A picture containing graphical user interface

Description automatically generated

**The Menace**

**Spring 2022 INFO6205**

**Project**

**Team Members**

Aishwarya Balyaya

Niul Panvalkar

Shubhang Shah

**Instructors**

Prof. Robin Hillyard

**Date: 04/27/2022**

Information Systems Department - Section 01

**Introduction:**

The design named MENACE-Tic-Tac-Toe: A pile of matchboxes that contains a number of beads and learned to play tic-tac-toe. Tic-Tac-Toe Board is a 3X3 square shaped grid . It works a bit like a Neural Network. Randomly optimized at the beginning, but after a few subsequent games, it adjusts itself to favorable moves which later succeeds in each increasing further situation.

**Aim:**

Train the model such that it gradually improves after losing the game based on the probability of its every wrong move. The model will learn from the wrong move taken by it for losing the game so that it can tackle the situation and ensure the next winning game.

**Approach:**

* Players are given a chance to play to implement their moves
* Aim is to minimize opponent’s benefit being human here and maximize menace benefit. Hence, used Min-Max Algorithm here.
* On the basis of the moves taken by the human, the menace model will store the current state.
* After the current state is stored, the random bead selected will denote the next move for menace.
* Menace will be punished if he losses the game for the wrong move taken by it by reducing its move by 1.
* Menace will be rewarded with three extra moves for the move taken by it leading to the winning game.
* A draw will be considered as being positive for the model being trained and improved and moves will neither be reduced nor increased.
* Draw conclusions from observations gained from its moves and hence the model is trained.

**Program:**

**Data Structures & Classes:** The data structures and classes are described as follows:

**Menace.py**: Train function here in Menace.py trains menace with random human moves received from random human moves function. Function also keeps track of menace actions based on which we train our menace

Text

Description automatically generated

Text

Description automatically generated

**Combinations.py**:

Text

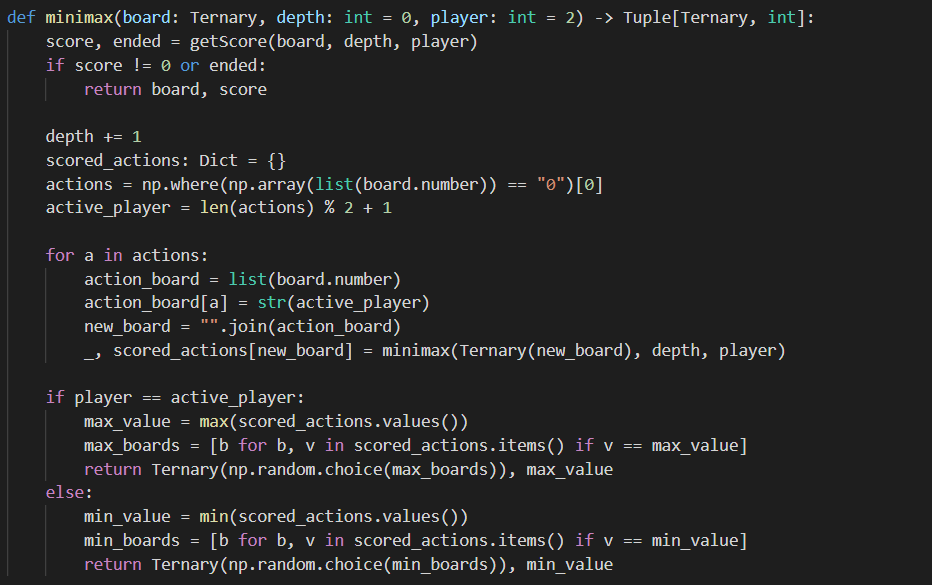
Description automatically generated

**Plot.py**:

Text

Description automatically generated

**MinMax.py:**



**Ternary.py:**

**Algorithms :**

**Step1:** Menace will be player 1 and Human will be player 2. Denoting all the fields by numbers 0-8. We are basically implementing Min-Max Algorithm which delivers an optimal action for the respective player, considering the opponent also to play finest.

**Step5:** The algorithm here uses depth-first search, exploring the position depth in the board then backtracks using recursive calls.

**Step3:** When the Menace has its turn, it tries to choose moves with the highest possible value.

**Assumption**(Opponent which is human in our case will be a perfect and astute player aiming to win and play optimally, whereas human being the opponent is playing the best move he can! That’s how Menace will be trained.)

**Step4:** When in human’s turn, we consider the action to be with the minimal optimum value.

**Step5:** Checking the value for every possible move, If menace wins for the particular move, we assign a value of 10 – the distance between the current board and the winning board

**Step6:** If menace losses the game for the particular move, we assign a value of the distance between the current board and the winning board – 10

**Step7:** When the total number of active becomes zero, stop the spread

**Step8:** Get the R factor from infectors and number of infected people and k factor when number of infected people is 0.7 \* people size i.e overspreading count