# **Artificial Intelligence and Machine Learning Fundamentals**

**Activity 2**: Teaching the Agent to Realize Situations When It Defends Against Losses

Follow these steps to complete the activity:

1. Create a function **player\_can\_win** such that it takes all moves from the board using the **all\_moves\_from\_board** function and iterates over it using a variable **next\_move**. On each iteration, it checks if the game can be won by the sign, then it return true else false.

def player\_can\_win(board, sign):

next\_moves = all\_moves\_from\_board(board, sign)

for next\_move in next\_moves:

if game\_won\_by(next\_move) == sign:

return True

return False

1. We will extend the AI move such that it prefers making safe moves. A move is safe if the opponent cannot win the game in the next step.

def ai\_move(board):

new\_boards = all\_moves\_from\_board(board, AI\_SIGN)

for new\_board in new\_boards:

if game\_won\_by(new\_board) == AI\_SIGN:

return new\_board

safe\_moves = []

for new\_board in new\_boards:

if not player\_can\_win(new\_board, OPPONENT\_SIGN):

safe\_moves.append(new\_board)

return choice(safe\_moves) if len(safe\_moves) > 0 else \

new\_boards[0]

1. You can test our new application. You will find the AI has made the correct move.
2. We will now place this logic in the state space generator and check how well the computer player is doing by generating all the possible games.

def all\_moves\_from\_board( board, sign ):

1. We will now place this logic in the state space generator and check how well the computer player is doing by generating all the possible games.

def all\_moves\_from\_board(board, sign):

move\_list = []

for i, v in enumerate(board):

if v == EMPTY\_SIGN:

new\_board = board[:i] + sign + board[i+1:]

move\_list.append(new\_board)

if game\_won\_by(new\_board) == AI\_SIGN:

return [new\_board]

if sign == AI\_SIGN:

safe\_moves = []

for move in move\_list:

if not player\_can\_win(move, OPPONENT\_SIGN):

safe\_moves.append(move)

return safe\_moves if len(safe\_moves) > 0 else \

move\_list[0:1]

else:

return move\_list

1. Count the possibilities that as possible.

count\_possibilities()

The output is as follows:

step 0. Moves: 1

step 1. Moves: 9

step 2. Moves: 72

step 3. Moves: 504

step 4. Moves: 3024

step 5. Moves: 5197

step 6. Moves: 18606

step 7. Moves: 19592

step 8. Moves: 30936

First player wins: 20843

Second player wins: 962

Draw 20243

Total 42048

We are doing better than before. We not only got rid of almost 2/3 of possible games again, but most of the time, the AI player either wins or settles for a draw. Despite our effort to make the AI better, it can still lose in 962 ways. We will eliminate all these losses in the next activity.