# Table of contents

# Table of Figures

# Table of Tables

# Abstract

Here I will outline the actions performed in this thesis along with a brief description of the topic and results.

# Chapter 1: Introduction

This section will define the problem, the background behind it, and the expected outcome as a result of the analytics performed

# Chapter 2: Research Design

The following section will detail the process of identifying a subject of research in this report and detail the specific objectives that were derived to address the research problem. The research objectives are broken down by the tasks required to address the problem and the gain that they will provide with respect to the research. Alongside this, the identification of a source of primary data is detailed which describes both the method of sourcing through experimentation and the specific use case for the data once established.

## Primary Data

Primary data for the problem set identified in this research will be established through an experimental process of the application of two of the top chess engines Stockfish and Leela Chess Zero to a collection of defined solution *Mate in X* puzzles. By running the engines on these puzzles, parameters for time of search and depth of search will be collected and statistically compared to evaluate any possible power gain in the selection of one engine over another. Furthermore, the engine with higher established capabilities will be incorporated into the development of a recommendation model in the form of developing a new metric for ranking openings by value gain rather than win rate. With this new metric established a comparative analysis will determine the feasibility of using engine-derived information for practical use outside of a game specific analysis scenario.

## Problem Identification and Clarification

Chess, a game that requires players to employ both their knowledge of tactics and to derive their assessment of positions, requires the implementation of the analysis of data throughout every game. Originally, chess games like many problem sets, required a thought-driven analysis on a human level. But chess analytics and many other fields have changed significantly with the implementation of machine learning with the development of chess engines.

Since the inception of chess algorithms in the 1940s by Claude E. Shannon (Shannon, 1950) and Alan Turing (Turing, 1953), they have changed the approach to the game with interest from both players and mathematicians. Algorithms devised for chess engines come in many formats, such as human-designed, Alpha-Beta-Pruning based engines like Stockfish (Stockfish, n.d.) and Neural Netowrk designed, Monte-Carlo Tree search engines like Leela Chess Zero (LeelaChessZero, 2023). This diversity in engine design offers different approaches to the assessment of a chess position. It creates natural competition to find the most potent position evaluation methodologies, which can be seen in the Top Chess Engine Championship, with both Stockfish and Leela Chess Zero having won the title of champion within the last decade (Wiki, 2023).

With chess engines being available to players for post-game and specific position analysis, the goal of this project is to expand the use-case of the data analytics tools known as chess engines to provide players with recommendations of openings to aid in the expansion of their repertoire. By using a chess engine to study openings and how a game that uses that opening progresses, an alternative evaluation metric could provide a better ranking basis for openings when compared to the overall player success rate.

The process of determining the feasibility of this must consider the following objectives: determining the engine which is most capable of assessing a position given limited computational ability; understanding the potential difference in the analysis of openings when compared to real-world outcomes; the uncovering of potential opening selection dynamics based on the scaled rating of players within chess datasets; and the evaluation and justification of recommendation differences identified by establishing alternative valuation metrics.

## Research Objectives

### Chess Engines and the Horizon Effect

Chess engines incorporate different algorithms for the determination of the optimal line of play in a given position. Stockfish incorporates an Alpha-Beta Pruning search algorithm that uses human-trained heuristics and weightings to determine the value of search paths and the cutoff points for the pruning of its search nodes. Leela Chess Zero uses an alternative search method known as the Monte-Carlo Tree search, which randomly iterates through more linear search nodes to reach an optimal solution. Unlike Stockfish, Leela Chess Zero is trained through a neural network, which through a series of games played against itself, developed its heuristics and weights to evaluate each search node.

The horizon effect is an obstacle for search algorithms where a solution to a problem lies beyond the depth restriction of a search algorithm, such as the depth limit option in Universal Chess Interface compatible engines like Stockfish and Leela Chess Zero.

The objective here is to evaluate and justify any measurable difference in the parameters required by engines with different problem-solving approaches in finding the pre-determined solution based on end-game puzzles.

## Chess Engine Evaluation and Real-World Outcomes

Chess Openings are defined by a classification based on codes from the Encyclopaedia of Chess Openings (ECO), (Matanovic, 1978), which can be further subdivided into sections: Opening, Line, and Variation. The Opening is the initial moves played, the Line is the approach taken after these moves, and the Variation is a slight difference to change the overall tactical approach of the opening.

A famous example is the “Queen’s Gambit”, and its lines and variations are classified under the umbrella of code D06 and its variations under codes D07-D69. Each opening has a use case depending on the player's desired approach to the game concerning time control and even knowledge of the opponent and their repertoire, but this is more so the case in classical over-the-board games.

Each opening has various success rates with different players. This is due to the opponent being able to vary the game significantly with every possible move after an opening is played hugely increasing the number of permutations a board can take. Given a chess engine’s ability to search both wider and deeper than a human, the purpose of this analysis is to derive insight into the possible benefits of a chess engine analysis can be considered concerning the real-world outcome of chess games.

## Player Rating and Opening Move Distributions

With a large variety of standardised chess openings being available to everyone who plays a game of chess, players can be as selective as they wish regarding how they approach the initial stage of a game. Considering behaviours that are expected at different skill levels, a reasonable assumption would be that highly skilled players would be rather selective of their openings and maintain a lower diversity in their selection. By using ecological diversity metrics, statistical methods, and machine learning applied to real-world data, this section aims to establish the possible rating divisions within the chess player population at which selection approaches of openings differ.

Furthermore, this exploration will aim to provide insight into the pools of openings selected by players. By establishing a possible pattern between a player at a particular skill level and the openings the choose from, information may be further incorporated into new areas of research such as the development of a recommendation model to help inform players of openings that would be suitable for their play style or historical use.

## Recommendation Model

## Validity Type

## Ethical Considerations

Due to the nature of this project relying on open-sourced and free to use chess engines in combination with open-licensed data, there are no ethical considerations with regard to fair usage. Both Stockfish and Leela use a GPLv3 license, meaning general use is under no restrictions and that any adjustments to source code must be documented and the original source must be packaged with the adjusted program, attribution is also necessary. Data from Lichess's database is under Creative Commons 0 (CC0), which allows for full access and usage without any attribution, but their website notes that it is appreciated. The use of data from Lichess is also supported and encouraged by the company.

Ethics regarding personal data are not of concern due to there being no personal identifying data collected in the PGN format from chess games except for usernames which belong to Lichess, which they are free to distribute.

Results from this project are obtained from using open-sourced engines on free access data. From this, all results will not be of a restricted nature and will have no ethical concerns regarding publishing.

Consideration must be given to the establishment of power gain from a chess engine given a specific subset of puzzles for testing. It is to be understood that this report does not imply an objective superiority of one engine over another for all uses. Readers can evaluate engines based on results from chess engine-specific comparative sources such as Top Chess Engine Champion (Wiki, 2023), which hosts competitions for engine developers to submit their engines for both tournament and league head-to-head comparisons.

# Chapter 3: Literature Review

# Chapter 4: Methodology

## 4.1 Data Collection and Pre-Processing

All data for this project was sourced from Lichess.org. Lichess provides a database of all games played on the website on a monthly basis and makes them available for use under the Creative Commons 0 License. Along with this they also publish all their puzzles along with games played.

The games are available as Portable Game Notations (PGN) files. These can be read as text documents and contain key information about games such as player names, their ratings, the result, time-control, and the moves played in the game.

PGN as a database format is standardised for Universal Chess Interface (UCI) software but does not allow for easy importation into analytics tools such as Excel or in this case Python, specifically Pandas. To account for this, a program with a specific function was developed to process a PGN format database of 200 plus Gigabytes of data into a usable CSV format.

The key problem to solve with this was handling the massive amounts of data due to be processed. To circumvent this issue, the text file was read in line by line where the indicator of information piece for each game would allow it to be assigned a position in a dictionary. This dictionary was then continually appended to itself for every new game and once enough games had been processed, was converted to a format that could be deposited in a Comma Separated Values (CSV) file at regular intervals in order to avoid the over usage of temporary memory and to allow for a continuation given a spontaneous failure as the process had to run for several days.

Once a sufficient number of games had been reached, in this case 10 million the function was interrupted, and the data was available for use. The number of games was determined by a combination of processing power, temporary memory available for importing of data, and in particular time limitations. Due to the nature of processing a large text file and being unable to import into temporary memory because of storage limits, the line by line approach allowed for consistent processing at a regular pace reducing the strain of processing power and memory availability.

Although there were tools available for public use on GitHub that could allow for imports into memory for processing, their splits were based on memory size rather than by games which may have split games in two resulting in their data being unavailable for processing. As this selective exclusion of games may have resulted in bias within the aims of this research, this method was determined to be less desirable to the text file processing method.

## 4.2 Puzzle Analysis

A database of puzzles was obtained from Lichess Puzzle Database (Lichess, 2023). This is an open database published by Lichess to provide its users with access to all the puzzles they use on their site. It also includes summary statistics for the puzzles.

The Comma Separated Values file provided by Lichess.org was imported into a Python environment using Pandas where it was filtered to extract all the puzzles fitting a ‘Mate in X’ description. This provided a variety of different puzzles on which an analysis could be performed to establish the difference between engine horizon problem-solving capabilities.

To provide sufficient testing for the engines, a range of mate lengths was chosen from the longest available. The database consisted of a range of ‘Mate in X’ problems ranging from 1 move to 10 moves. Five was chosen as the lowest value to provide a depth where the engines would need to execute a minimal search and 8 was chosen as the upper limit to allow for enough comparison in distribution for the selected engine parameters. Both populations for ‘Mate in 9’ and ‘Mate in 10’ puzzles would not have provided a large enough sample to compare distributions, both having 9 and 2 puzzles respectively, whereas puzzles with 5, 6, 7, and 8 moves have 3058, 2320, 503, and 183 puzzles in their samples.

A Python program using the Python-Chess library (Python-Chess, 2023) was developed to implement an experimental procedure to test both Stockfish and Leela Chess Zero, two of the top open-source engines. In this program, a custom function was developed to utilise the python-chess Universal Chess Interface (UCI) *analyse* function in an open-ended analysis of each puzzle. This analysis allowed the engines to access the puzzle and perform their search until a mate with the expected X value was found. Once found the function would cease its execution and return the parameters the engine had used to find its solution to the ‘Mate in X’ puzzle.

This approach was found to have more granularity for analytics as opposed to the Python-Chess engine limit protocol. By providing defined limits for depth of search and time searching, the engine would be tested multiple times to see at which combination of parameters it could determine a solution. The results of this process showed multiple combinations of depth and time that provided solutions for each problem but did not allow the engine to approach the problem in its base state. The change to an indefinite analysis pending a solution allowed each tested engine to approach the puzzle and return an exact combination without intervention by the tester.

The results from each search by the engine were returned as an information dictionary as dictated by each of their UCI protocols. Although these Python dictionaries had different formatting, they both contained the parameters of interest for this experiment: *time*, *seldepth*, and *nodes*. The *time* is defined as the time until a solution was reached, *seldepth* is defined as the depth at which the accepted solution was selected from, and *nodes* are the number of branches searched by an engine in its approach to finding a solution.

Once all the above parameters were extracted from the engines during the analysis process, each distribution was compared to its corresponding parameters for each categorical ‘Mate in X’ group to provide a one-to-one experiment.

All statistical tests performed were tested with an alpha = 0.05. This is an established value proposed by Fisher as a means to suggest a cutoff point where confidence could be attributed to a result (Fisher, 1954). This value is used conventionally in many types of research and is generally considered that 95% is an acceptable confidence level (Thiese & Ronna, 2016).

As distributions were being compared, the correct statistical tests had to be selected to ensure an accurate comparison of data was being performed. The choice of test was primarily driven by whether the data was likely to conform to a normal distribution or not, which required its own statistical test. The Shapiro-Wilk test was used to assess whether the Null Hypothesis that each of the distributions for the individual variables split by category was normally distributed. As both *seldepth* and *nodes* are discrete values, non-parametric tests would be the choice for them.

The Shapiro-Wilk test tests for normality in distributions by comparing the assumption that the distribution would adhere to a normal curve and generating a test statistic based on the deviation from normality (SHAPIRO & WILK, 1965). The hypothesis for the test in this case was for *time* distributions being normal given a P-value of 0.05. All results for this test showed that the distributions were not normally distributed meaning the comparative tests would need to be non-parametric.

The Mann-Whitney U test is a non-parametric comparative test that is considered as powerful as the parametric t-test (MacFarland & Yates, 2016) which is one of the most common tests used in research journals (Yim, et al., 2010). Due to its usefulness as a comparative test, it was implemented into this analysis to compare equivalent engine parameters between engines within their respective ‘Mate in X’ categories.

The results of the Mann-Whitney U test were checked to determine whether the distributions were similar, and boxplots were used to plot the display the parameter values for visual comparison of quartiles and median values. The values for *seldepth* were plotted with a normal range y-axis of values and the *time* and *nodes* parameters were plotted using a Log range y-axis to allow the data to be read with granularity with the skewed nature of the values.

## 4.3 Player Opening Distribution Analysis

This objective aimed to investigate and determine the relationship between players, their ratings, their diversity in the selection of openings, and the pools of openings they may be more inclined to select from. Given the sample of 10 million games obtained from Lichess, a pre-processing procedure was required to refine the sample of games into a dataset that is appropriate for the investigations outlined for this section. The particular filters chosen to refine the dataset were based on the topic of appropriate openings, frequency of opening use by players, and volume of games played by players in order for there to be sufficient data to base estimations on.

Appropriate openings were chosen by the complexity required to analyse the collection of openings. Using the naming convention of Opening, Line, and Variations for differentiating openings, there were 3,398 distinct openings. With the Variation inclusive openings being used to a lesser degree by the entire population and being highly dependent on the response by black, they were excluded to reduce granularity within the dataset and provide a more focussed analysis resulting in 1,478. Alongside this, black openings when considered were dependent and the initial moves played by white. As this would have required a sub-analysis that generalised this objective to a degree that would require a much lengthier and more complex research procedure, they were excluded to allow this research to focus on openings that could be chosen rather than required by a player, resulting in 423 openings remaining.

The refinement of openings used was then further applied by the assessment of the minimum volume of occurrence for an appropriate analysis. Given that sampling at 10% of the total population would be required for the engine analysis of games within this research project, only openings that were played one hundred times or more would be assessed. This had the positive effect of reducing the input of infrequently played games into the clustering methods that were employed later.

The third filter applied was for openings that were not used by 10 or more unique players would not be considered. This requirement was established by testing of the diversity metrics within the data. When openings were played by less than 10 players, it resulted in many fixed values of the diversity metrics, which skewed early attempts at clustering.

The final filter for the dataset of games was based on the volume and diversity of games played by users. In order to consider player diversity similarly to the opening diversity, a minimum limit of having played 10 games within the first 10 million 10+ minute games played in June 2023 on Lichess was required to avoid the diversity index spikes. Along with this, to ensure association suitability players had to have used two or more openings. Without this requirement, the clustering efforts would have been skewed by players that only ever stuck to one opening, which were consistently 0 for these players who were out of scope for the primary research objective.

With the population of games being finalised based on the filters for both openings and players, analyses were performed to determine both the diversity in opening choice by players based on rating and to determine the points at which players chose from different pools of openings within the entire available population.

#### Opening Diversity

The approach in this section was to use a combination of diversity metrics based on providing sufficient information to allow for a balanced approach. The particular non-parametric diversity measures chosen for this objective's assessment were the Gini Impurity index and Shannon Diversity Index. A research paper in XYZ determined these indexes to be suitable in providing measurements in different ways that account for variety in occurrence and balance in occurrence respectively (Stirling, 2007).

With both of these calculated for each player and their use of each opening within the dataset, the data was brought forward for machine learning analysis. The K-Means Clustering method provides a suitable algorithm for the assessment of these items due to its properties of simplicity in implementation and scalability with large data (Ning & Liu, 2022) which is a concern given the population of unique players being 110,662.

By combining player rating, Gini diversity and Shannon diversity indexes as input into the clustering model, a process of determining optimal clusters was performed by assessing multiple cluster values denoted as K. K being the input integer value, in this case of a range of 1-10, was tested as the cluster value and assessed using a combination of the “Elbow Method” and the “Silhouette Method”. The “Elbow Method” is one of the oldest approaches for determining K within K-Means clustering but has the disadvantage of being a subjective measure by the user (Shi, et al., 2021). To counter this, the secondary measure of the “Silhouette Method was incorporated as it has the advantage of being non-resource intensive and having simple evaluation metrics (Dudek, 2020).

#### Player Opening Selection

After assessing the results of this clustering procedure, a contingency table was used to further investigate the differences between different rating groups and their selection of openings. By using a simple binning procedure to determine 3 even splits as a form of testing, a chi-squared test for a contingency table on the bins crossed with all opening use showed statistical significance in the difference between rating groups and their opening use. This implied that there was rationale for the clustering analysis of opening use and player rating.

In order to cluster on the categorical opening variable, the dataset was transformed into *Count Encoded* format data*. Count Encoding* is a method similar to one-hot encoding that retains information about the volume of occurrences within the dataset. With this transformed dataset, the data could be fed into a clustering method as previously performed in the diversity analysis section of this method. The value of K was interpreted based on the combined output of the *Elbow Method* and the *Silhouette Method* and distinct clusters were obtained from the final implementation of the clustering algorithm.

## 4.4 Opening Analysis

The objective of analysing openings using chess engines was driven by the computational advantage of depth of search in comparison to typical human assessment of a chess position. By taking the engine evaluation of the starting point of an opening and comparing it at regular intervals in multiple games, the general trend of game results can be measured. The advantage to be determine by this possible approach is the negation of critical blunders or time-control having a detrimental effect on the outcome of a game.

The first consideration for this procedural analysis of games was the engine to be used. After the puzzle analysis it was found that Stockfish was the optimal engine for a combination of time required, depth of search possible, depth of search required to find optimal pathways, and nodes search through its algorithms.

The output filtered game dataset from the player opening distribution analysis was used as the key dataset for understanding games as it contained the population of games and players that are for consideration for further analysis of a recommendation model. Containing 2,777,554 games, any meaningful analysis from an engine would take months given a fair evaluation time of 1 second or more. Early tests showed selected search depths of analysis between 9 and 12 when time was limited to 0.01 seconds in order to analyse more games. This resulted in standard deviations that negated any value from the mean engine parameters collected, with games of the same position also having different evaluations.

To counter this computational limit, a stratified sample of 1% of the dataset resulting in 27,776 games was implemented, stratifying on the openings used and a win/not-win variable to ensure proportions of success were considered within the sample analysis. This sampling allowed a scaled up increased time analysis which resulted in higher search depth, allowing for a more even assessment of the positions. Another key restriction implemented was the restriction of access to a hash table. Stockfish, once opened, can hold onto previous analyses, and use these as information on whether a position is strong or not. In order to allow for a fair assessment of every position independently, the hash table was cleared between each game processed by the engine.

Once the engine limit for time was established, a program set up to continually analyse for multiple days had to be established. To avoid loss in case of sudden hardware or software failure, the program was designed to process the dataset in chunks and iteratively write the results to a CSV file. By performing the analysis with this method, computational resources were freed to help counter possibilities of temporary memory shortage and allow the engine run smoothly.

Post completion of the engine analysis, the data was transformed such that each opening contained mean and median evaluation metrics, in this case centipawn value, the depth of search at which the evaluation was selected and the nodes required in the search.

## 4.5 Recommendation Model

Section 2 - Analysis of Openings

Excluded those that ECO code not joined on as Lichess ads some descriptions to non-descript openings

Chapter 5: Implementation

# Chapter 6: Results

## 6.1 Puzzle Analysis

Table - Result of Engines Solving Mate in X Problems

|  |  |  |
| --- | --- | --- |
| **Mate Puzzle Solved?** | **Stockfish** | **Leela Chess Zero** |
| Yes | 3046 | 3023 |
| No | 1 | 24 |

Here Stockfish shows an indication that it has a better overall ability to overcome search horizons more effectively by solving all but one of the Mate in X puzzles provided to it. Leela Chess Zero also performed well given the task, solving all but 0.8% of the puzzles, with one of the unsolved being the same as the one Stockfish did not manage to solve.

Table - Result of Shapiro-Wilk test for Stockfish and Leela Chess Zero Evaluation *time*

|  |  |  |
| --- | --- | --- |
| **Mate in X** | **Stockfish (Statistic, P-Value)** | **Leela Chess Zero**  **(Statistic, P-Value)** |
| 5 | (0.06541, <0.001) | (0.24242, <0.001) |
| 6 | (0.13197, <0.001) | (0.28906, <0.001) |
| 7 | (0.37938, <0.001) | (0.35686, <0.001) |
| 8 | (0.59305, <0.001) | (0.40843, <0.001) |

Table - Result of Whitney-Mann U test for Parameter Comparison

|  |  |  |  |
| --- | --- | --- | --- |
| **Mate in X** | **Time**  **Stockfish < Leela Chess Zero (Statistic, P-Value)** | **Seldepth**  **Stockfish < Leela Chess Zero**  **(Statistic, P-Value)** | **Nodes**  **Stockfish > Leela Chess Zero**  **(Statistic, P-Value)** |
| 5 | (171289, <0.001) | (1714897, <0.001) | (4305251, <0.001) |
| 6 | (19794.5, <0.001) | (89463.5, <0.001) | (195024.5, <0.001) |
| 7 | (7321, <0.001) | (12230.5, <0.001) | (27207.5, <0.001) |
| 8 | (299, <0.001) | (490, 0.002) | (1135, <0.001) |

The Shapiro-Wilk test uses the Null Hypothesis that a distribution follows a normal distribution. The results of the Shapiro-Wilk test displayed in Table 1 for Stockfish parameter normality shows a consistent result of P-value < 0.05 wherein the Null Hypothesis can be rejected such that the data is considered non-normal. Non-normal distributions are seen in Table 2, where a P-value < 0.05 indicates the distribution of *time* values obtained from the engine analysis by Leela Chess Zero can be compared with Stockfish *time* values using a non-parametric test.

With the results seen in Table 2 indicating non-normal distribution of values, the Mann-Whitney U test was used. The results can be seen in Table 3 where a collection of test statistics and P-values are displayed for each engine parameter collected between the two chess engines for each of the categorical ‘Mate in X’ values. In these results it can be seen that Stockfish outperforms Leela Chess Zero in terms of a reduction lower time needed to solve, a lower overall depth of search required to come to a ‘Mate’ conclusions and a higher volume of nodes searched within the time of analysis. The *time* and *seldepth* are indicative of higher performance with respect to search horizon problems, whereas the *nodes* difference is expected given the different search structure of Alpha-Beta Pruning algorithms versus Monte-Carlo Tree Searches.

The combination of a both higher solve rate and more efficient search parameters indicates that Stockfish has an overall better performance with respect to the Horizon Effect, making it the choice engine for this project in analysing openings and their effect on a chess game.

## 6.2 Player Opening Distribution Analysis

# Chapter 7: Discussion

# Chapter 8: Conclusion

# Appendix A: Workflow

# Appendix C: Data Permissions

# Reference List