play, thus opening the possibility for more fine-tuned analysis of match-ups between players.

3.3.1 Calculating Proportion of Game Results Per Base Opening

Before applying our clustering algorithm, we needed to aggregate the data to obtain the proportion of game results per base openings. This involved counting the number of outcomes for each base opening, and then calculating the proportions of each outcome. Note that we filtered the data to only include base openings with a significant sample size to remove outliers and thus improve the quality of our clustering. We required a minimum of 10,000 games, which was reasonable for the subsample of approximately 5 million games. Figure 8 shows the proportion of game results for the top 15 base openings, sorted in descending order of the number of games.

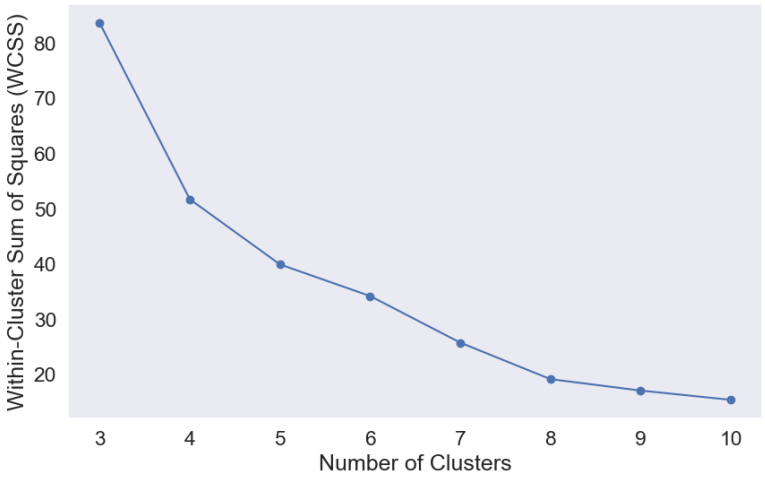
Figure 8: Proportion of Game Results for Top 15 Base Openings

Result	BaseOpening	0-1	1-0	1/2-1/2	NumberOfGames
0	Sicilian Defense	47.216623	48.439828	4.343549	563111
1	Queen's Pawn Game	45.039865	50.393009	4.567126	425563
2	French Defense	47.288915	48.353329	4.357756	283724
3	Scandinavian Defense	46.489923	49.226190	4.283888	247929
4	Italian Game	44.863825	51.363461	3.772714	233890
5	Caro-Kann Defense	47.645196	47.770477	4.584327	225093
6	King's Pawn Game	46.211763	49.641571	4.146666	205201
7	Queen's Gambit Declined	43.947297	51.320293	4.732410	147240
8	English Opening	44.784967	50.485404	4.729629	147094
9	Philidor Defense	44.547113	51.168048	4.284839	141989
10	Modern Defense	46.655062	49.019217	4.325721	130406
11	Ruy Lopez	44.377249	50.972528	4.650223	126166
12	Bishop's Opening	45.022163	51.165676	3.812161	111223
13	Indian Defense	46.588548	48.835912	4.575541	103944
14	Russian Game	45.739654	50.287710	3.972636	103055
15	Scotch Game	43.028900	52.729006	4.242094	102803

3.3.2 Implementing K-Means Clustering

We decided to use K-means clustering over alternatives such as hierarchical agglomerative clustering because it is easy to implement and interpret, and our underlying data is relatively simple. We used the percentages of each game result for our features and we used the elbow method to determine the optimal number of clusters. The elbow method involves plotting the explained variation (also known as the within-cluster sum of squares – WCSS) as a function of the number of clusters, and then choosing the number of clusters at the elbow of the curve. Figure 9 shows the elbow method for our data – we can see that the elbow occurs at four clusters.

Figure 9: Elbow Method for K-Means Clustering of Base Openings by Game Results

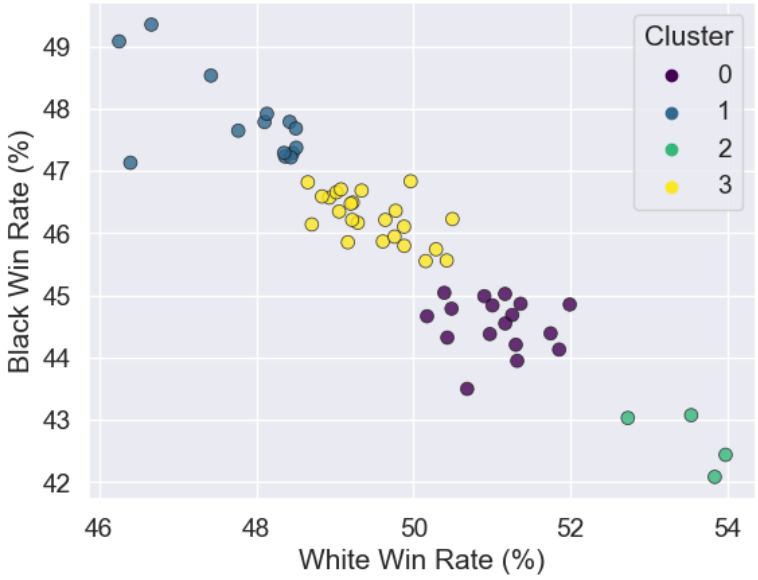


Additionally, we recorded the silhouette score for each number of clusters to validate our choice. The silhouette score is a measure of how similar an object is to its own cluster compared to other clusters – it ranges from -1.0 to 1.0, where a higher score is good and indicates that the object is well-matched to its own cluster and poorly-matched to neighbouring clusters. Figure 10 shows the silhouette score for each number of clusters from 3 to 10 – it confirms that the optimal number of clusters is four, with the best silhouette score of 0.478.

Figure 10: Silhouette Scores for K-Means Clustering of Base Openings by Game Results

Number of Clusters	**Silhouette Score**
3	0.441
4	0.478
5	0.457
6	0.432
7	0.427
8	0.439
9	0.440
10	0.432

Figure 11: Base Openings K-Means Clustered by Game Results



3.4 Regression Analysis of Base Opening Popularity

Figure 12: Popularity of Base Opening vs. White Win Rate (All Rated Players)

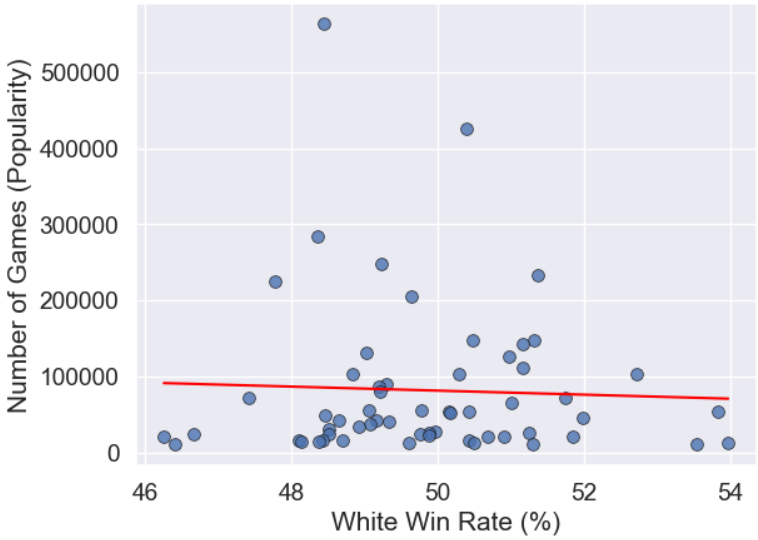
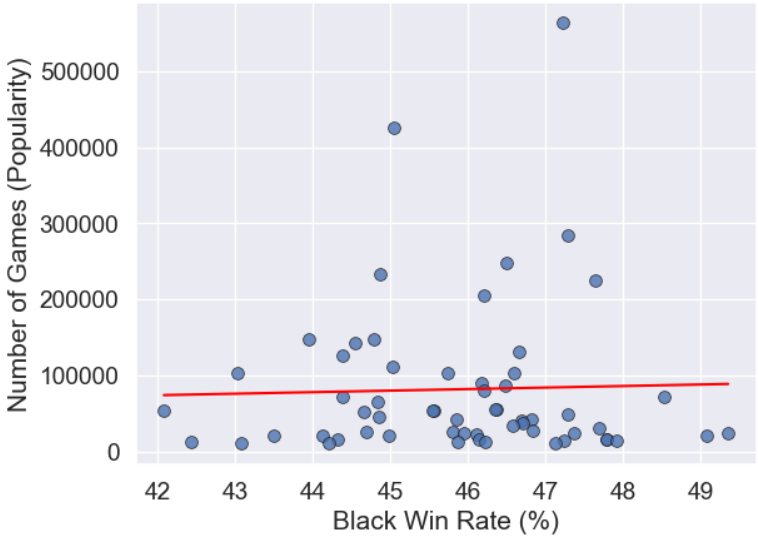


Figure 13: Popularity of Base Opening vs. Black Win Rate (All Rated Players)



4 Project Discussion and Conclusion

4.1 Project Outcomes

4.2 Limitations

4.3 Future Work

References

[1] B. Wall, "Earliest Chess Books and References." `http://billwall.phpwebhosting.com/articles/oldtexts.htm`, 2013. Date Accessed: 3 April 2023.

[2] P. R. Wonning, *A short history of the game of chess: Chess history in brief*, vol. 6. Mossy Feet Books, 2014.

[3] "Computer Chess Rating Lists: April 2023." `https://ccrl.chessdom.com/ccrl/4040/`, 2023. Date Accessed: 3 April 2023.

[4] Stockfish, "Stockfish Testing Framework." `https://tests.stockfishchess.org/users`, 2023. Date Accessed: 3 April 2023.

[5] B. A. Beheim, C. Thigpen, and R. Mcelreath, Strategic social learning and the population dynamics of human behavior: The game of Go *Evolution and Human Behavior*, vol. 35, no. 5, pp. 351–357, 2014.

[6] B. J. Copeland and D. Proudfoot, Turing and the computer *Alan Turing's Automatic Computing Engine: The Master Codebreaker's Struggle to Build the Modern Computer, Oxford University Press, Oxford*, pp. 107–148, 2005.

[7] C. E. Shannon, XXII. Programming a computer for playing chess *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, vol. 41, no. 314, pp. 256–275, 1950.

[8] F.-h. Hsu, M. S. Campbell, and A. J. Hoane Jr, Deep Blue system overview in *Proceedings of the 9th international conference on Supercomputing*, pp. 240–244, 1995.

[9] F.-h. Hsu, IBM's deep blue chess grandmaster chips *IEEE micro*, vol. 19, no. 2, pp. 70–81, 1999.

[10] S. Strogatz, One giant step for a chess-playing machine *New York Times*, pp. 1–6, 2018.

[11] Y. Seirawan, H. A. Simon, and T. Munakata, The implications of Kasparov vs. Deep Blue *Communications of the ACM*, vol. 40, no. 8, pp. 21–25, 1997.

[12] "About Stockfish." `https://stockfishchess.org/about/`, 2022. Date Accessed: 22 November 2022.

[13] "Fishtest Distributed Testing Framework." `https://www.talkchess.com/forum3/viewtopic.php?start=0&t=47885`, 2022. Date Accessed: 22 November 2022.

[14] J. v. Neumann, Zur theorie der gesellschaftsspiele *Mathematische annalen*, vol. 100, no. 1, pp. 295–320, 1928.

[15] S. Maharaj, N. Polson, and A. Turk, Chess AI: competing paradigms for machine intelligence *Entropy*, vol. 24, no. 4, p. 550, 2022.

[16] D. Silver, T. Hubert, J. Schrittwieser, I. Antonoglou, M. Lai, A. Guez, M. Lanctot, L. Sifre, D. Kumaran, T. Graepel, *et al.*, Mastering chess and shogi by self-play with a general reinforcement learning algorithm *arXiv preprint arXiv:1712.01815*, 2017.

[17] G. Chaslot, S. Bakkes, I. Szita, and P. Spronck, Monte-Carlo tree search: A new framework for game ai in *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, vol. 4, pp. 216–217, 2008.

[18] M. Klein, Google's AlphaZero destroys Stockfish in 100-game match *Chess. com*, vol. 6, 2017.

[19] Lichess, "lichess.org open database." `https://database.lichess.org/#standard_games`, 2022. Date Accessed: 1 March 2023.

[20] G. Keener, "Chess is Booming." `https://www.nytimes.com/2022/06/17/crosswords/chess/chess-is-booming.html`, 2022. Date Accessed: 7 April 2023.

[21] L. Watson, "Chess Boom Hits New Heights With 1 Billion Games Played In February." `https://www.chess.com/news/view/chess-boom-1-billion-games-played-in-february`, 2023. Date Accessed: 7 April 2023.

[22] C. Ronaldo, "Victory is a State of Mind." `https://www.instagram.com/p/ClJqW1ZtmIl/`, 2022. Date Accessed: 7 April 2023.

[23] F. I. C. Server, "FICS Games Database." `https://www.ficsgames.org/`, 2023. Date Accessed: 7 April 2023.

[24] S. Jenkins, "When chess is hard and cheating is easy, the next move is complicated." `https://www.washingtonpost.com/sports/2022/09/27/magnus-carlsen-hans-niemann-chess-cheating-controversy`, 2022. Date Accessed: 23 November 2022.

[25] A. Beaton and J. Robinson, "Chess Investigation Finds That U.S Grandmaster 'L'ikely Cheated' More Than 100 Times." `https://www.wsj.com/articles/chess-cheating-hans-niemann-report-magnus-carlsen-11664911524`, 2022. Date Accessed: 23 November 2022.

[26] C. Beam, "A Cheating Scandal Has Rocked the Chess World. The 'Chess Detective' Is on the Case." `https://time.com/6227677/magnus-carlsen-hans-niemann-kenneth-regan-chess-scandal/`, 2022. Date Accessed: 23 November 2022.

[27] Z. Qureshi, "pgn2data: A library that converts a chess pgn file into a tabulated CSV data set." `https://github.com/zq99/pgn2data`, February 2021. Date Accessed: 1 March 2023.

[28] D. J. Barnes, "pgn-extract: A Portable Game Notation PGN Manipulator for Chess Games." https://www.cs.kent.ac.uk/people/staff/djb/pgn-extract/. Date Accessed: 1 March 2023.

[29] Dask Development Team, "Dask: Library for dynamic task scheduling." https://dask.org, 2016. Date Accessed: 1 March 2023.

[30] pandas Development Team, "pandas-dev/pandas: Pandas." https://doi.org/10.5281/zenodo.7549438, January 2023.

[31] M. Costalba, "scoutfish: Chess Query Engine." https://github.com/mcostalba/scoutfish, December 2016. Date Accessed: 1 March 2023.

[32] Lichess, "Chess rating systems." https://lichess.org/page/rating-systems. Date Accessed: 1 March 2023.

[33] FIDE, "Deloitte/FIDE Chess Rating Challenge." https://www.kaggle.com/competitions/ChessRatings2/overview, May 2011. Date Accessed: 1 March 2023.

[34] L. Le Cam, The central limit theorem around 1935 *Statistical science*, pp. 78–91, 1986.

[35] A. Matanović, M. Molorović, and A. Božić, Classification of chess openings 1971.

Appendices

A Additional Outputs

Figure 14: Effect of Higher Rating on White Win Rate on Lichess in 2022

```
Results of All Games:

1-0         19938052
0-1         18450502
1/2-1/2      1733174
Name: Result, dtype: int64

Out of 40121728 games:
  White won 19938052 games (49.69%).
  White lost 18450502 games (45.99%).


Results of Games When White Had a Higher Rating:

1-0         10662552
0-1          8242031
1/2-1/2       846503
Name: Result, dtype: int64

Out of 19751086 games when White had a higher rating:
  White won 10662552 games (53.98%).
  White lost 8242031 games (41.73%).
```

B Additional Plots

Figure 15: Distribution of Absolute Rating Difference on Lichess in 2022

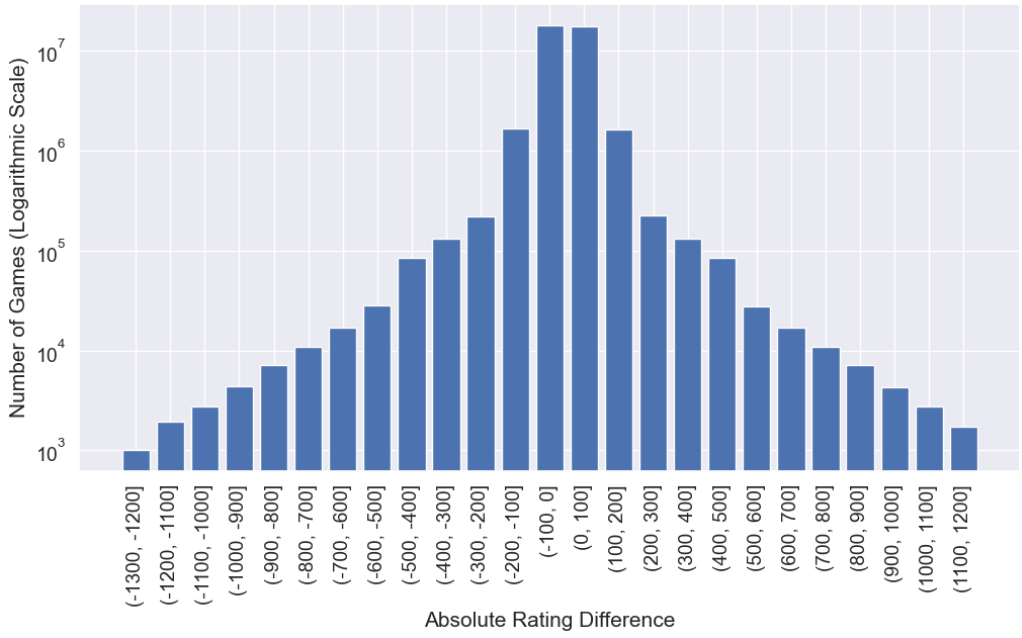


Figure 16: Distribution of Relative Rating Difference on Lichess in 2022

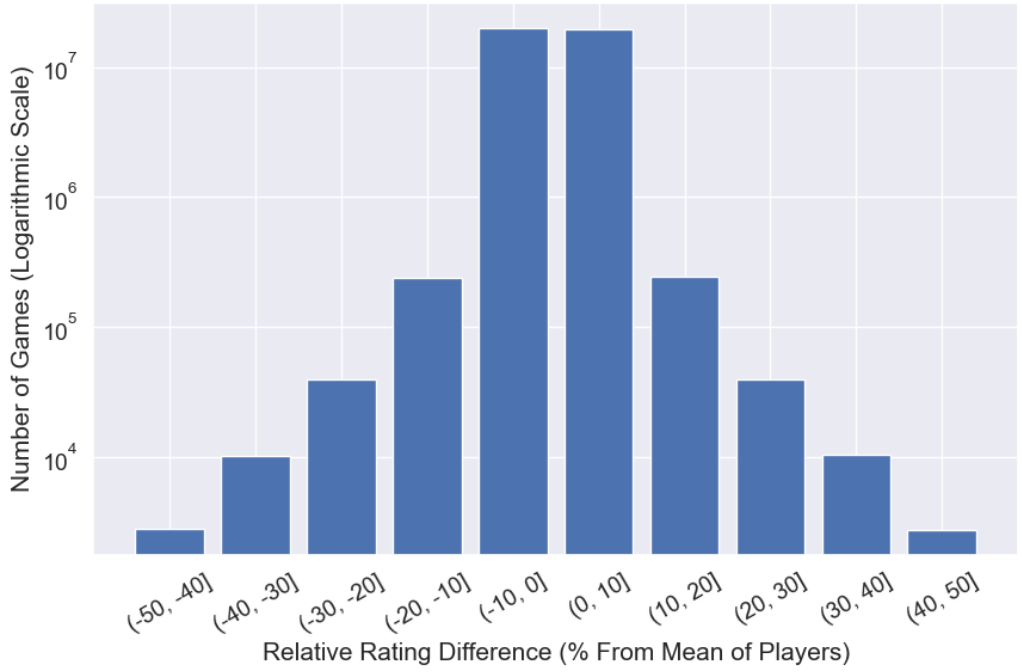


Figure 17: Mean White Win Rate for Each EloDiff Bin

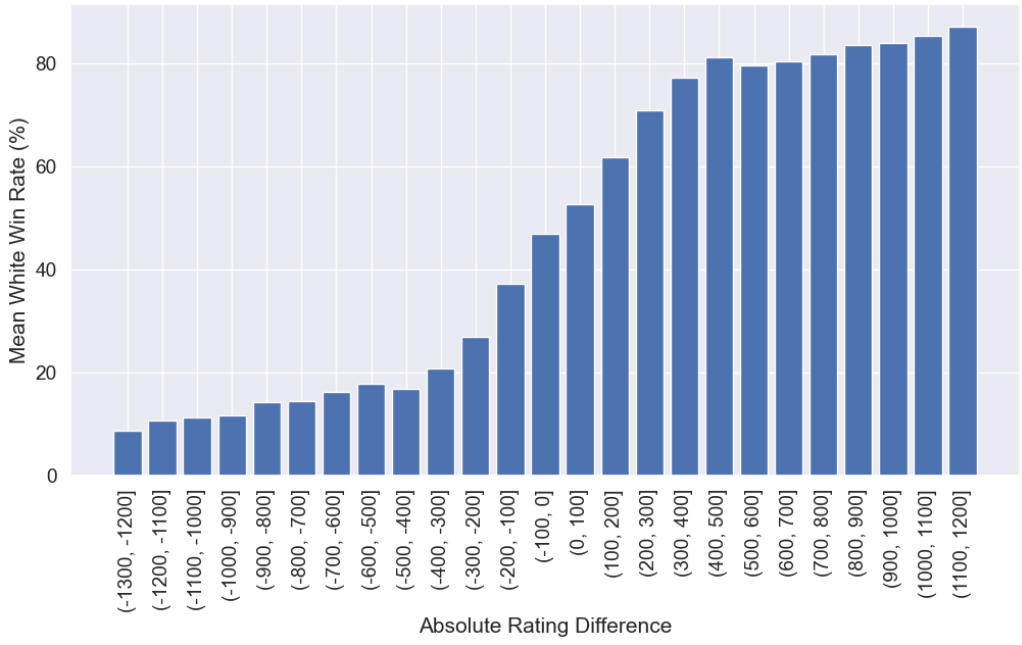


Figure 18: Most Popular Openings by Category on Lichess in 2022 (Rated 2000 and Above)

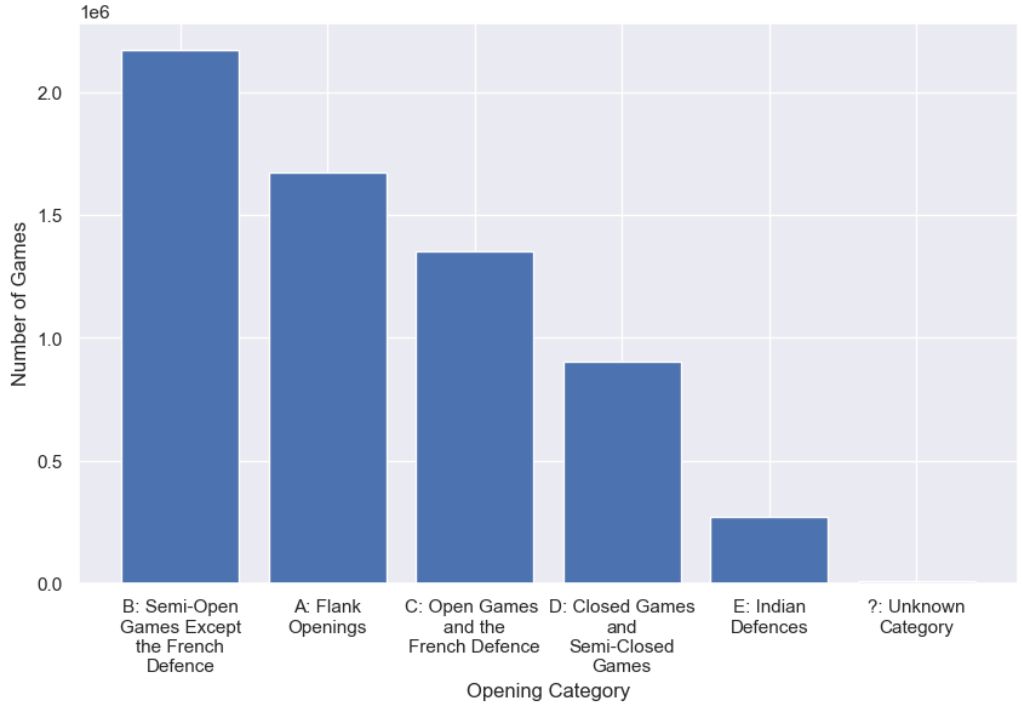


Figure 19: Most Popular Base Openings on Lichess in 2022 (Rated 2000 and Above)

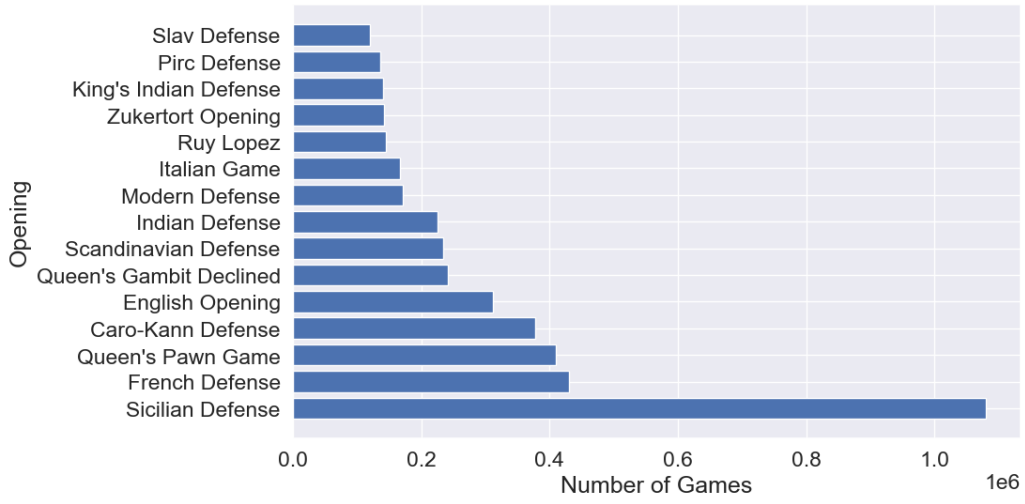
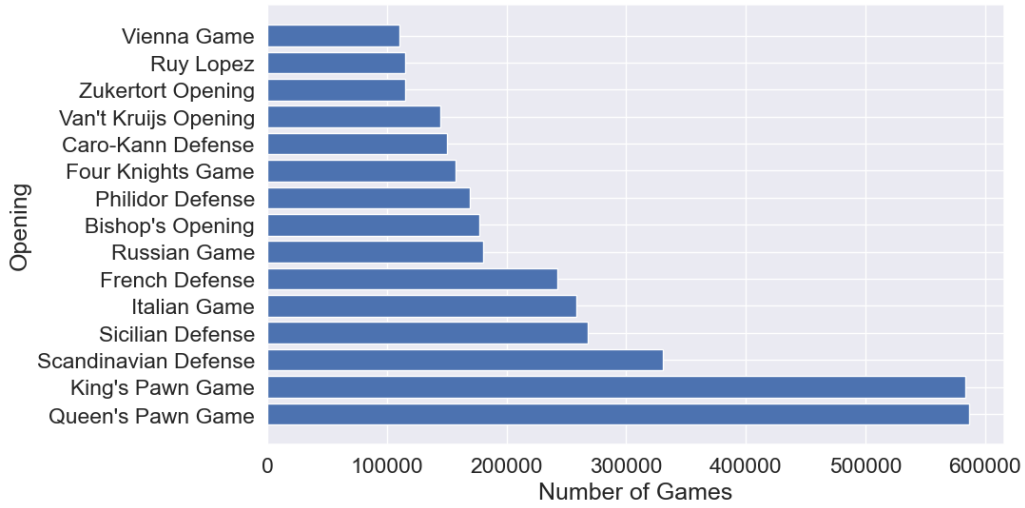


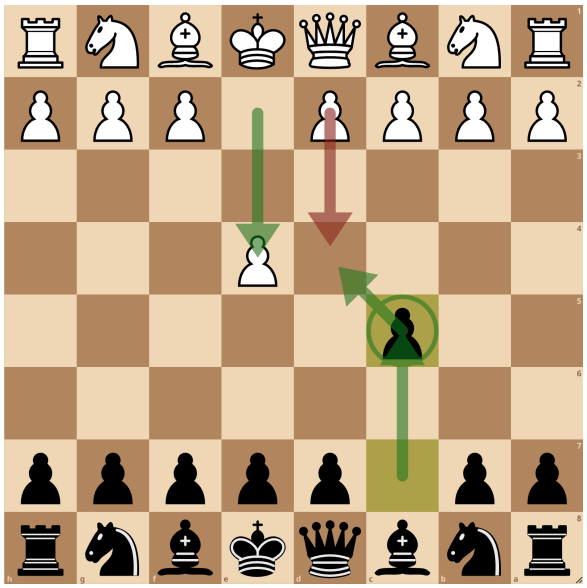
Figure 20: Most Popular Base Openings on Lichess in 2022 (Rated 1200 and Below)



C Examples

Figure 21: Examples of Chess Openings

(a) Sicilian Defence



(b) Queen's Gambit