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: _____

1 Introduction

2 Project Specification

3 Design

3.1 Downloading the Data

Collecting the data correctly was paramount to the success of my project, and it was important to use a large sample size to ensure that our insights represent the general population of chess games. I used the Lichess Open Database [1] of standard rated games for my data source – they upload tens of millions of games every month in PGN format, and they are easily accessible to the public. I 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. I will only be analysing Rated Bullet, Rated Blitz, and Rated Rapid games – I 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.

I 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, I aimed to collect a sample of approximately 5 million chess games from each month, so I took 6 million games from each month as this included Rated Classical games that I would later filter out.

3.2 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 [2] 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. I 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, I wrote my own Python script to convert each PGN file into a single CSV file containing only the metadata of each game. I used the python-chess library to parse the PGN files and iterate over each game. I skipped Rated Classical games and games that involved a non-human player. To optimise performance, I split each month's PGN into six files each containing 1 million games with pgn-extract (a PGN manipulator written in C) [3], and I used Python's multiprocessing library to process each month in parallel – this reduced the processing time for each month from 120 minutes to 20 minutes.

Once I had a CSV for each month, I combined them into a single CSV file containing my final data set. I converted this CSV to a folder of Parquet files using Dask [4], which is a library for parallel computing in Python and extends common interfaces like pandas [5] to handle big data in a larger-than-memory environment. This significantly reduced the load time for the data, and it meant I could use the Dask library to perform parallel computations on the data.

For low-level, turn-based analysis of games, I created a .scout file using Scoutfish (a tool for querying chess databases written with C++) [6] to enable fast queries. For example, I 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 I 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, I will primarily focus on high-level analysis of game metadata using Dask DataFrames.

Figure 1: Diagram of the Data Pipeline

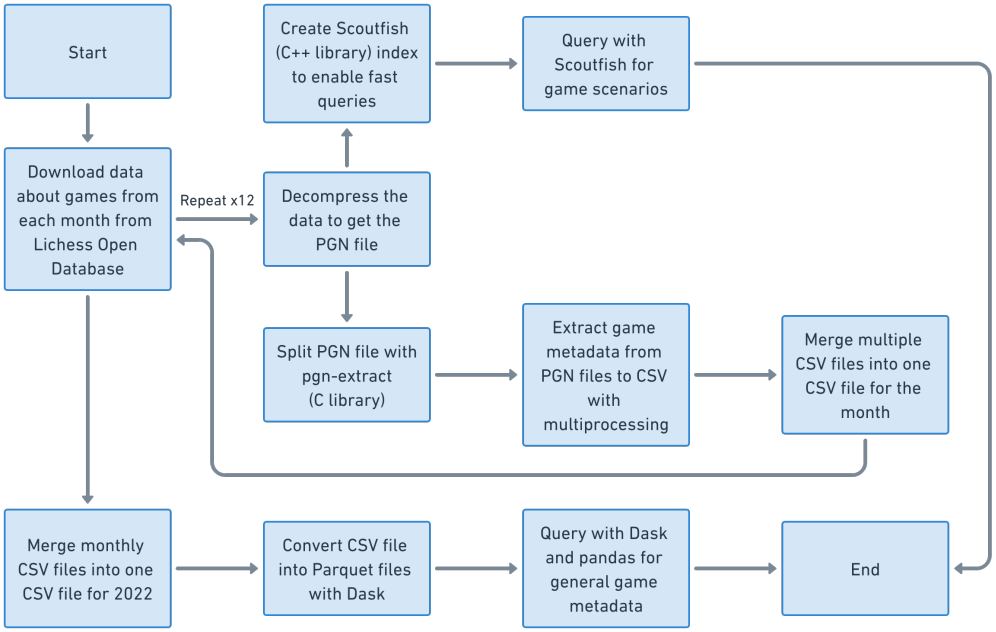
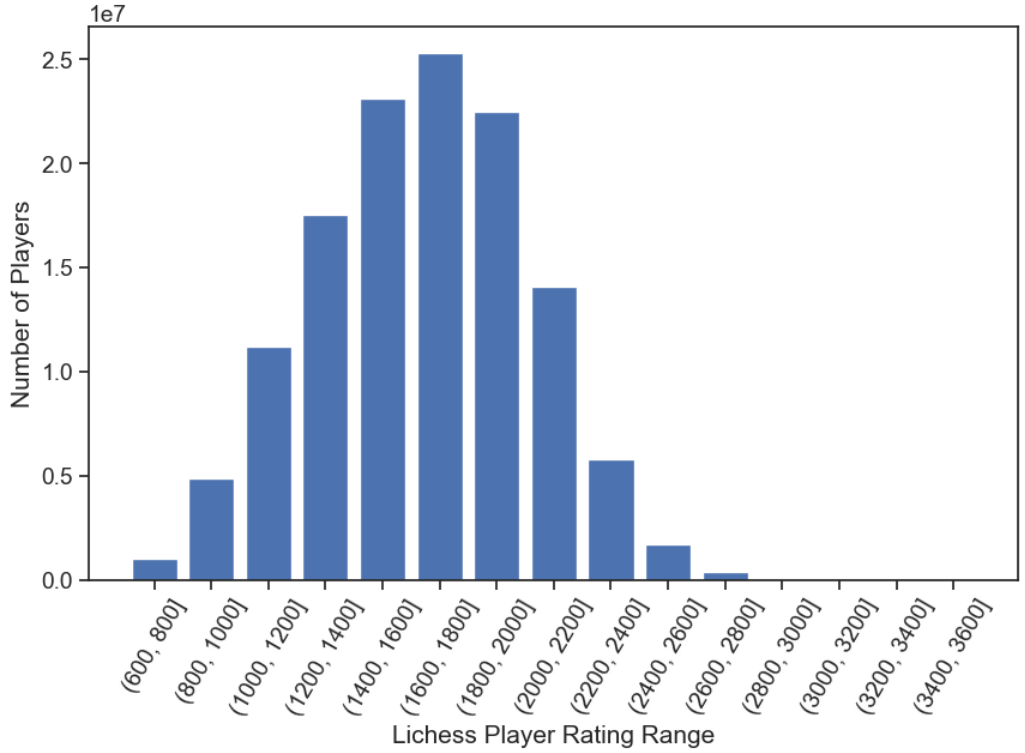


Figure 1 shows an overview of my data pipeline. It shows how I used the Lichess Open Database to collect the data over 12 months. On one hand, I aggregated it into a single CSV file, and converted it to a folder of Parquet files to query with Dask to analyse game metadata. On the other hand, I used this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

4 Analysis of Data

4.1 Initial Data Exploration

Figure 2: Distribution of Player Ratings



To start with, I counted the ratings of players in my 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 [7, 8]. The Glicko 2 rating system starts at 1500, and it is artificially limited to a minimum of 600 in Lichess. I 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 73,395, 7,295, 459, and 1 player(s) respectively. 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 2 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 [9]. 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 dsitribution 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.

Figure 3: Most Popular Openings by Category

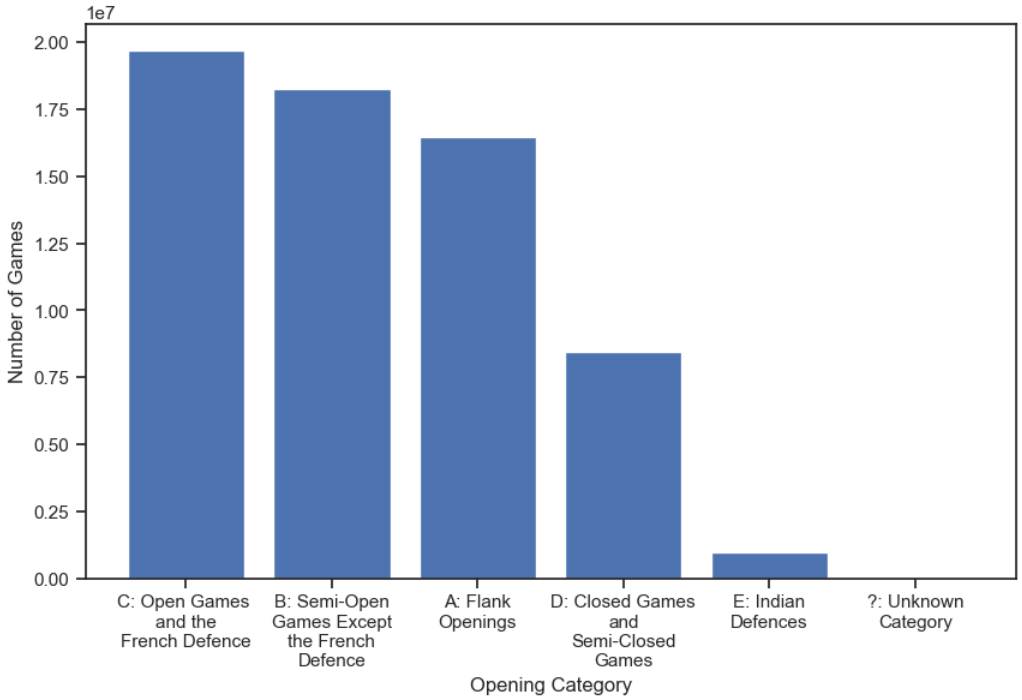
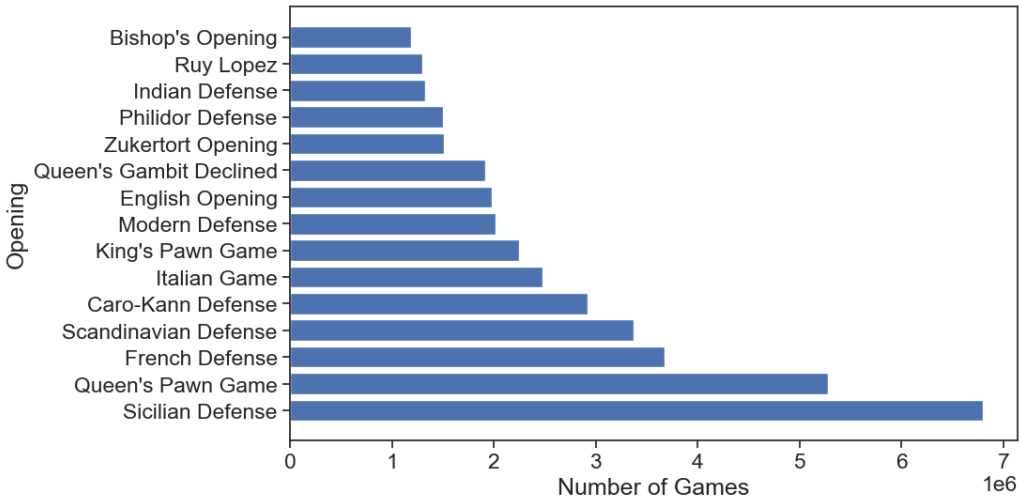


Figure 4: Most Popular Openings



5 Conclusion

5.1 Limitations and Future Work

References

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

[2] 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.

[3] 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.

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

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

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

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

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

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