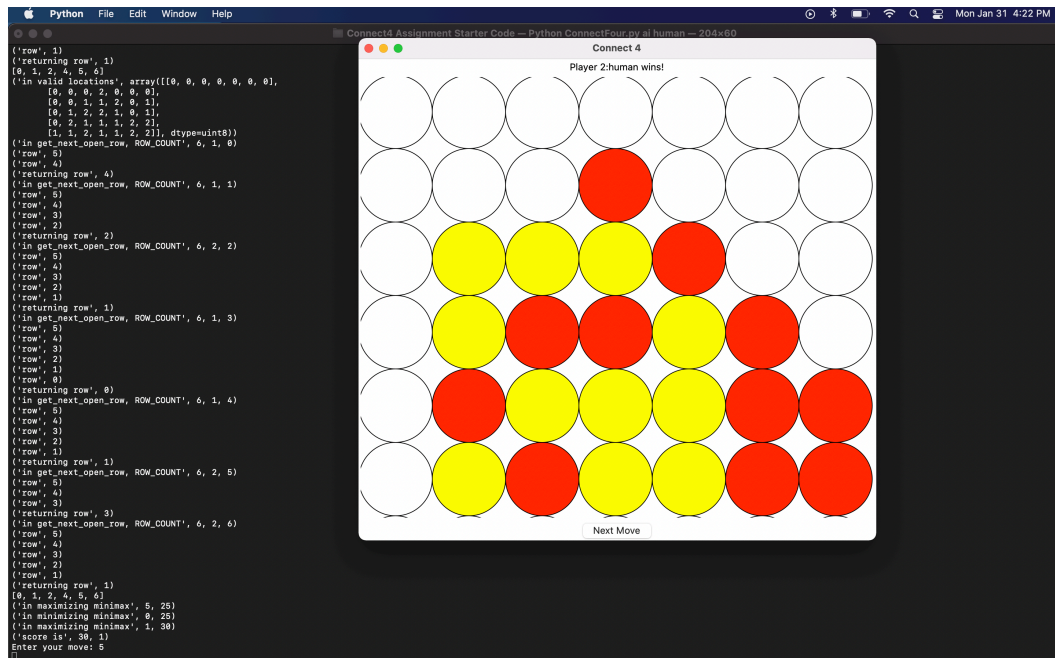


Connect 4 Assignment 2 Report

Attached alongside is the file Player.py containing the minimax function with alpha beta pruning and expectimax functions. I referred [this](#) blog for understanding minimax and alpha beta pruning and [this](#) for Expectimax algorithm

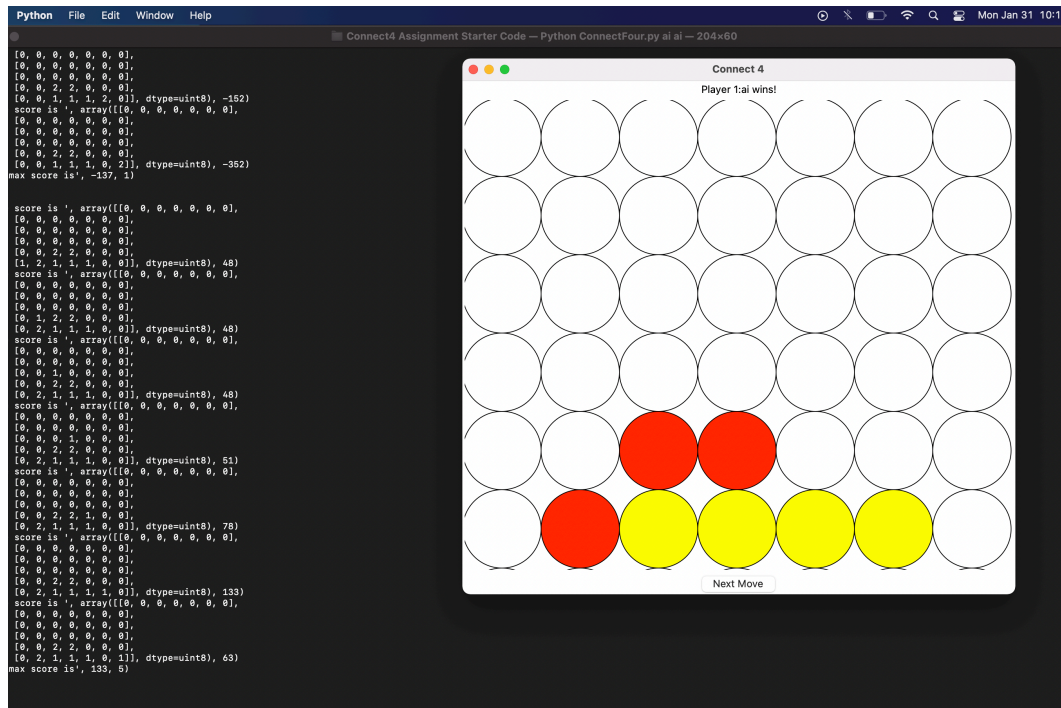
1. I used the number of pieces in the same row, column or diagonal to find the heuristic value of the board at a particular state. For 2 pieces, the score values to 15, for 3 it evaluates to 30 and for 4 to a full 100. Similarly, opponents pieces also have an impact on the score.
2. As the depth increase my algorithm performed a tad worse till depth of 4, after that started slipping a bit. In 5 seconds, the algorithm could work unto a depth of 3. It played at its optimal at the depth of 1.
3. Yes, as depth increased my algorithm, failed to detect nearest victory for me and I could beat it.



The screenshot shows a Python IDE window titled 'Connect 4 Assignment Starter Code - Python ConnectFour.py: 204x60'. The code on the left is a Python script for a Connect 4 game. It includes a board representation as a 6x7 grid of circles, with red and yellow pieces placed. The code defines a minimax function with alpha beta pruning and an expectimax function. The game is currently in a state where Player 2 (human) has just won, as indicated by the text 'Player 2: human wins!' on the board. The code also includes a 'Next Move' button at the bottom of the board interface.

I could beat the algorithm

- When the algorithm plays itself, the agent playing first has a distinct advantage and wins the game.



AI vs AI. Agent playing first wins