Bradley Chess

Abraham Borg, Sarom Thin, Mehar Rekhi

California State University, Monterey Bay

CST 499 Final Directed Capstone

Project Advisor Sam Ogden

Fall B 2022

Executive Summary

Chess is a highly complex game. However, specific trends and playstyles become dominant and repeatedly appear in play throughout different periods. These trends are things like popular openings. There is a lot of data that can be gleaned from every chess game, and we can analyze this data and make predictions about the outcome of a game. The team will use Dash and Plotly to create a chess analysis dashboard web application. The dashboard web app will present an analysis of several million chess games. It will show charts and figures that describe the playstyle of grandmasters through the years, starting from the 1970s to the present day. The dashboard will also feature an interactive chess board that will have an indicator showing the best move.

The chess data will be used to train a model to predict the best move in response to an opponent's current move. This recommendation will be based on several parameters. For example, the program will consider the current pieces left on the chess board. This project will benefit the chess and data science enthusiasts and players at the Fresno Chess Club. These players will be able to analyze their games and other players' games at the club. The analysis is essential to players because it will allow them to offload the analysis to the computer instead of their minds. They will see the trends in their playstyle and use that information to improve their chess skills.

Table of Contents

Part I

Introduction

Project Description

The name of the project is BradleyChess. The team will create a web application with an interactive dashboard that provides an analysis of chess games. In addition, there will be a machine learning engine that will provide a recommended move based on the current status of the game.

Millions of chess games have been analyzed, and it has revealed that there are common patterns that occur. For example, our dataset has many openings for White and Black. However, the total number of common openings for White is much smaller compared to Black. It must be due to the nature of the game since White always moves first; therefore, Black usually plays reactively. There are many other examples as well. So, what the project aims to do is create a machine learning engine that will train by observing over a million games. Once the training is done then theoretically the engine will be able to provide the next best move a player can make on their turn.

It is important to emphasize that the goal is not to create a tool that will lead a player to victory. Rather, the goal is to provide a tool that will help others study chess that is played competitively. Ultimately, it will be a tool used to learn more about the game. That is why the application will consist of other components such as game statistics or interactive visual graphs. These extra resources will help players understand the logic behind every move.

End Product Users

The end users for this application will be chess players and data science enthusiasts. There will be two small test groups to sample our software during the developmental phase. One

is the members of the Fresno Chess Club and the other are the members of the Fresno Data Science Club. Once the team reaches the production environment the application will be available to these two organizations where they will be able to access the application through the web. End users can play a game of chess and observe game records gathered since the 1970s, where millions of games have been recorded up until now. Users will also have an input interface to get a recommended move for their chess game. They will also have the ability to save game results where their user login will store and organize the data. This application will only be available to these two organizations in the near future, but eventually the goal is to make it available to the general public.

Feasibility

Environmental Scan

This project is based on the foundational work completed by Abraham, Mehar, and Sarom in the data science course, https://github.com/abecsumb/DataScienceProject. In that project, the objective was to predict whether a player would win the game based on parameters like the openings and the ELO rating of each player. The model used for this was kNN classification, and this model predicted the correct outcome about 70% of the time.

The team previously used a simple machine learning approach similar to this project, https://www.kaggle.com/code/andriizelenko/chess-classification. Our model was almost 15% better at predicting a win than the model shown in the link. Also, there are many chess engines and AI projects centered around chess prediction and analysis. One example is this, https://tinyurl.com/2z2pp4as and this https://www.chess.com/forum/view/general/machine-learning-in-chess. The chess engines are very complicated, and so are some machine learning models already out there. This project will

be different because we don't seek to develop an excellent chess engine. What we will make is a chess prediction program. The logic and best analysis are outsourced to the best players and chess engines. Our program will take all the available chess data and suggest the move most likely to win the game. This approach is like the greedy method; we make the best move at each step based on available data and analysis. Any algorithms and logic are limited to checking if a suggested chess move is legal and other basic checks. The chess meta (the best-understood way to play the game) has changed over the years, so this chess program will consider that when making predictions. The chess program will also consider the ELO ratings of each player. Based on our research, no other program is available that uses a purely statistical approach to play the game. Other chess programs are very complicated, and other chess machine learning programs try to do too much and are also very complex. Our chess program will prioritize simplicity and limit the metrics used for predictions to the most crucial information. Finally, it bears repeating that this program will seek to make the best decision at each move based on current legal moves and their likelihood to win based on historical data and analysis of chess games. In other words, our program will not plan and analyze future positions. This simple approach is what sets our project apart.

Project Justification

Chess is a complex game that may be intimidating for those new to it. The elo ratings between tiers normally have large gaps as well, which usually leads to high level plays only being experienced in tournaments.

This application aims to make chess more accessible for newcomers. Since there will be over a million game records within the database from high elo players. There should be an abundance of high quality games to observe. The application will also have the machine

learning engine that does the hard thinking first, then processes that information to make it digestible and easy for newcomers.

This tool will also be for data science enthusiasts. They will not only learn how to observe chess games, but they can also examine how large data is processed. As well as see how the chess notion was utilized during the training process. Overall this will be a tool that will be appealing to multiple groups.

Mitigation of Ethical and Legal Issues

An ethical concern that this application may run into is cheating. This application will have the feature to guess the next best move. Having the ability to predict may lead others to use the tool for malicious purposes. For example, gaining unfair advantages in an online tournament. Tournaments may have high rewards on the line, and of course require every participant to follow the rules to ensure fairness. However, if there are any fraudulent players caught in action then the tournament organizer can take legal actions to get compensation. Unfortunately, cheating can't be prevented as the Bradley Chess is just a tool that people can use freely. Fortunately, tournaments understand that it's not the fault of the chess engines, but the fault of users if they decide to cheat. That is why online tournaments have developed anti-cheating systems themselves to prevent such malicious use.

A feature of the application will be to insert game results into a database. This means that a login will most likely be used to keep track of users. Thus data privacy needs to be considered. Since a small team will be engineering this project. The handling of credentials will be outsourced. The plan is to use OAuth2 to manage login. OAuth2 is actually a popular standard that uses a platform service such as google, facebook and others to access data from another hosting website. That other host could be the Bradley Chess application. When using

OAuth2, the aforementioned websites will create an access token for users that will serve as a password for the host website. This means that the host will never know the true passwords or personal information of the user. These companies have a reputation for having robust security as well.

The team will also integrate the chess database and a chessboard interface from a third party source. This may indicate that there will be copyright concerns. Fortunately, these resources are open source licensed by MIT.

Project Life and Enhancements After Project Completion

Since the team plans to move this application into production for a small group. The requirement has been kept to a minimum. At the very least there will be two input fields on the frontend of the application. Each input field represents the two opponents of the game, and the fields will be able to send a chess move in the form of a string to the backend. The backend will take that string and feed it into the machine learning engine and return the next best move the player can make. The frontend will also have an ascii board that will show the users the current status of the chess match. For example, the positions of the pieces on the board. The main goal is to create a machine learning engine in the backend, and send best move information to the client.

After the project is completed the next step is to implement features the team intended to have. Due to having a small time frame to complete the project. A lot of features have been omitted. Once the team has delivered the final product to the client, then we can proceed to add authentication. Replace the ascii chess board with an interactive chess board interface, as well as add interactive graph visualization and pulling information from chess api.

Initially the application will only be available to the Fresno Chess Club and the Fresno Data Science Club, but later on the team would like to implement the other intended features and release this to the public. Now, the project life would be dependent on browser support. The frontend application is relying on the React framework. As long as the browser will continue to support the libraries or dependencies the team has used for the project. Then the application should remain stable. In terms of major updates that may affect browser support of current dependencies, then the application will be expected to last at least a year if not much longer.
>>

Part II

Design Requirements/Usability Testing

Platform

The webpage itself was made using React and its components. However, the chess move predictor was made using Flask. Flask is a backend service that allows for the development of web applications in Python. Some advantages to using Flask are that there is a built-in development server and swift debugger. The chessboard will update automatically as the user inputs a chess move into a text box and also when the RL agent responds with a move. A more interactive chess board was not feasible within the given time constraints. There is additional content on other web pages in our site. Another reason React was used on the frontend is because the chessboard we have is made using React.

Major Functions

Functions	User Requirements
Enter a move	Users will be able to input a move into the predictor, to get the next best outcome.
Visibility for the user	The next move will be printed out as a list with other moves available.
Graphical chessboard	Moves will be shown on a graphical representation of the chessboard. The chessboard will update as the users make chess moves.
Chessboard	Users will be able to have a chessboard be displayed towards the bottom and be allowed to play a game.
Agent Implementation	The Agent, or the computer, will be playing white and the user will be black. Allowing for the Agent to test and compete with the human's ELO rating. The agent was trained using the State Action Reward State Action (SARSA) algorithm.

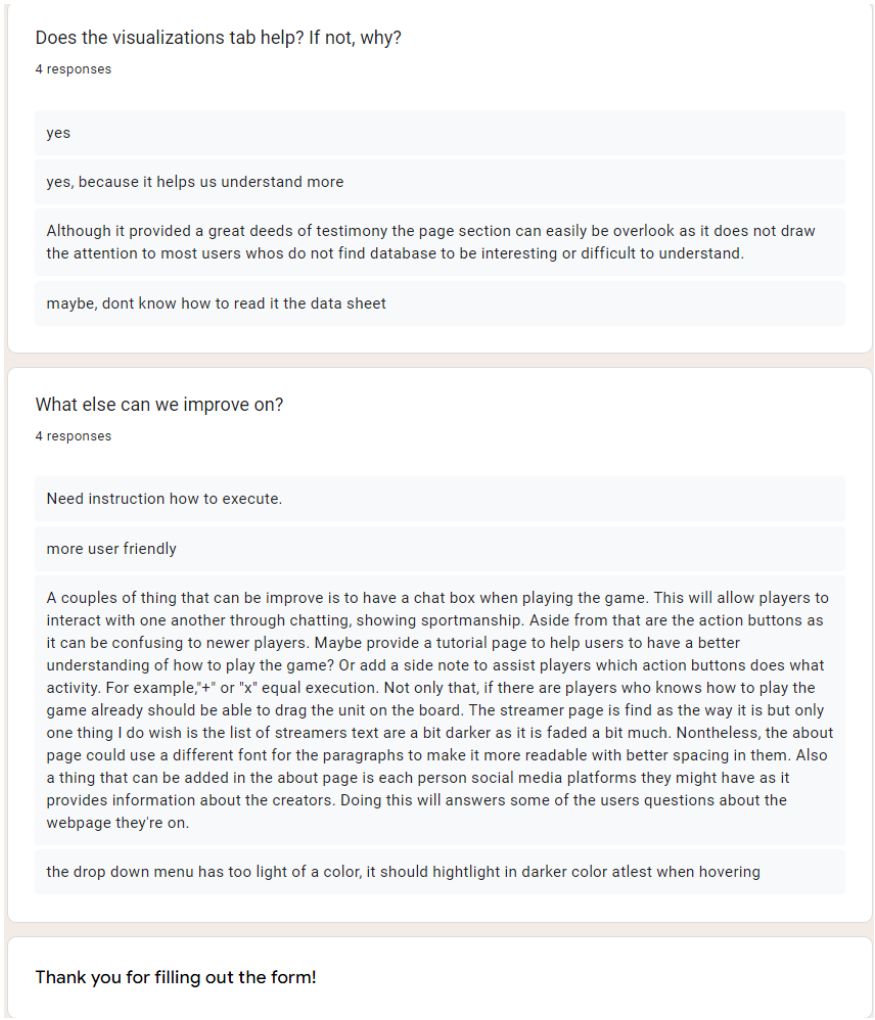
Figure 1. Major Functions

Usability Test Plan

- **Goals for Test Plan:** The targeted audience gets firsthand experience with the website and its predictor moves. The players will also be able to play a game of chess from the chessboard implemented in the website. The average chess player (400 ELO) will be challenged by the agent.
- **Target Audience:** Players at the Fresno Chess Club, Chess players, and various friends. Fresno Data Science Club.
- **Evaluation:** The evaluation will be completed through a Google form submission with a feedback form.
- **Procedure of actual project testing:** Players will play on the website through a PC. The games can help in improving the player's skills, while the visualizations can help in expanding the player's knowledge by displaying previous game moves through other chess games.

Usability Testing Responses

The responses we got back were greatly appreciated. For example, most of the people that reviewed our website enjoyed the visualization tab because it aided in a better understanding for the user. Some of them wanted the website to be a bit more user friendly, others wanted instructions and suggested adding a chat box. We will definitely take this critique into consideration and continue to build upon it. As seen on Appendix A and B.



Does the visualizations tab help? If not, why?

4 responses

yes

yes, because it helps us understand more

Although it provided a great deeds of testimony the page section can easily be overlook as it does not draw the attention to most users whos do not find database to be interesting or difficult to understand.

maybe, dont know how to read it the data sheet

What else can we improve on?

4 responses

Need instruction how to execute.

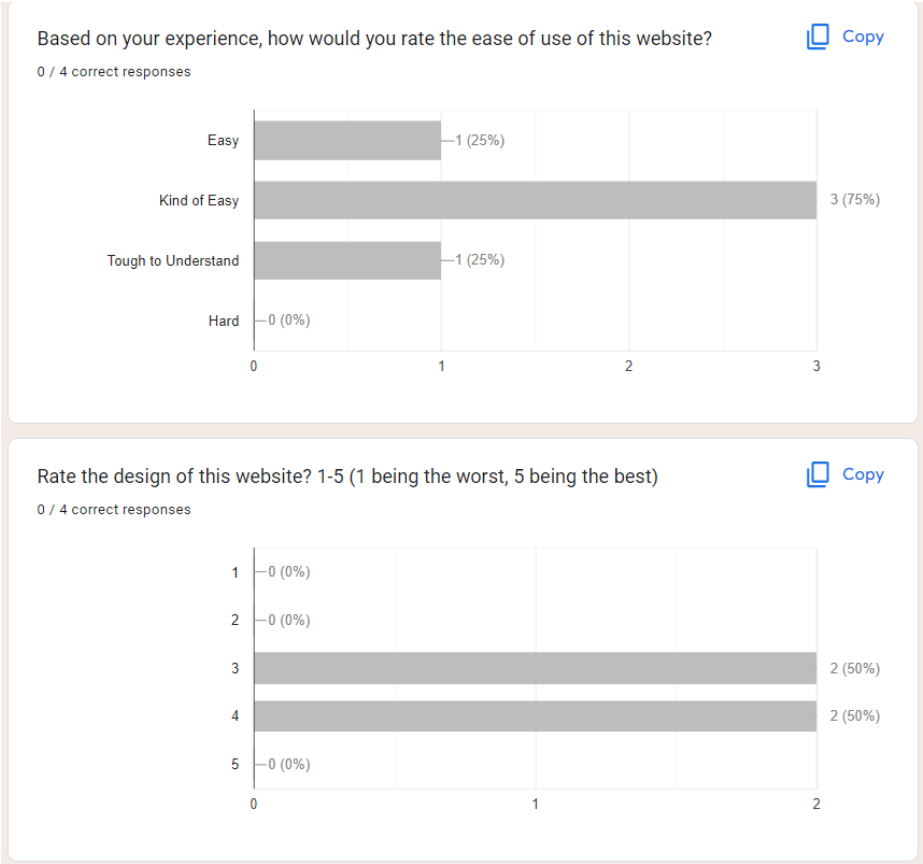
more user friendly

A couples of thing that can be improve is to have a chat box when playing the game. This will allow players to interact with one another through chatting, showing sportmanship. Aside from that are the action buttons as it can be confusing to newer players. Maybe provide a tutorial page to help users to have a better understanding of how to play the game? Or add a side note to assist players which action buttons does what activity. For example,"+" or "x" equal execution. Not only that, if there are players who knows how to play the game already should be able to drag the unit on the board. The streamer page is find as the way it is but only one thing I do wish is the list of streamers text are a bit darker as it is faded a bit much. Nontheless, the about page could use a different font for the paragraphs to make it more readable with better spacing in them. Also a thing that can be added in the about page is each person social media platforms they might have as it provides information about the creators. Doing this will answers some of the users questions about the webpage they're on.

the drop down menu has too light of a color, it should hightlight in darker color atlest when hovering

Thank you for filling out the form!

(Appendix A)



(Appendix B)

Part III

Final Discussion/Results

Timeline/Budget

Since none of us had experience in Flask and Dash, we began the project preparations earlier. This in fact did help us be more prepared for the shortened timeline that was given. Most of the milestones were met, however not without bugs as expected. Implementing the chessboard's logic was a bit of a tough aspect, nevertheless we pushed through and made it work.

Milestones

Week #1:	Dash app with basic visualizations, chessboard, and text field for chess move suggestions. Database complete with backend

	functionality. SARSA Implementation plan will be complete.
Week #2:	Dash chessboard logic and interactive pieces will be complete, additional visualizations with user input. SARSA algorithm python code due.
Week #3:	Dash will be complete and all chess logic is functional and chessboard is interactive. The SARSA agent has been trained and the algorithm can provide move suggestions that are shown to the user on the Dash.
Week #4:	Testing of components and other functionality. Additional features included games of professional chess players. Make the GitHub repo visible so the data science club can provide feedback.
Week #5:	Modifications or additions to Dash dashboard based on user input. Additional testing of code. Modifications to SARSA algorithm based on Fresno data science meetup members.

Figure 2. Weekly Milestones

The budget wasn't too big of an issue, seeing as how we thought most of this could have been accomplished without the need of any purchases. One of the few things that expended our budget a bit more was the ram. Abraham ended up needing to install more ram on his end, all the while spending a bit more on gas for the drives to the Fresno Chess Club.

Final Implementation

The web app has an interactive chessboard on the homepage where the user can play a game against the reinforced learning agent. Multiple agents were trained with different playstyles, and at the start of each game, a random agent will be selected. This feature will ensure that the user gets a unique game with each playthrough. See figure 3 for an overview of the site.

Figure 3. Bradleychess site overview

The user can use buttons or enter text to move the pieces. The agent will respond to user moves automatically and move the pieces on the chessboard. The next page on the site is the visualizations page. The user can view the data preparation and visualization. The chess database format is shown and the user will see the many moves for each game along the columns and the games as rows. In addition, the user can view live streams of popular chess players by going to the streams page. This functionality is made possible by using the API of chess.com

The algorithm that was used is SARSA reinforced learning. The algorithm itself is contained in a few lines of code. However, to make the algorithm work correctly, hundreds of

lines of code were needed. This chess engine implementation was wholly custom, and every aspect was carefully developed. Figure 4 shows the core of the SARSA algorithm,

```
    # check if game ended after opponent's move
    if curr_state['turn_index'] >= num_chess_moves:
        break
    else: # game continues
        # CRITICAL STEP, this is the SARSA algorithm
        next_Qval = curr_Qval + self.rl_agent.settings.learn_rate * (reward + ((self.rl_agent.settings.discount_factor * est_Qval) - curr_Qval))

        # on the next turn, this Q value will be added to the Q table. so if this is the end of the first round,
        # next round it will be W2 and then we assign the q value at W2 col
        curr_Qval = next_Qval
        # this is the next state, s'  the next action, a' is handled at the beginning of the while loop
        curr_state = self.environ.get_curr_state()

# reset environ to prepare for the next game
training_results.write(f'Game {game_num} is over.\n')
training_results.write(f'Game result is: {self.get_game_outcome()}\n')
training_results.write(f'The game ended because of: {self.get_game_termination_reason()}\n')
self.reset_environ()
```

Figure 4. SARSA algorithm critical section

The agent was trained on tens of thousands of games by using an adversarial agent. The objective of training is to fill out a ‘Q’ table. This table represents the playstyle that the agent has determined for itself. The Q table is really just a table of numbers (see figure 5). However, these numbers form the foundation of how the agent will behave. At each turn, the agent’s behavior is determined by the Q table.

	W1	W2	W3	W4	W5	W6	W7	
e4	486771.625804	-63844.280598	-10363.151819	-8546.441108	-4724.073906	-5388.332637	-5596.323159	-
d4	149402.492174	-63284.003685	-9770.877295	-8691.693398	-4840.308219	-5327.977568	-5451.920204	-
Nf3	-411090.642211	-64329.658818	-10068.879236	-8353.687250	-4788.890745	-5813.788834	-5362.916344	-
c4	-495962.469144	-63118.549618	-9853.275509	-8402.062373	-4722.058824	-5500.464553	-5110.693877	-
g3	-559137.785531	-63039.749173	-18560.461359	-9130.233109	-6489.978400	-6156.120131	-15089.457778	-2
f4	-680757.237002	-63132.638035	-10185.275851	-8216.705780	-4701.717132	-5594.186518	-5293.507393	-
Nc3	-668787.943434	-63121.500451	-9818.090937	-8402.714338	-4907.831907	-5470.428021	-5449.348033	-
b3	-654454.997913	-65232.860028	-10052.949138	-8228.747275	-4918.185184	-5350.693154	-5184.065070	-
b4	-677613.705334	-63063.080452	-9962.117766	-8211.341674	-5801.990246	-5534.230757	-5345.120231	-
d3	-623651.810454	-63387.456303	-10116.549042	-8278.184691	-4879.095251	-5407.745895	-5130.858078	-
e3	-683942.198374	-63035.708401	-9875.774805	-8390.137169	-4819.674860	-5457.994904	-5567.516822	-
h3	-591569.000976	-63068.601440	-9805.643873	-8246.821105	-4708.668921	-5552.928532	-5349.957216	-
c3	-660722.506415	-63102.279578	-10043.605611	-8300.388450	-4872.203709	-5514.137861	-5145.720073	-
a3	-560895.201522	-63684.510716	-10149.669302	-8293.978772	-5006.549084	-5482.865167	-5433.920342	-

Figure 5. Q table

Discussion

There were various different bugs that we were faced with. One problem for the front end was a conversion of a Jupyter Notebook into either an HTML or PDF, in order to have it display. Another complication was how to implement the Chessboard into the website and after that how to carry over the logic of the game itself. The solution was to meet with another person that aided us in a better understanding for the implementation, which most definitely helped us push through this obstacle. One of our targets was to use Plotly as well, however due to time restraints and not having too much experience in it, allowed us not to do so.

Mostly all of what we planned did end up making the final cut, however the usability testing took a bit longer. The Fresno Chess club had just gone to their break right as we had a product ready for their testing, which was a bit of an unfortunate event. This project made us all

realize the importance of time management, some more than others. We all took roles according to our strengths and weaknesses.

Implementing the machine learning algorithm correctly was very difficult. The most important part is defining the environment and what it means for the state to change. There were many versions of the implementation. In the first version, the environment was simply the list of legal chess moves and the current turn. In another it was everything, including the previous actions on the chessboard. Another important part of the implementation was defining the proper decision making policy. Many versions of this were tried as well. The best performance was gained by having the adversarial agents play out the moves in the chess database exactly as shown, and learning that way. In that implementation, the policy was the decisions that the great chess players made - the assumption being that they had very good reasons to make those chess moves. Another challenge was simply time and computational resources. It takes a very long time to train agents, and there is constant adjustment of parameters that needs to happen. Ideally, the agents would have been trained millions of times, but we were not able to achieve those numbers.

Appendix A

Team Members and Division of Work

Abraham Borg: Team lead, built the Machine Learning engine using the SARSA algorithm.

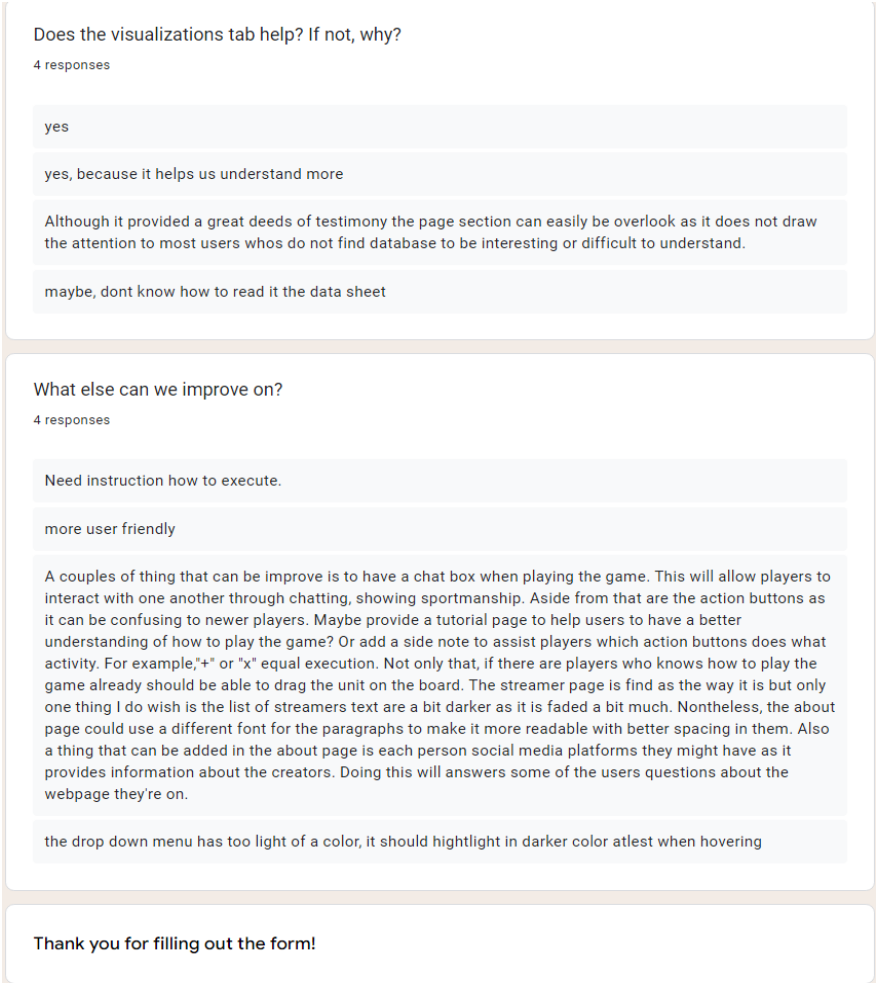
Mehar Rekhi: Frontend Developer, built React app and created the routes of the webpage.

Sarom Thin: Backend Developer, built Flask app and integrated the Machine Learning model.

User Responses

Google Form:

https://docs.google.com/forms/d/e/1FAIpQLSdKGTIzMDuQY8ncPQHrllNj1J1D9ZPS_ziuAZXIX7ScKQzhmQ/viewform



Does the visualizations tab help? If not, why?

4 responses

yes

yes, because it helps us understand more

Although it provided a great deeds of testimony the page section can easily be overlook as it does not draw the attention to most users whos do not find database to be interesting or difficult to understand.

maybe, dont know how to read it the data sheet

What else can we improve on?

4 responses

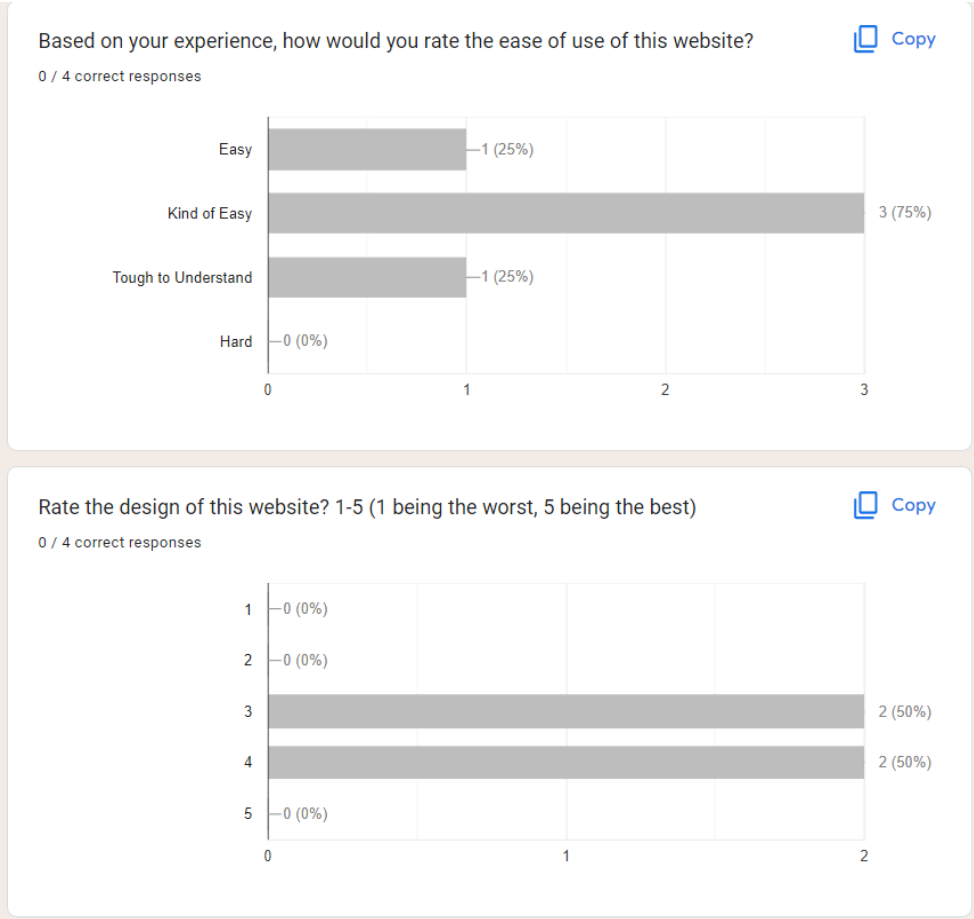
Need instruction how to execute.

more user friendly

A couples of thing that can be improve is to have a chat box when playing the game. This will allow players to interact with one another through chatting, showing sportmanship. Aside from that are the action buttons as it can be confusing to newer players. Maybe provide a tutorial page to help users to have a better understanding of how to play the game? Or add a side note to assist players which action buttons does what activity. For example,"+" or "x" equal execution. Not only that, if there are players who knows how to play the game already should be able to drag the unit on the board. The streamer page is find as the way it is but only one thing I do wish is the list of streamers text are a bit darker as it is faded a bit much. Nontheless, the about page could use a different font for the paragraphs to make it more readable with better spacing in them. Also a thing that can be added in the about page is each person social media platforms they might have as it provides information about the creators. Doing this will answers some of the users questions about the webpage they're on.

the drop down menu has too light of a color, it should hightlight in darker color atlest when hovering

Thank you for filling out the form!



Bradley Site Overview(Pg 15):

SARSA Algorithm Critical Section(Pg 16):

```
    # check if game ended after opponent's move
    if curr_state['turn_index'] >= num_chess_moves:
        break
    else: # game continues
        # CRITICAL STEP, this is the SARSA algorithm
        next_Qval = curr_Qval + self.rl_agent.settings.learn_rate * (reward + ((self.rl_agent.settings.discount_factor * est_Qval) - curr_Qval))

        # on the next turn, this Q value will be added to the Q table. so if this is the end of the first round,
        # next round it will be W2 and then we assign the q value at W2 col
        curr_Qval = next_Qval
        # this is the next state, s'  the next action, a' is handled at the beginning of the while loop
        curr_state = self.environ.get_curr_state()

# reset environ to prepare for the next game
training_results.write(f'Game {game_num} is over.\n')
training_results.write(f'Game result is: {self.get_game_outcome()}\n')
training_results.write(f'The game ended because of: {self.get_game_termination_reason()}\n')
self.reset_environ()
```

Q Table(Pg 17):

	W1	W2	W3	W4	W5	W6	W7	
e4	486771.625804	-63844.280598	-10363.151819	-8546.441108	-4724.073906	-5388.332637	-5596.323159	-
d4	149402.492174	-63284.003685	-9770.877295	-8691.693398	-4840.308219	-5327.977568	-5451.920204	-
Nf3	-411090.642211	-64329.658818	-10068.879236	-8353.687250	-4788.890745	-5813.788834	-5362.916344	-
c4	-495962.469144	-63118.549618	-9853.275509	-8402.062373	-4722.058824	-5500.464553	-5110.693877	-
g3	-559137.785531	-63039.749173	-18560.461359	-9130.233109	-6489.978400	-6156.120131	-15089.457778	-2
f4	-680757.237002	-63132.638035	-10185.275851	-8216.705780	-4701.717132	-5594.186518	-5293.507393	-
Nc3	-668787.943434	-63121.500451	-9818.090937	-8402.714338	-4907.831907	-5470.428021	-5449.348033	-
b3	-654454.997913	-65232.860028	-10052.949138	-8228.747275	-4918.185184	-5350.693154	-5184.065070	-
b4	-677613.705334	-63063.080452	-9962.117766	-8211.341674	-5801.990246	-5534.230757	-5345.120231	-
d3	-623651.810454	-63387.456303	-10116.549042	-8278.184691	-4879.095251	-5407.745895	-5130.858078	-
e3	-683942.198374	-63035.708401	-9875.774805	-8390.137169	-4819.674860	-5457.994904	-5567.516822	-
h3	-591569.000976	-63068.601440	-9805.643873	-8246.821105	-4708.668921	-5552.928532	-5349.957216	-
c3	-660722.506415	-63102.279578	-10043.605611	-8300.388450	-4872.203709	-5514.137861	-5145.720073	-
a3	-560895.201522	-63684.510716	-10149.669302	-8293.978772	-5006.549084	-5482.865167	-5433.920342	-