Tic Tac Toe Solver

Method used – Minmax Algo

Name- Samarth Shukla

Branch- CSE-AI (C)

University Roll number- 2024011003002 12

Date – 11-03-2025

Introduction:-

Tic tac toe is a two player game in which each player supposed to put their respective symbol either “X” or “0” in straight line in a 3x3 matrix for their win.

The game based on AI simulation will use a minmax algo concept for the most precise win or to draw the match. Minmax algo is decision making algorithm. It works by counting and find the all possible move for the win if possible of for the draw.

In this project I have made a unbeatable AI using minmax which a user max can take upto draw

Methodology:-

In this game the maximizing player is AI and the human user is set to be the minimising player. In the first step the user enters a valid entry and after that the AI will calculate all the future possibilities using the minmax algo and will take a move accordingly for the best win

Code Typed:-

```
import math



# Function to check if a player has won

def is_winner(board, player):

    win_patterns = [(0, 1, 2), (3, 4, 5), (6, 7, 8),  # Rows
```

```
                (0, 3, 6), (1, 4, 7), (2, 5, 8),  # Columns
                (0, 4, 8), (2, 4, 6)]  # Diagonals
    for pattern in win_patterns:
        if all(board[i] == player for i in pattern):
            return True
    return False


# Function to check if the board is full
def is_board_full(board):
    return all(space != ' ' for space in board)


# Minimax algorithm to evaluate moves
def minimax(board, depth, is_maximizing, player, opponent):
    if is_winner(board, player):
        return 1
    if is_winner(board, opponent):
        return -1
    if is_board_full(board):
        return 0
```

```
if is_maximizing:
    max_eval = -math.inf
    for i in range(9):
        if board[i] == ' ':
            board[i] = player
            eval = minimax(board, depth + 1, False, player, opponent)
            board[i] = ' '
            max_eval = max(max_eval, eval)
    return max_eval
else:
    min_eval = math.inf
    for i in range(9):
        if board[i] == ' ':
            board[i] = opponent
            eval = minimax(board, depth + 1, True, player, opponent)
            board[i] = ' '
            min_eval = min(min_eval, eval)
    return min_eval
```

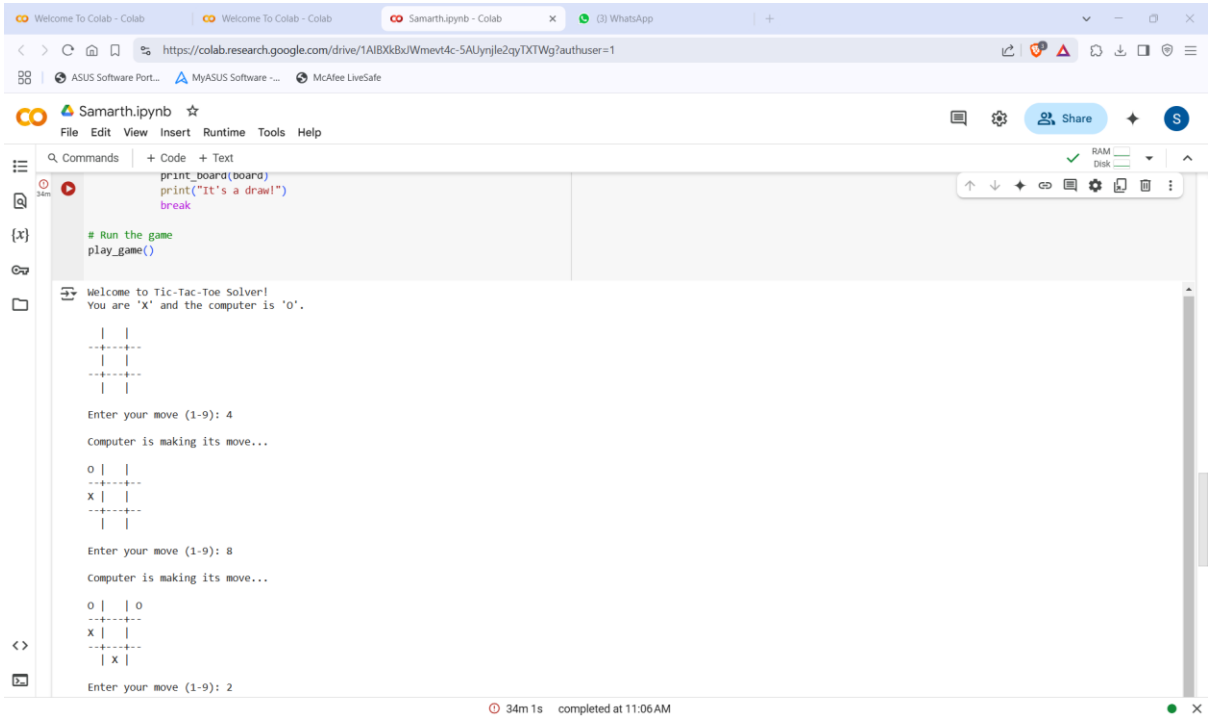
```
# Function to get the best move
def get_best_move(board, player, opponent):
    best_value = -math.inf
    best_move = -1
    for i in range(9):
        if board[i] == ' ':
            board[i] = player
            move_value = minimax(board, 0, False, player, opponent)
            board[i] = ' '
            if move_value > best_value:
                best_value = move_value
                best_move = i
    return best_move


# Function to print the board
def print_board(board):
    print(f"\n{board[0]} | {board[1]} | {board[2]}")
    print("--+---+--")
```

```
    print(f"{board[3]} | {board[4]} | {board[5]}")
    print("--+---+--")
    print(f"{board[6]} | {board[7]} | {board[8]}")


# Main function to play the game
def play_game():
    board = [' '] * 9
    player = 'X'  # Human player
    opponent = 'O'  # AI (computer)


    print("Welcome to Tic-Tac-Toe Solver!")
    print("You are 'X' and the computer is 'O'.")


    while True:
        # Print the current board state
        print_board(board)


        # Player move
        while True:
```

```
    try:
        move = int(input("\nEnter your move (1-9): ")) - 1
        if move < 0 or move > 8 or board[move] != ' ':
            print("Invalid move, try again.")
            continue
        break  # If the move is valid, break out of the loop
    except ValueError:
        print("Invalid input, please enter a number between 1 and 9.")

board[move] = player

if is_winner(board, player):
    print_board(board)
    print("\nYou win!")
    break
if is_board_full(board):
    print_board(board)
    print("It's a draw!")
    break
```

```
        # Computer (AI) move
        print("\nComputer is making its move...")
        best_move = get_best_move(board, opponent, player)
        board[best_move] = opponent

        if is_winner(board, opponent):
            print_board(board)
            print("\nComputer wins!")
            break
        if is_board_full(board):
            print_board(board)
            print("It's a draw!")
            break

# Run the game
play_game()
```

Screenshot


Samarth.ipynb
File Edit View Insert Runtime Tools Help
Commands + Code + Text
print_board(board)
print("It's a draw!")
break
# Run the game
play_game()
Welcome to Tic-Tac-Toe Solver!
You are 'X' and the computer is 'O'.
Enter your move (1-9): 4
Computer is making its move...
Enter your move (1-9): 8
Computer is making its move...
Enter your move (1-9): 2
34m 1s completed at 11:06 AM

Welcome To Colab - Colab | Welcome To Colab - Colab | Samarth.ipynb - Colab | (3) WhatsApp

https://colab.research.google.com/drive/1AIBXkBxJWmevt4c-5AUynjle2qyTXTWg?authuser=1

ASUS Software Port... | MyASUS Software -... | McAfee LiveSafe

Samarth.ipynb

File Edit View Insert Runtime Tools Help

Commands + Code + Text

```
  |   |
--+---+--
  |   |

Enter your move (1-9): 4

Computer is making its move...

O |   |
--+---+--
X |   |
--+---+--
  |   |

Enter your move (1-9): 8

Computer is making its move...

O |   | O
--+---+--
X |   |
--+---+--
  | X |

Enter your move (1-9): 2

Computer is making its move...

O | X | O
--+---+--
X | O |
--+---+--
  | X |

Enter your move (1-9): 5
Invalid move, try again.

Enter your move (1-9): 5
Invalid move, try again.
```

34m 1s completed at 11:06 AM

ENG IN 11:10 11-03-2025