Predicting Chess Matches Based on Opening Moves & Rating

Final Project Report

Introduction

Chess is a game that has been played for centuries and is known for its complexity, requiring players to possess strategic planning, critical thinking, and a strong understanding of the game's rules and pieces. Due to its intricacy, chess has been an important application in the development of both early and modern machine learning.

Since the late 1990s, researchers and computer scientists have focused on creating chess engines and artificial intelligence to play the game. One of the most famous examples of this effort is IBM's Deep Blue chess computer. In 1996, Deep Blue made history by becoming the first computer to win a game against a reigning world champion, Garry Kasparov. The match was highly publicized and attracted worldwide attention, demonstrating the capabilities of artificial intelligence and machine learning.

The development of chess engines and AI has progressed significantly since the days of Deep Blue. Modern chess engines, such as Stockfish and AlphaZero, can play at a superhuman level, beating even the strongest human players. These engines use a combination of machine learning techniques, including deep neural networks and reinforcement learning, to analyze millions of chess positions and make the best move decisions.

The study of chess has also led to advancements in other areas of machine learning, such as computer vision and natural language processing. The game provides a useful testing ground for developing and testing new machine learning algorithms and techniques.
Chess is a game that always starts with the same initial board and piece setup. As a result, one of the critical elements of chess is the opening moves, which set the stage for the rest of the game. However, for novice players, choosing the right opening move can be challenging, and often leads to unfavorable outcomes.

Predicting the outcome of a chess game can serve different purposes depending on the individuals involved. Chess enthusiasts may find it entertaining to use predictive models to test their analytical skills and to gain insights into the playing strategies of top players. For coaches and trainers, predictive models can be useful in analyzing the strengths and weaknesses of their own players and those of their opponents, thus allowing them to create effective training plans and make strategic decisions during the game. Bettors may also use predictive models to make informed betting decisions, while researchers in the field of artificial intelligence and machine learning can develop and test new algorithms using chess as a valuable test bed.

For me, finding motivation to work on a project comes from working on something I truly enjoy. I struggled to find a dataset that aligned with my interests and skills. It wasn't until I stumbled upon a chess-related dataset that I realized I had found the perfect project for me.

Chess has always been one of my favorite games, and I find myself coming back to it year after year. It's a game that requires critical thinking, strategic planning, and a deep understanding of the rules and pieces. Analyzing chess games and predicting the outcome of matches has always been something that interests me. I believe that when you work on something you truly enjoy,

the quality of the work you produce is better, and you are more motivated to see the project through to completion.

As the sole member of this project, I took on the responsibility of completing all aspects of the work. This included writing the proposal, progress reports, and final report, as well as handling the data preprocessing and programming of the model myself.

Writing the proposal and progress reports required a great deal of organization and attention to detail. I had to ensure that I was accurately communicating my ideas, outlining the scope of the project, and providing updates on my progress. Additionally, I had to be able to communicate complex ideas and technical concepts in a clear and concise manner.

The data preprocessing and model programming aspect of the project required a great deal of technical skill and expertise. I had to transform the dataset to prepare it for modeling, and then implement the algorithms and techniques necessary to train and optimize the model. This required a deep understanding of programming languages and machine learning concepts.

Approach

The dataset I used for this project can be found on Kaggle here. It includes game data of 20,000+ chess matches on Lichess, a free-to-play chess server. It includes features such as turn count, player rating, move order, opening names, victory status, and the winner.

At the outset of my project, I planned to use Decision Trees (DT) as my primary approach. DTs are a popular type of supervised learning algorithm that can be used for classification and regression analysis. Essentially, DTs make predictions by asking a series of questions and using the available data to provide answers (Masters In Data Science: Decision Tree).

As I delved deeper into machine learning techniques for analyzing chess, I discovered that the evaluation score of many chess bots is determined by a decision tree that considers all possible moves. However, as I began to consider the complexity of the game and the vast number of variables involved, I realized that using a Decision Tree model on its own would not be sufficient.

That's when I began exploring alternative approaches and came across Random Forests. A Random Forest is essentially a collection of decision trees that combine their individual results to generate an overall prediction. This is typically achieved through a process of averaging or majority voting.

By using a Random Forest model, I was able to overcome the limitations of the Decision Tree approach and account for the complexities involved in predicting the outcome of a chess match. This allowed me to achieve more accurate and robust results, which ultimately helped me to gain a deeper understanding of the game and its underlying mechanics.

I encountered challenges while implementing random forests because I was struggling to preprocess my data. One of the features of my dataset was the move order, which was represented as a string of moves in Chess Standard Algebraic Notation (SAN). I faced difficulty in numerically encoding this feature because one-hot encoding would require a huge number of features, making it difficult to accurately represent the data. That's when I came up with the idea of representing the state of the board as a one-hot encoded tensor of 8 x 8 x 12. This was a great solution because it allowed me to capture the state of the board at any given move and avoid

the issue of different-sized move vectors. Also, it allowed me to check the accuracy of my model at any arbitrary number of moves less than the total number of moves.

In order to prepare my data for modeling, I knew that I needed to streamline the features to focus on what was most relevant to predicting the outcome of a chess match. I recognized that some features, such as player names and game ID, were not likely to have a significant impact on the result, and therefore decided to remove them from my dataset. Similarly, I found that the opening play and time increment features were not essential for the purpose of my model and could have potentially led to overfitting. However, I didn't stop at simply removing features; I also looked for ways to add new features that could enhance the accuracy of my model. One feature that I found particularly useful was the state of the board, which I represented as a one-hot encoded tensor of 8 x 8 x 12. By including this information, I was able to capture the state of the board at any given move, which was crucial for accurately predicting the outcome of a game.

Fortunately, I was lucky enough to obtain a clean dataset, which meant that there were no missing or incomplete values that required imputation. This made the data preprocessing phase relatively straightforward and efficient. To further prepare the dataset for modeling, I employed various techniques such as label encoding, which allowed me to convert categorical variables like the opening name, victory status, and the winner of the match into numerical representations. Additionally, I normalized the player ratings, which were often quite large, to ensure that they did not dominate the other features during model training. By normalizing the ratings, the data became more manageable for the model to interpret, and the resulting predictions were more accurate.

In a final attempt to incorporate the moves as a feature in my dataset, I experimented with logistic regression using one-hot encoded move vectors. This approach resulted in an overwhelming number of features, with over 16,000 one-hot encoded moves. Unfortunately, logistic regression took 6 hours to complete, and even then, it produced minimal results.

My final model for the chess outcome prediction was a convolutional neural network (CNN). Before deciding on this model, I conducted extensive research on various machine learning algorithms that could be applied to the task at hand. As I delved deeper into the realm of machine learning and its applications in chess, I discovered that a convolutional neural network would be the most effective choice for this project.

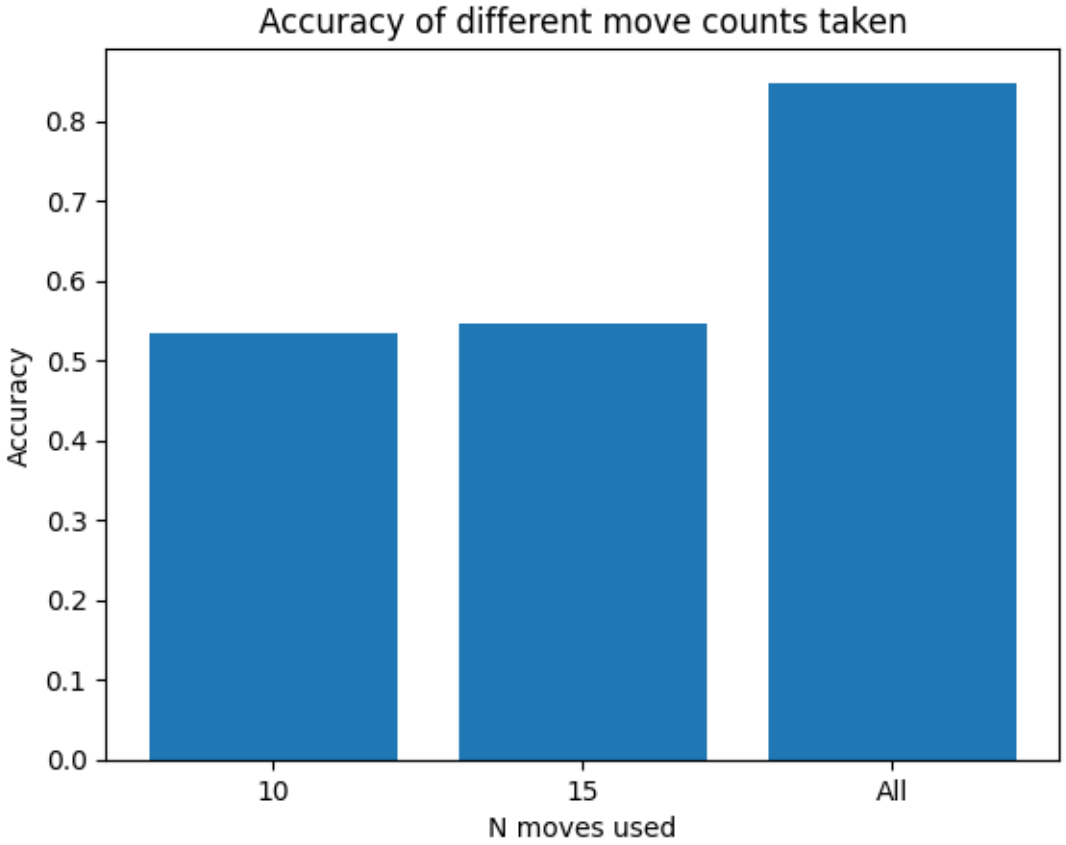
A convolutional neural network is a type of deep learning neural network that is widely used in computer vision tasks, such as image and video recognition. This neural network is designed to automatically and adaptively learn spatial hierarchies of features from input data by employing a series of convolutional layers. In essence, a CNN is a sophisticated system of layers that can recognize and extract features from patterns in an image.

In the context of this project, the CNN model was well-suited to the task of chess outcome prediction because of its ability to capture the spatial relationships between pieces on the chess board. The CNN architecture is designed to effectively extract features from an image, and the state tensor of the chess board can be thought of as an image with pieces in certain positions. By training the model on this data, the CNN could recognize and understand the patterns of piece movement and their spatial relationships, allowing it to accurately predict the outcome of a game.

Results

During the testing phase, I discovered that the accuracy of the convolutional neural network model was highly dependent on the state of the chessboard. The model showed impressive accuracy when considering the state of the board at the very end of the game, but its performance was rather subpar when it came to the first 10 and 15 moves. This was a crucial insight because it highlighted the importance of the late game in determining the outcome of a chess match.

These findings also indicate that the early game is more unpredictable and that the result of the match becomes more apparent as the game progresses. This is likely because the strategic choices that players make in the early game are far more diverse and less deterministic than the moves made towards the end of the game, which are often more predictable. The complexity of the opening moves and the freedom of options in the early stages of the game may make it difficult for the model to accurately predict the outcome of the match. However, the late game is typically more straightforward, with fewer possible moves available and a clearer path towards the end of the game. As a result, the convolutional neural network model performed much better in predicting the outcome of the match based on the state of the chessboard at this stage.



When the CNN was trained on data containing the first 10 and 15 moves of a game, the accuracy of the predictions was only marginally better than random guessing. This was not unexpected, as the early stages of a chess game are characterized by a high degree of uncertainty, with many potential outcomes still on the table.

However, when the CNN was trained on data containing the final state of the board before a player won, the model demonstrated an impressive average accuracy of 85 percent.

Conclusion

Throughout this project, I was able to acquire a deeper understanding of the supervised learning models, the importance of data preprocessing, and neural networks, especially convolutional neural networks. Although I had a lot of trial and error while implementing different models and preprocessing techniques, I was eventually able to develop an accurate prediction model using a convolutional neural network that accurately predicted the outcome of a chess game based on the state of the board, although it performed better in the latter stages of the game. This project gave me a great opportunity to explore and work with complex machine learning models and techniques and has been a valuable learning experience for me.

Acknowledgements

During my research on data preprocessing, I stumbled upon AlphaGo, a groundbreaking computer program that defeated a professional human Go player. One of the key features of AlphaGo was the use of a state tensor to represent the Go board. This inspired me to explore the possibility of applying the same technique to the game of chess, which ultimately proved successful in my project.

If you are interested in learning more about AlphaGo, you can check out the program's official website here.

Additionally, I found the Kaggle challenge of predicting the winner of a chess game after the first move to be another source of inspiration for my project. The challenge provided valuable insights into the complexities of predicting the outcome of a chess game and can be accessed on Kaggle.