**Machine Learning and Strategy Games**

A Capstone Project

by

**Quinn Agabob and Ryan Burns**

Submitted to the Champlain College Faculty

in partial fulfillment of the requirements for the degree of

**BACHELOR OF SCIENCE**

**in**

**COMPUTER SCIENCE and INNOVATION**



Champlain College

2023

Copyright © by Quinn Agabob and Ryan Burns

2023

ABSTRACT

Our goal with this project was to design and implement an artificial intelligence to play a strategy game, as we wanted to work on an interesting use case for this emerging technology. Using a Python version of Connect 4 as a base for the project, we implemented a scoring system and used this to generate game data, which would later be used to train a machine learning algorithm. Screen interpretation and mouse movement were also implemented, so that our artificial intelligence would be able to interact with the game in a way similar to how a human might. In the end, our artificial intelligence was successfully able to play Connect 4 by interpreting the screen and controlling the mouse in order to make moves on its own. This project was ultimately a great learning experience for us, since there were many new concepts that we needed to figure out how to implement throughout this process.

Table of Contents

[**CHAPTER 1: INTRODUCTION**](#_30j0zll) 6

[Problem Statement 6](#_l8gkuhthla9p)

[Purpose Statement 6](#_3znysh7)

[Context 6](#_2et92p0)

[Significance of Project 8](#_tyjcwt)

[**CHAPTER 2: LITERATURE**](#_3dy6vkm) 10

[Literature Overview 10](#_1t3h5sf)

[Software 13](#_3rdcrjn)

[**CHAPTER 3: METHODS**](#_26in1rg) 15

[Design 15](#_lnxbz9)

[Frameworks 17](#_35nkun2)

[Algorithms 18](#_1ksv4uv)

[Analytical Methods 19](#_44sinio)

[Features 19](#_2jxsxqh)

[Test Plan 21](#_z337ya)

[Criteria and Constraints 21](#_3j2qqm3)

[**CHAPTER 4: RESULTS**](#_4i7ojhp) 23

[Final User Interface 23](#_2xcytpi)

[Analytical Results](#_2bn6wsx) [26](#_2hx34j6l59em)

[Testing Results 27](#_3as4poj)

[**CHAPTER 5: CONCLUSION**](#_49x2ik5) 28

[Challenges and Solutions 28](#_147n2zr)

[Future Work 30](#_23ckvvd)

[Project Importance 31](#_ihv636)

[**REFERENCES**](#_32hioqz) 32

# 

# CHAPTER 1: INTRODUCTION

## Problem Statement

Machine learning is an emerging technology that can be applied in order to complete tasks in unique and methodical ways. We want to design and implement a machine learning algorithm that can understand a strategy game, so that we can explore uncommon and interesting uses for it, as well as show off our skills as programmers.

## Purpose Statement

The purpose of this project is to create an artificial intelligence that is able to play a strategy game by interacting with opponents in the way a human might. It will serve as a creative and fun way to look into an otherwise underexplored application of artificial intelligence.

## Context

Artificial intelligence is essentially a system that is able to make decisions in a way similar to that of a human. This involves writing a program that tells the computer what to do given a certain situation. The concept of artificial intelligence is something that has been around for a while, but it is quickly becoming more mainstream, especially with all of its easily accessible applications in the modern day. Still, there are still certain applications that are not explored as often as others.

Video games are also something that have been around for a long time, and they have also become more mainstream over time. It would not be long before artificial intelligence would be used within video games to control certain behaviors, but using this technology to recreate the behavior of an actual player was much less common. Video games are all about giving a human player a set of instructions and putting them in situations where they are able to use that knowledge, so putting an artificial intelligence in this position seems like it would be a reasonable next step. Although video games and artificial intelligence are closely related in terms of making the game itself more interesting for the player, the concept of exploring how an artificial intelligence will adapt when put in the same situation as a human player is not incredibly common.

Machine learning is one of the major pieces that will be important in this project. Through the use of algorithms and data containing possible game states, we will be able to create a model that is able to improve without being given specific instructions about the game being played. Our focus will be to create an artificial intelligence that is not only capable of playing a strategy game, but it will also be able to adapt to what is happening in the game.

It is important to note that despite this specific application of artificial intelligence being underexplored, this is not the first time that a project like this one is being created. Projects such as AlphaZero, which is an artificial intelligence that can play board games like chess, are something to consider. Even smaller scale projects, such as the process of creating a chess AI outlined in an article by author Logan Spears is something that we will be able to learn from as we move forward. Some of the similar methods that we will be using include the creation and use of datasets containing possible game states, and using this data in order to train the model. However, the aspect of the project that will set it apart from most others is the ability for the program to interpret the game state using the output of the screen. Although the general idea of our project has been attempted in the past, we still plan on making certain features stand out.

By creating an artificial intelligence to play a strategy game, we will be researching and investigating an application of this concept that is more uncommon compared to other modern applications. It will ultimately be a way for us to take something that we enjoy and combine it with a concept that is not only useful towards the field of artificial intelligence, but a learning experience for both of us.

## Significance of Project

This project is important to us because machine learning is an emerging technology that is extremely versatile and efficient. Furthermore, machine learning software is able to discover patterns in data in ways that are far superior to what humans are capable of. The information gained by machine learning can be used to create solutions to real-world problems through predictive analytics. During development of this project, we want to be able to produce machine learning software that is capable of reading and interpreting an external game, producing a decision, and then outputting a response to that external game.

In the gaming industry, artificial intelligence is significant to both developers and players. For players, systems such as matchmaking are made easier if the game does not always have to find a lobby full of players. Games such as Call of Duty mobile will fill lobbies with artificial intelligence and players alike. Some players feel as if this takes away from the positives of player versus player combat, but not needing to fill entire lobbies with players can make matchmaking quick and easy for games with a small player base. From the perspective of a developer, artificial intelligence is great for producing a game, because it reduces the human labor necessary. “Artificial intelligence can automatically register and record the changes that it needs to make because AI learns on its own” (“The Impact of Artificial Intelligence In Online Gaming”). Without artificial intelligence, game developers would instead have to hire game testers, or wait for community feedback on any bugs or glitches. Using artificial intelligence, there is no need to devote manpower to this, as it can analyze the functionality of the game and point out changes that might need to be made.

# 

# CHAPTER 2: LITERATURE

## Literature Overview

The content of our project revolves mostly around machine learning, using Python to accomplish this, and putting it all together in the form of an artificial intelligence that can play a strategy game. To reflect this, many of the articles and other references that we will be making use of throughout the project have to do with one or more of these main ideas. “What is Machine Learning?” is an article by IBM with a target audience of people who know little about machine learning, and is excellent for teaching beginner concepts of machine learning. This article does not provide any information on the implementation of machine learning, but it does have excellent information on the theories of machine learning and how machine learning impacts the world around us.

An important source in terms of inspiration for our own project was “Train Your Own Chess AI”, which is an article by Logan Spears that documents his personal journey in making his own chess AI. This is a great article because it is a documented project that is very similar to ours. Spears talks about the theories and implementations of creating chess AI as well as examples of his code and chess games that the AI produced. It also gives information about some of the world’s greatest chess engines and the types of “super AIs”. Additionally, different types of artificial intelligence algorithms and machine learning methods are mentioned in “Algorithms for Artificial Intelligence” by Neil C. Rowe. One of the most important parts of artificial intelligence is the way in which human behavior is simulated, and this article provides descriptions of several methods that are able to do so. We will need to have a good understanding of what our options are before writing the code for the project, so this source will be useful for the early stages of the project.

“Analysis of Artificial Intelligence Applied in Video Games” is another source that directly relates to the topic of our project. In the article, Chengshuo Jiang writes about a few instances of artificial intelligence that are used to play games. The content of the article ranged from board games such as Go and the AlphaGo artificial intelligence, to simple video games on the Atari 2600. The author also includes a discussion section including some additional insight on machine learning and algorithms, and how they are being used and further improved.

Many of our sources deal with more of the specifics of machine learning, ranging from specific machine learning methods, where they can be used, and how these methods can be implemented in a Python development environment. For example, “Machine Learning: Hands-On for Developers and Technical Professionals” is a book on machine learning with a high-tech audience. Each chapter is a topic of machine learning and contains excellent information on the theory and execution. This book has vast amounts of extremely useful information on machine learning. The only negative of this book is that it does not use examples in Python code. Overall, this book is excellent for understanding theory. Similarly, “Machine Learning: Master Supervised and Unsupervised Learning Algorithms with Real Examples” is a book on the process of machine learning. Although it is mostly theory and has no code examples, it has great information on the mathematical aspects of machine learning and regression.

An article that will also be useful to us is “Machine Learning Made Easy: A Review of Scikit-learn Package in Python Programming Language” by Jiangang Hao and Tin Kam Ho. Since we will be using this particular Python library in our project, it is important to understand what it can be used for. The authors walk through the connections that scikit-learn has with machine learning and data transformation, which makes up a large part of what we will be trying to accomplish with this project. Also pertaining to machine learning methods, author Jorge Castañón writes about several different types of machine learning methods in “10 Machine Learning Methods that Every Data Scientist Should Know”. The author walks through each of the included methods, and mentions where they fall when it comes to machine learning implementation. We are aware that there are many possibilities with a machine learning and artificial intelligence-based project, so this article might help us choose a direction to start with.

Relating to machine learning, decision trees are a type of supervised learning algorithm that can be used to segregate outcomes. Decision trees are vital to our project because our artificial intelligence will need to use them in order to predict outcomes of the games we will be using. “Classification Based on Decision Tree Algorithm for Machine Learning” is a scholarly article on the concept of decision trees in the field of machine learning. The content of this article is perfect for our project because it is an in-depth analysis on the practice of machine learning. It has information on different types of algorithms used in decision trees and the pros and cons of each. This will greatly help us comprehend decision trees and how we can have our artificial intelligence utilize them in order to strengthen its strategic decision making process. An additional article covering the basics of decision trees and certain strategies involving them is “Decision Trees in Machine Learning” by Prashant Gupta. The author covers a few different techniques regarding the use and setup of decision trees, and explains how decision trees and algorithms can be used in machine learning. The article also mentions the Python library scikit-learn, which is already something that we plan on using during the project.

Lastly, “The Impact of Artificial Intelligence In Online Gaming” is an article highlighting the pros of artificial intelligence in the gaming industry. It is short and designated for a low-tech audience. Since it is useful to know the basic applications of artificial intelligence in gaming for this project, this article is sufficient. If we needed to expand on the topic of how artificial intelligence is used in the creation of games, we would need a more detailed source. However, since we are only interested in basic applications, this article provides a good place to start.

## Software

Since we are not the first to work on a project like this, there are certain aspects from other projects that we can use as inspiration for our own. A past project that will be a useful reference for us comes from an article by Logan Spears about creating an artificial intelligence that can play chess. This project had a very similar concept, as it had to do with training an AI to play a strategy game. The process that Spears goes through in the article is something that we can gather ideas from, as we will most likely be following a similar process during the creation of our own artificial intelligence.

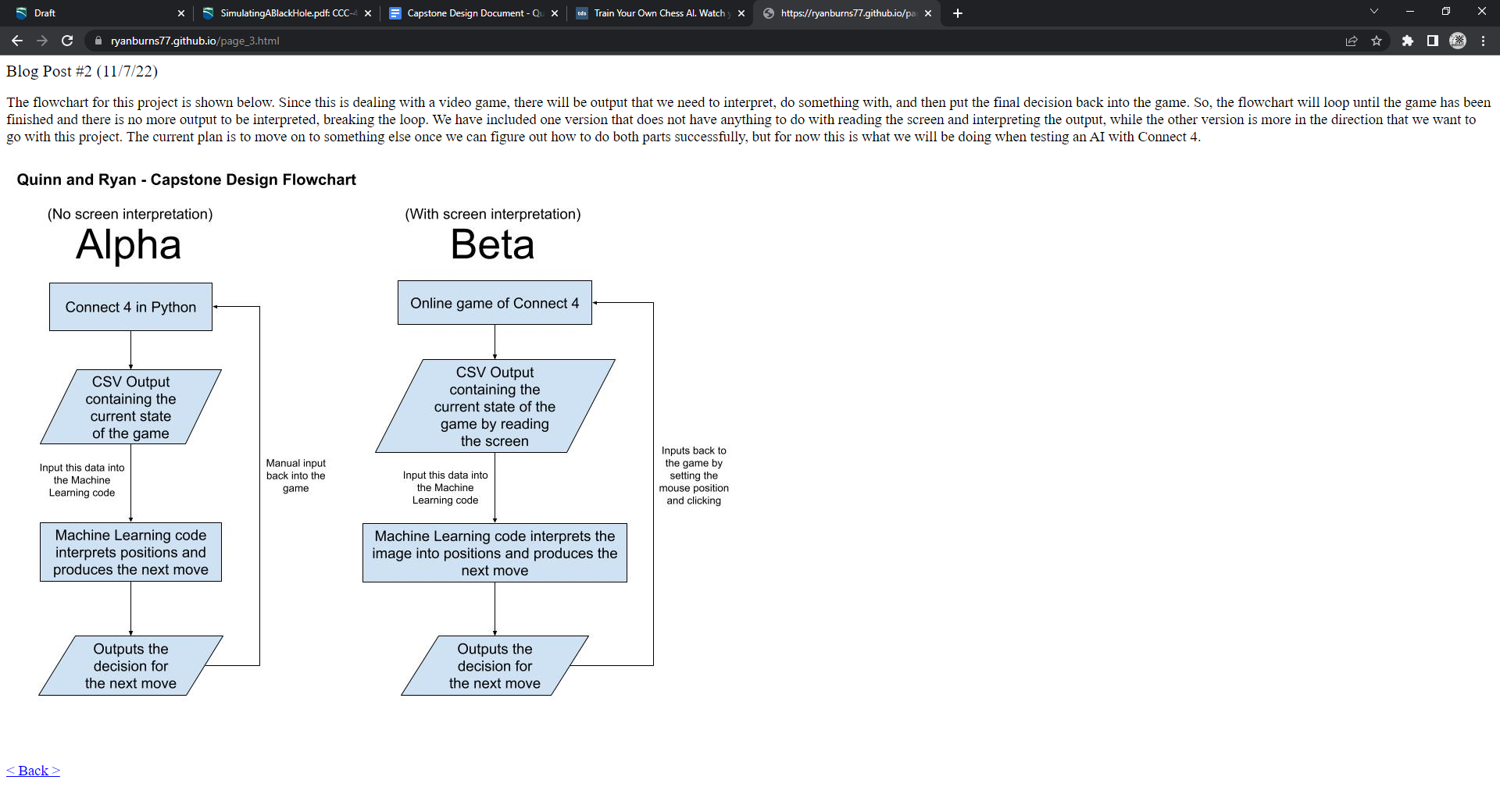
Even AIs for more difficult games like Go can serve as inspiration for our own project. This game is known for its complexity, since the number of possible board configurations is even higher than in something like chess. It is for this reason the previous artificial intelligence programs for Go were not able to play at a very high level, and is why newer artificial intelligence projects like AlphaGo are such a challenge to create. Although we will not be looking into board games as complex as Go, it is still important to learn about what is possible in the field of artificial intelligence.

The content of the project itself will be created using PyCharm, which will be used to write and edit the Python code for the artificial intelligence that we will be creating. Since our initial method of testing the code involves using a game programmed in Python, we will also be using PyCharm to get an understanding of how the game code works so that we can ultimately create an artificial intelligence model for the project. One of the primary reasons that we chose Python as the language to write our program in was because of the useful libraries that are available. For example, libraries like pandas and numpy will be especially useful for us given the nature of the project, since they deal with data analysis and performing calculations using that data. Additionally, in order to implement the machine learning aspect of the project we will be using scikit-learn along with a dataset containing possible game states. This will allow the model to improve similarly to how a human player might, which will ultimately fulfill one of the major parts of the project.

# 

# CHAPTER 3: METHODS

## Design



*Figure 1: Game and AI interaction at different stages of the project*

We are going to design an artificial intelligence that can play Connect 4. For the Alpha stage of the project, we will be running the game directly in Python, making use of source code by Keith Galli as the foundation. We will be storing game states from finished games into .csv files, and inputting these files into our code so that the machine learning algorithm can interpret the state of the game. The algorithm will then forecast the best move when playing additional games against an opponent.

Our machine learning algorithm will be looking for patterns that were used to win previous games and will try to avoid patterns that previously lost games. By recognizing these patterns, our artificial intelligence can determine which player is winning the game and by how much. Based on the configuration of the board, each player will be assigned a score based on this information. How accurately our artificial intelligence can detect who is winning and by how much is vital for creating strategic prowess. We will be using a decision tree to forecast the future outcomes of the game and correlate them with a score. Having an accurate interpretation of the state of the game is very important, because if our artificial intelligence model assigns scores inaccurately, it could end up in a position in which the end goal is believed to be favorable, but actually is not.

For the Beta stage of the project, our machine learning code will interpret information from a website containing the game, rather than directly from the Python code. To do this, we will create two new processes: a method to determine what is currently happening on the website and code that will make inputs to the website. To get information from the website we will need to read the screen to determine the state of the game. We can do this by determining the colors in particular spots on the screen, and interpreting the game state as a format that the machine learning code can understand. The machine learning code will determine the best move, just as it did in the previous stage. The decision will be inputted to this online version of Connect 4 by setting the mouse position and clicking to place a piece in the appropriate location.

## Frameworks

Our development environment for this project will be PyCharm, and we will be using Python to write the programs necessary to create a functioning artificial intelligence with aspects of machine learning. This was our program of choice, as we were both familiar with it, and choosing a single development environment for Python code would make the entire process much more streamlined. We may also decide to use services such as GitHub in order to easily share Python code once development is started.

The initial stage of the project involves creating an AI that can play Connect 4, with the intent of moving onto something more complex once the framework for playing and interpreting the states of a strategy game has been implemented. We will be using a Python version of Connect 4 as the base for this project so that the state of the game can be retrieved easily. Implementing alternate methods of interpreting the game state, such as determining what is being displayed on the screen, will be the next step. We will be making use of Python libraries to accomplish this, including pandas, numpy, scikit-learn, mouse and pillow. Both numpy and pandas are libraries that we are very familiar with. Pandas is great for structuring data and numpy is useful for general calculations. Scikit-learn is the core of our machine learning process, and will be a great tool in strengthening our AI. Libraries like mouse and pillow will be used during the Beta portion of our project. Pillow will be used to interpret the game state by reading colors on the screen, and mouse will allow our AI to automatically input mouse movements and clicks so that it can interact with an online version of the game.

## Algorithms

The most important part of this project is how well we will be able to train the machine learning code to determine who is in a winning position and by how much. This will be useful to us in determining if a sequence of moves is viable or not. We can do this by giving certain patterns to the machine learning code, with a value attached to each of them.

Decision trees are a popular machine learning algorithm used for classification and regression tasks. We are going to use a decision tree to forecast possible game states to and determine the value of the leaves in our decision trees. In a decision tree, the data is split into smaller and smaller subsets based on the values of the input features. “Decision trees are one of the powerful methods commonly used in various fields, such as machine learning, image processing, and identification of patterns. DT are a successive model that unites a series of the basic test efficiently and cohesively where a numeric feature is compared to a threshold value in each test.” (Jijo & Abdulazeez). Our decision tree will look at the possible sequences and determine which move is the best by examining the future game states. We will start off with only one branch containing the maximum amount of leaf nodes, and then add a branch while keeping the leaf nodes at the maximum amount. The more branches we have should hypothetically make our AI a better Connect 4 player. However, this is limited to how well our AI can understand the game state in terms of who is winning and by how much. For our project we will use a random forest regressor. A random regressor is a machine learning algorithm that utilizes many decision trees. This type of model is called ensemble learning, where one machine learning model takes in many other machine learning models.

## Analytical Methods

The analytical section of our project will be our machine learning code. This part involves a Connect 4 game state being inputted into our Python code, where our machine learning algorithm will then process the information and determine the best decision for the next move. The state of the game will be delivered to our machine learning algorithm in the form of a two-dimensional list. Using a 2D list will make the game state easy to read, format, import and export. Our machine learning code will judge all possible moves (maximum of 7), will determine the value of each move, and then choose what move to make. The value of each move will be determined by two measures. The first is by immediate presence, such as how the game state will be after the first move. The second measure involves a practice similar to how current chess AIs run. This is done by using a decision tree to determine all possible game states in the near future, making a value for each game state (who is winning and by how much), and then choosing the best move. The best move is the final branch in which the lowest value is the greatest compared to the lowest value of the other final branches. Therefore, even if the opponent plays perfectly, the game will still play out how the machine learning code forecasted. The strength of our code can be expressed as the AI’s ability to understand the state of the game and its ability to foretell the flow of the game.

## Features

As outlined in Figure 1 above, we will primarily be using the .csv file format to transfer the output data from the game and input it into the artificial intelligence for interpretation. For this early stage in development, the most important feature that we will need to implement is the data transfer process, so that the artificial intelligence and machine learning code are able to determine the next move based on this information.

The primary feature to be implemented during the second stage of the project is the ability for our program to determine which pieces are in which positions based on the colors of pixels being displayed on the screen. This will be accomplished through the use of Python libraries like pillow, and the goal for this stage will be to get this information into a format that is the same as what we had been using in the previous part. Additionally, rather than having to manually input what the AI determines the best move is, we will also use Python libraries like mouse to give our program some control over the location of the cursor on the screen. We plan on using an online version of the game to test the ability of our code to read the screen, so another goal of ours is to allow the program to control the mouse input to make moves on its own. This will ultimately complete the game loop in the online version without needing to input the AI’s move, as it will be able to perform that action without us needing to do anything.

Once we decide on another strategy-based game to continue with, the general structure of our code and the features that we will have already implemented will remain relatively the same. Ideally, the only differences will be the game output and the input to the artificial intelligence and machine learning code, as well as specific aspects of the screen reading and mouse input code, since we will need to find a different online source when changing the game that is being played.

## Test Plan

We plan to test our code against another artificial intelligence. This is the most consistent option, because while a human player is too unpredictable in skill, an AI has a more grounded skill range. There are different AIs found online with different skill ranges, so we will be able to record how well our AI does against the others to determine the strength of our code. In the beginning, we will be outputting the game states to a .csv file, but later in the project we will enable our code to read the screen in order to determine the state of the game, then determine mouse position and input to complete the move. This final method is how our AI will be able to interact with an online version of a strategy game.

## Criteria and Constraints

Since the end goal of this project involves creating an artificial intelligence, there are certain criteria and constraints that we must take into consideration as we move forward. A main goal of ours is to ensure that the outcome of the project is more beneficial than it is harmful. We will be able to do this by identifying possible issues now, so that we can avoid them once we begin constructing the project. For example, a possible environmental constraint would involve code efficiency and its relationship to energy usage and the environment. Despite the smaller scale of our project in comparison to much larger and more significant uses for artificial intelligence, it is still important to take this into consideration when writing code. Additionally, although there are constraints in the field of artificial intelligence regarding legal, economic and ethical impacts, we do not expect to run into any negative impacts in these areas throughout the course of the project, again due to the small scale that we will be working on.

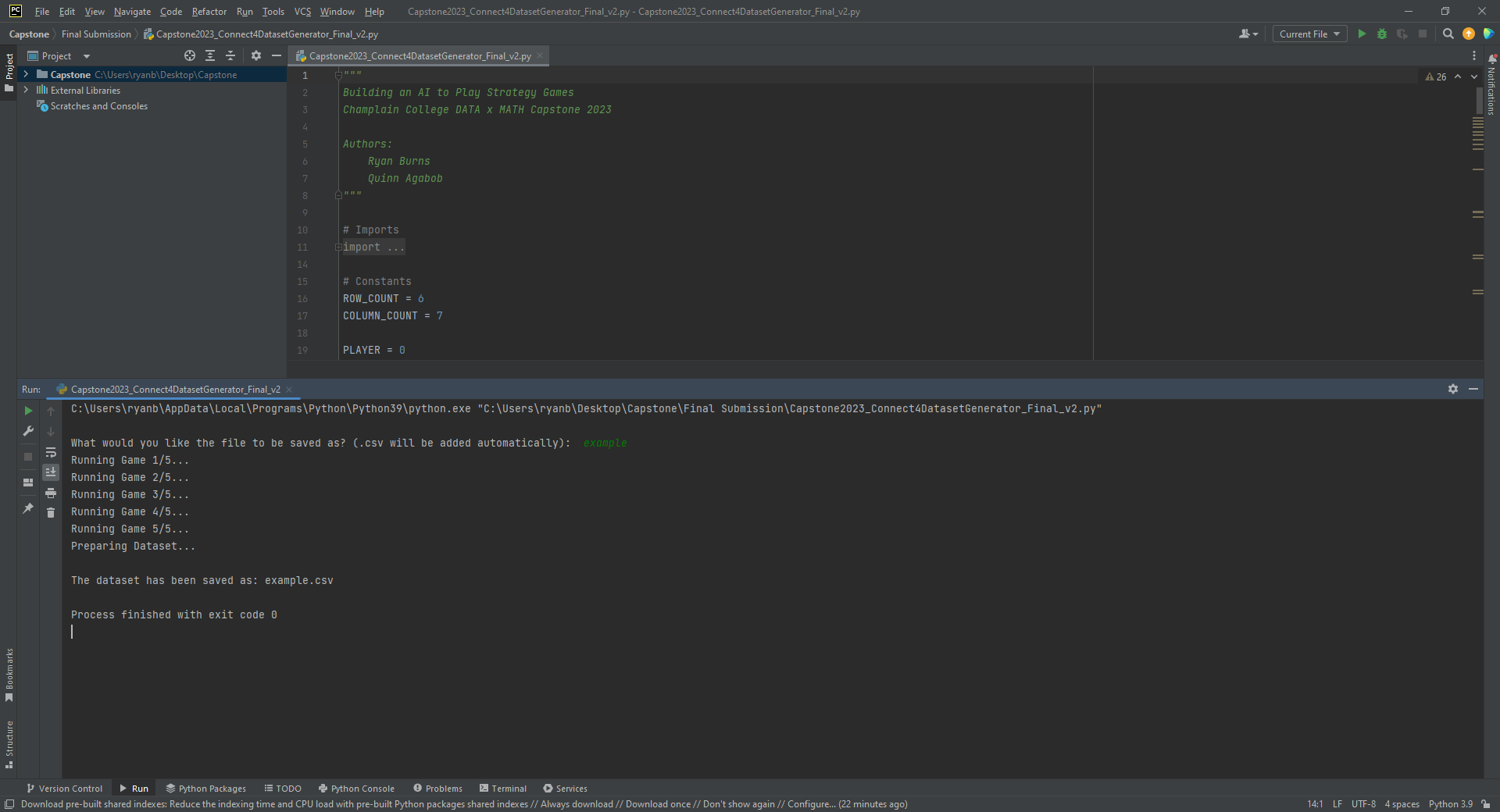
If all goes according to plan, by the end of this project we will have an AI model that has been trained with Connect 4 game data, as well as a different, more complicated strategy game. Due to the nature and scale of what we are trying to create, many of the common constraints are not applicable, especially since we are not planning on having the results of this project publicly released or published in any way, apart from the final deliverable. The results will simply be a representation of what we have learned and accomplished in the field of artificial intelligence, and how we were able to apply it to a subject that we both enjoy.

# 

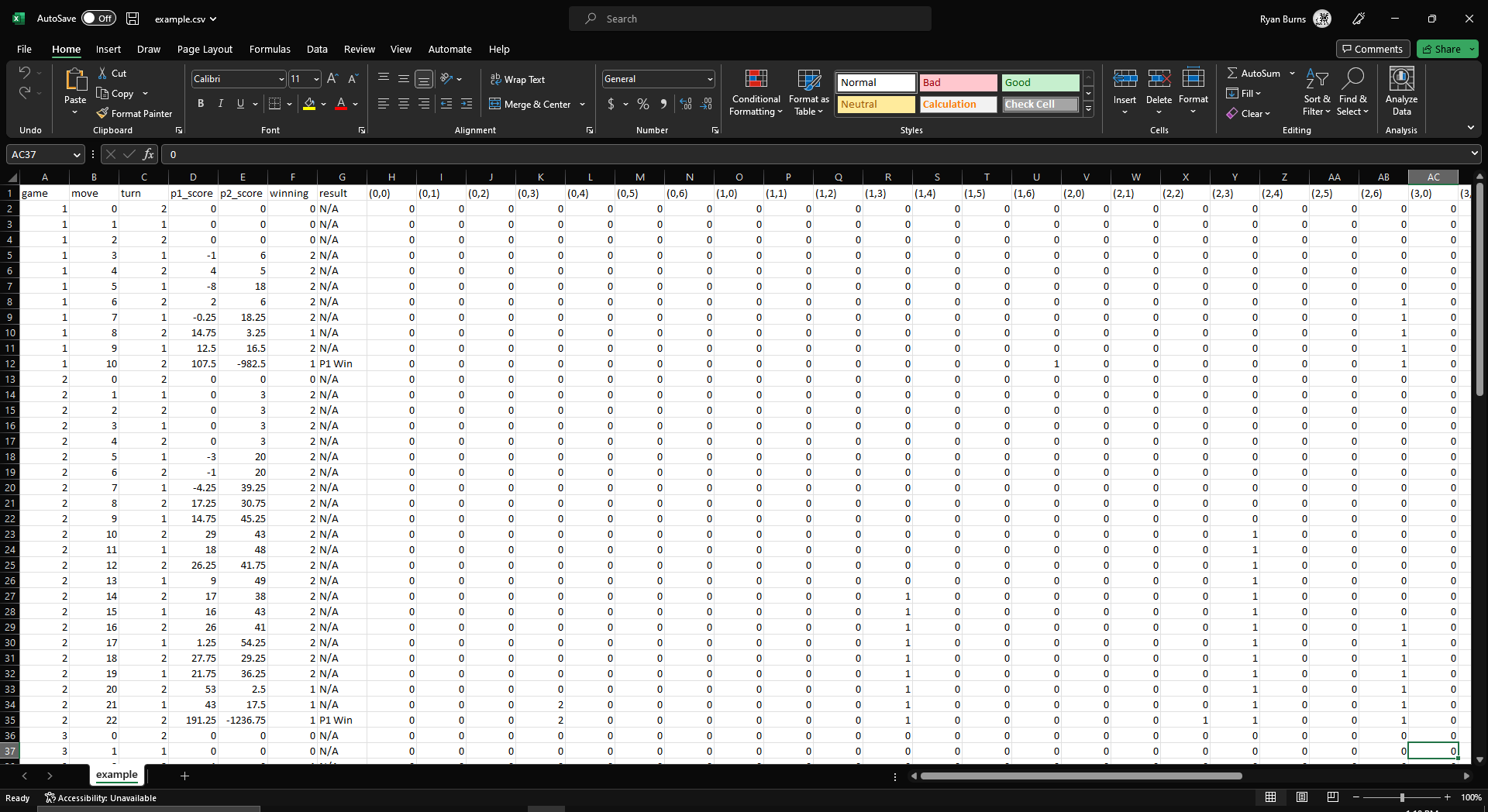
# CHAPTER 4: RESULTS

## Final User Interface

The final version of our project generally did not involve user interfaces. The two main pieces that we needed to be successful in our project implementation were the game state generator and the actual game loop that would allow our AI to control the mouse on its own. The most important part of the dataset generator was the file output shown in Figure 3, since this is what we would ultimately be using in order to train our machine learning algorithm. The actual user interface aspect of this part of the project was very minor, as it only asks the user for a filename and displays the status of the generator (Figure 2).



*Figure 2: Simple user interface when using the dataset generator*



*Figure 3: A dataset containing finished games of Connect 4*

The dataset generator is able to keep track of each game being played, and output every turn of each game into a .csv file, along with a few columns containing additional information about the current state of the game. The information columns come first in the output, followed by the current state of each position on the board, from left to right, top to bottom. Although not every board position is shown in Figure 3, the corresponding value will be zero when empty, one when occupied by Player 1, and two when occupied by Player 2. Additionally, this program was written in a way that would allow us to create a dataset with a variable number of games.

The other piece of the project was the artificial intelligence program, which is where machine learning and screen reading were implemented, and is what allows the AI to make its own moves through the use of the mouse. We did not need to create a user interface for this part of the project, since we would be making use of an existing Connect 4 website and allowing our AI to interact with it. While the program is running, the current interpretation of what the board looks like on the screen is output to the console. Figure 4 shows a side-by-side comparison of the board being displayed on the screen and what our AI is interpreting it as.

|  |  |
| --- | --- |

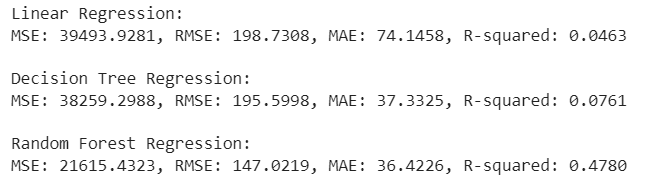
*Figure 4: An example of how the website display is interpreted using screen reading*

Since there needed to be a way to determine where the opponent has placed their next piece without having direct access to this, the program works by saving images of the board and comparing pixel colors in each of the positions where it is possible to make a move. Once the opponent makes their move, the red color of the opponent’s piece will be recognized as different from the background color, and the move will be reflected in the program output. Once the game has been finished on the website, our program will also recognize that the game has been finished.

## Analytical Results

## The goal of our project was to create an artificial intelligence to play strategy games and determine the best moves by implementing a machine learning algorithm. Before we decided which type of algorithm we wanted to use in our final implementation, we had to create multiple types of models and test the accuracy of each of them. We experimented with three models: Linear Regression, Decision Tree Regression, and Random Forest Regression. The tests that we conducted were Mean Squared Error (MSE), Root Mean Squared Error (RMSE), Mean Absolute Error (MAE), and R-squared.

Mean squared error is a representation of the average distance between the true value and predicted values. The model that had the smallest MSE was Random Forest Regression. Root mean squared error is the square root of the average of the distance between the true value and predicted values. Just like MSE, for RMSE, Random Forest Regression was the most accurate model. Mean absolute error finds the average absolute difference between predicted and true values. Using this test, Random Forest Regression and Decision Tree Regression were the most accurate models. R-squared is a measure of goodness of fit for a model to the data. Random Forest Regression was by far the most accurate model using this test. Based on the final results shown below in Figure 5, we decided to use Random Forest Regression as the model for our machine learning algorithm.



*Figure 5: Results after conducting several tests on different algorithms*

## Testing Results

A brief overview of what and how you tested your project codebase. You are required to integrate a formal testing framework with your codebase and have at least 30% code coverage with unit tests.

Testing the accuracy of our project ended up being somewhat challenging, since there is no way for us to know the best possible move given the current state of the board. To determine accuracy, our main metric was R-squared. This was a great metric for determining whether a model was accurate, and using the process outlined in the section above, we determined which machine learning model was the best fit for this project. We determined that Random Forest Regression was the best because of its significantly better R-squared result.

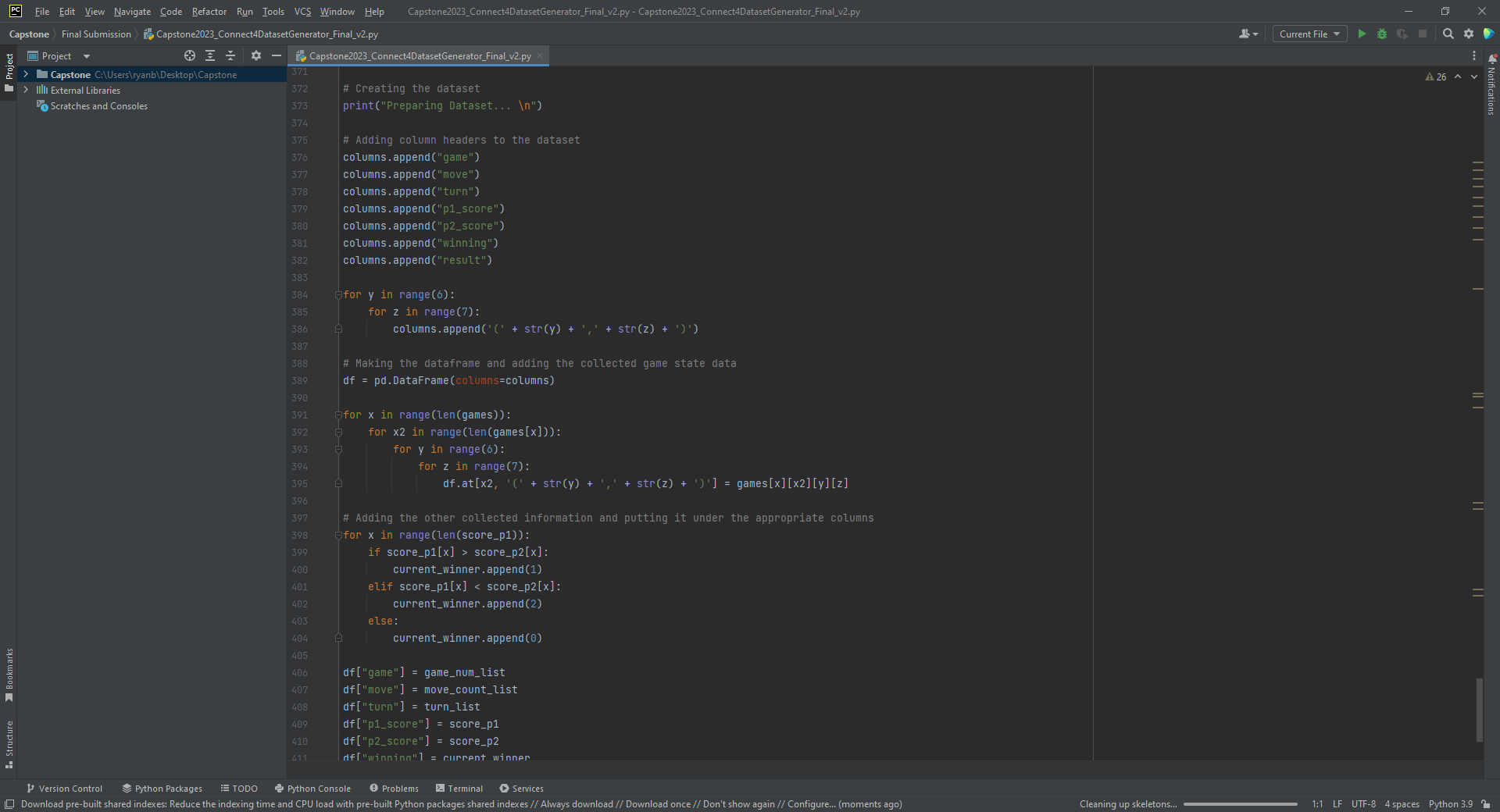
# 

# CHAPTER 5: CONCLUSION

## Challenges and Solutions

Throughout the development of our project, there were several challenges that we were tasked with solving. However, not all of these challenges came as a surprise once we began working. Some of these had been recognised during the design process, the most significant of which was the use of libraries in Python that we had little to no previous experience using. For example, screen reading and mouse movement were two of the most important parts of the project when it came to making it stand out from similar projects. The libraries that were used to do this, pillow and mouse, were unfamiliar to us. Although we had known from the beginning that this would be a challenge when we got to it, our solution was to make good use of the resources available to us. The implementation of these features was not perfect, but we were ultimately able to accomplish what we planned on doing while designing this project.

Data conversion was also a very important, yet challenging part of our project. While the game is being played, the game states are output as 2D lists. However, when the game information is saved, it must first be converted into a dataframe format. We decided to store each turn of each game in one row, where each position in the game state has a dedicated column. Additional information such as game number, turn, and player scores is also included. After each move is made, it is appended into a list, and after all games are played, they are converted into a dataframe. This is done using the code shown in Figure 6. First, we run a for loop for each game played (line 391), which is followed by a for loop that goes through the moves in that game (line 392). Then, a for loop for the rows (line 393) and another for the columns (line 394). Then we can assign the cell position by utilizing the variables created during the row and column loops, and assign it to the corresponding piece in the game.



*Figure 6: A code sample showing part of the dataset generator implementation*

Likely the most important challenge that we came across was deciding on a stopping place for our project, since it was becoming unlikely that we would be able to implement everything that was initially planned. Specifically, we had mentioned that building an AI that could play Connect 4 was only supposed to be one part of the project, and that we wanted to eventually train our model to be able to play a different, more complicated strategy game instead. There came a point during development in which it was decided that Connect 4 would be the only strategy game that we would be focusing on. This was a challenge that we had been hoping would not come up during the development of our project, but our solution was to make the most of the remaining time so that the project could be the best it possibly could be without going beyond Connect 4. Although the full completion of the initial project design would have been ideal, we were able to implement all of the other features that were discussed during the initial design phase.

## Future Work

Even though development on our Connect 4 AI has been completed, there are still many opportunities for future work on the project. One of the most important of these would be the improvement of the machine learning algorithm, since it still had some flaws in our final version. We could take the project in a few different directions from here, such as going back to the drawing board in terms of creating a scoring system that would best reflect the type of strategy that we would want the AI to be using. The scoring system is important when generating datasets to be used with the machine learning algorithm, so something as simple as this could significantly improve the results.

Now that we have the framework for an AI that can play strategy games, another logical next step would be to work towards having it play a more complicated game, since this was something that we had already planned to do before having to cut it from the project. This would involve making changes to both the dataset generator and the machine learning aspect of the project, but we would be able to use what we previously learned about machine learning and Python libraries in order to make these changes. In terms of reading the screen, it would be much more difficult to determine the state of the board by comparing individual pixel colors in a strategy game like chess, so we would need to look into different methods. Regardless of what might need to be done to advance our project further in the future, the open-ended nature of our project allows us to have many options for future work.

## Project Importance

The importance of our project comes from the unique combination of features that ultimately allowed us to be able to improve throughout the development process. Not only did we want our project to be something that we were both interested in working on, but using this process as a learning experience was also something that was very important to us. Much of the time spent working on this project was an important investment, whether it was spent researching machine learning methods, figuring out how to use new types of functions in Python, or simply waiting for game datasets to be generated for later use. Although the features that we were able to implement in the project are nothing new, especially considering the constant advancements in artificial intelligence, we believe that the particular combination of features on display is our contribution to the field of artificial intelligence. Even as artificial intelligence advances and the catalog of use cases expands, we will be able to look back on this project as something that we were interested in, something that we contributed, but most importantly: something that we learned from.

# 

# REFERENCES

Bell, J. (2020). *Machine Learning: Hands-on for Developers and Technical Professionals*.

*EBSCO* (2nd ed.). Wiley. Retrieved November 9, 2022, from

<https://discovery-ebsco-com.cobalt.champlain.edu/c/uomcmi/details/h573gjd6ov>

Castañón, J. (2019, May 1). *10 Machine Learning Methods that Every Data Scientist Should*

*Know*. Towards Data Science. Retrieved November 9, 2022, from

<https://towardsdatascience.com/10-machine-learning-methods-that-every-data-scientist-should-know-3cc96e0eeee9>

Galli, K. (2019). *Connect4-Python: Connect 4 Programmed in Python using Pygame*. GitHub.

Retrieved November 9, 2022, from <https://github.com/KeithGalli/Connect4-Python>

Gupta, P. (2017, May 17). *Decision Trees in Machine Learning*. Medium. Retrieved December

12, 2022, from

<https://towardsdatascience.com/decision-trees-in-machine-learning-641b9c4e8052>

Hao, J., & Ho, T. K. (2019). Machine Learning Made Easy: A Review of Scikit-learn Package in

Python Programming Language. *Journal of Educational and Behavioral Statistics*, *44*(3),

348–361. Retrieved December 12, 2022, from

<https://doi.org/10.3102/1076998619832248>

IBM Cloud Education. (2020, July 15). *What is Machine Learning?* IBM. Retrieved November

9, 2022, from <https://www.ibm.com/cloud/learn/machine-learning>

Jain, R. K., Doshi, R., Lakhwani, K., & Hiran, K. K. (2021). *Machine Learning: Master*

*Supervised and Unsupervised Learning Algorithms with Real Examples*. BPB

Publications. Retrieved November 9, 2022, from

<https://library.champlain.edu/record=b2277923>

Jiang, C. (2020). Analysis of Artificial Intelligence Applied in Video Games. Retrieved

November 9, 2022, from

<https://www.computer.org/csdl/proceedings-article/icaice/2020/914600a142/>

[1rCg6iw52Vi](https://www.computer.org/csdl/proceedings-article/icaice/2020/914600a142/)

Jijo, B. T., & Abdulazeez, A. (2021). Classification Based on Decision Tree Algorithm for

Machine Learning. *Journal of Applied Science and Technology Trends*, *2*(01), 20–28.

<https://doi.org/10.38094/jastt20165>

Rowe, N. C. (2022, July). Algorithms for Artificial Intelligence. *Computer*, *55*(7), 97–102.

Retrieved November 9, 2022, from <https://www.computer.org/csdl/magazine/co/2022/07/09810070/1EzDSNirAdy>

Spears, L. (2021, August 17). *Train Your Own Chess AI*. Towards Data Science. Retrieved

November 9, 2022, from

<https://towardsdatascience.com/train-your-own-chess-ai-66b9ca8d71e4>

*The Impact of Artificial Intelligence In Online Gaming*. Analytics Insight. (2021, December 22).

Retrieved December 12, 2022, from

<https://www.analyticsinsight.net/the-impact-of-artificial-intelligence-in-online-gaming/>