Policy Iteration Report

Question 3 – Write a Policy Iteration agent in PolicyIterationAgent.java by implementing the initRandomPolicy(), evaluatePolicy(), improvePolicy() & train() methods. The evaluatePolicy() method should evaluate the current policy (see your lecture notes), specified in the curPolicy field (which your initRandomPolicy() initialized). The current values for the current policy should be stored in the provided policyValues map. The improvePolicy() method performs the Policy improvement step, and updates curPolicy. **(6 points)**

Answer 3 –

Summary for Policy Iteration Agent Implementation

1) Functions Modified:
    - initRandomPolicy()
    - evaluatePolicy()
    - improvePolicy()
    - train()
2) Detailed Implementation Analysis:

(a) initRandomPolicy() Method

Purpose:

- Initialize a random policy for all valid game states
- Iterates through all states in the policy values
- For each state, selects a random move from the possible moves
- Ensures each state has a valid move assigned randomly

(b) evaluatePolicy() Method

Purpose:

- Evaluate the current policy's value function
- Iteratively updates state values until convergence
- Uses the Bellman equation to compute expected values
- Checks for convergence by comparing old and new state values
- Stops when the maximum change is less than the delta threshold

(c) improvePolicy() Method

Purpose:

- Improve the current policy by finding better moves
- Looks for moves that maximize expected value for each state
- Uses one-step look-ahead to find the best move
- Returns true if the policy was improved, false otherwise

(d) train() Method

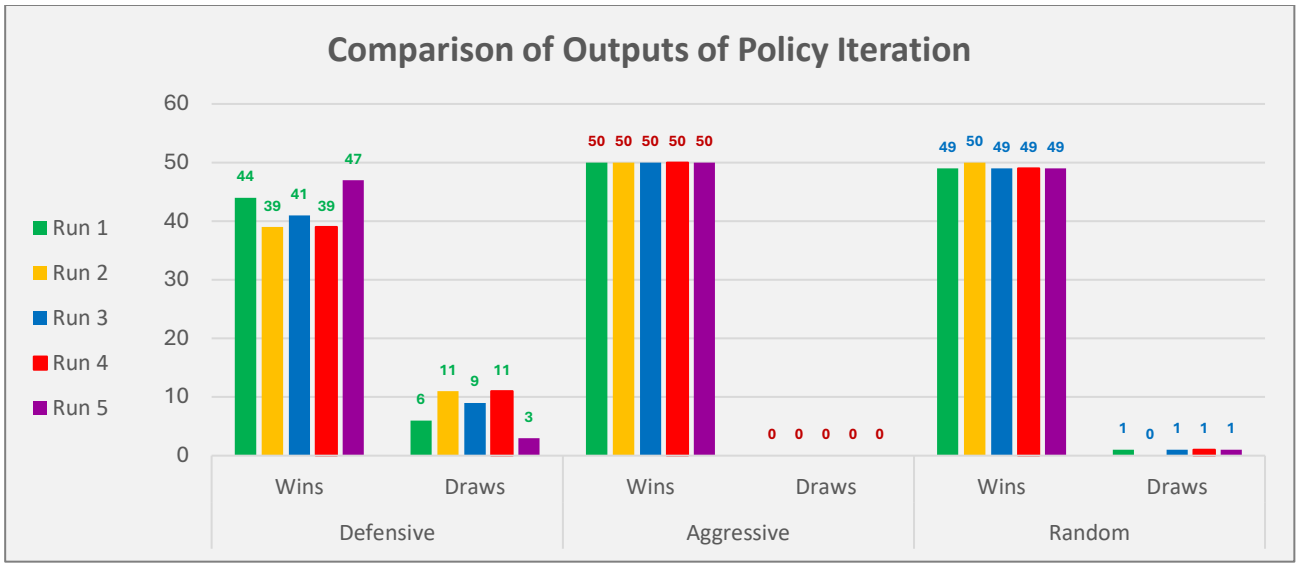
Purpose:

- Train the agent using Policy Iteration algorithm
- Alternates between policy evaluation and policy improvement
- Continues until the policy stabilizes (no further improvements)
- Sets the final policy for the agent

Question 4 – Test your Value Iteration Agent against each of the provided agents 50 times and report on the results – how many games they won, lost & drew against each of the other rule-based agents. The rule-based agents are random, aggressive, defensive. **(1 point)**

Answer 4 –

Iteration/ Agent	Defensive			Aggressive			Random		
	Wins	Losses	Draws	Wins	Losses	Draws	Wins	Losses	Draws
Run 1	44	0	6	50	0	0	49	0	1
Run 2	39	0	11	50	0	0	50	0	0
Run 3	41	0	9	50	0	0	49	0	1
Run 4	39	0	11	50	0	0	49	0	1
Run 5	47	0	3	50	0	0	49	0	1



The program was run for a total of 250 iterations split into 5 runs of 50 iterations each. As we can see from the tabular column, we obtain the best result during iteration 5 of the Value-Iteration Agent.