MINI PROJECT -1

Name: Chakradhar reddi vitta

ID : 934595987

vittac@oregonstate.edu

PART – A

1)

The agent moves in a 4x4 grid, aiming to reach the goal at (3,3).

Some states contain fire (dangerous) and water (less dangerous).

The agent can move Up, Down, Left, or Right, but there is a chance of slipping in another direction.

Rewards:

+100 for reaching the goal.

-10 for stepping on fire.

-5 for stepping on water.

-1 for each regular move to encourage shorter paths.

I have setup a environment where agent follows all this above conditions and moves in a right direction to reach the goal.

2)

Policy for gamma = 0.3:

State (0, 0): L

State (0, 1): L

State (0, 2): D

State (0, 3): D

State (1, 0): U

State (1, 1): R

State (1, 2): R

State (1, 3): D

State (2, 0): D

State (2, 1): D

State (2, 2): R

State (2, 3): D

State (3, 0): R

State (3, 1): R

State (3, 2): R

State (3, 3): None

Objective values for gamma = 0.3:

State (0, 0, 0, 0): -1.072415

State (0, 1, 0, 1): -10.619763

State (0, 2, 0, 1): -10.348550

State (0, 3, 0, 0): -0.152639

State (1, 0, 0, 0): -1.341412

State (1, 1, 0, 0): -1.459676

State (1, 2, 0, 0): -0.105740

State (1, 3, 0, 0): 4.843320

State (2, 0, 0, 0): -1.124496

State (2, 1, 1, 0): -3.856861

State (2, 2, 1, 0): 1.410669

State (2, 3, 0, 0): 23.754969

State (3, 0, 0, 0): 0.103937

State (3, 1, 0, 0): 4.727306

State (3, 2, 0, 0): 23.754969

State (3, 3, 0, 0): 100.000000

Policy for gamma = 0.95:

State (0, 0): D

State (0, 1): D

State (0, 2): R

State (0, 3): D

State (1, 0): D

State (1, 1): R

State (1, 2): R

State (1, 3): D

State (2, 0): D

State (2, 1): D

State (2, 2): R

State (2, 3): D

State (3, 0): R

State (3, 1): R

State (3, 2): R

State (3, 3): None

Objective values for gamma = 0.95:

State (0, 0, 0, 0): -1.072415

State (0, 1, 0, 1): -10.619763

State (0, 2, 0, 1): -10.348550

State (0, 3, 0, 0): -0.152639

State (1, 0, 0, 0): -1.341412

State (1, 1, 0, 0): -1.459676

State (1, 2, 0, 0): -0.105740

State (1, 3, 0, 0): 4.843320

State (2, 0, 0, 0): -1.124496

State (2, 1, 1, 0): -3.856861

State (2, 2, 1, 0): 1.410669

State (2, 3, 0, 0): 23.754969

State (3, 0, 0, 0): 0.103937

State (3, 1, 0, 0): 4.727306

State (3, 2, 0, 0): 23.754969

State (3, 3, 0, 0): 100.000000

When the discount factor was low, the agent avoided wildfire and water because it focused more on short-term rewards. It tried to stay away from dangerous states to avoid losing points. But when the discount factor was high, the agent started thinking more about the future rewards. It sometimes moved through wildfire or water if it helped reach the goal faster. This means that with a higher discount factor, the agent cared more about the long-term goal, even if it had to take some small losses along the way.

3)

Policy for gamma = 0.95:

State (0, 0): D

State (0, 1): D

State (0, 2): R

State (0, 3): D

State (1, 0): D

State (1, 1): R

State (1, 2): R

State (1, 3): D

State (2, 0): D

State (2, 1): D

State (2, 2): R

State (2, 3): D

State (3, 0): R

State (3, 1): R

State (3, 2): R

State (3, 3): None

Objective Values:

State (0, 0, 0, 0): -1.072415

State (0, 1, 0, 1): -10.619763

State (0, 2, 0, 1): -10.348550

State (0, 3, 0, 0): -0.152639

State (1, 0, 0, 0): -1.341412

State (1, 1, 0, 0): -1.459676

State (1, 2, 0, 0): -0.105740

State (1, 3, 0, 0): 4.843320

State (2, 0, 0, 0): -1.124496

State (2, 1, 1, 0): -3.856861

State (2, 2, 1, 0): 1.410669

State (2, 3, 0, 0): 23.754969

State (3, 0, 0, 0): 0.103937

State (3, 1, 0, 0): 4.727306

State (3, 2, 0, 0): 23.754969

State (3, 3, 0, 0): 100.000000

Policy iteration was done with a discount factor of 0.95. It found the best policy and objective value, which were the same as value iteration. Policy iteration worked faster because it updated the full policy at each step, while value iteration took more steps to reach the same result.

PART – B

1)

Position: (0, 0), Move: D, Score: -1

Position: (1, 0), Move: D, Score: -1

Position: (1, 0), Move: D, Score: -1

Position: (2, 0), Move: D, Score: -1

Position: (2, 0), Move: D, Score: -1

Position: (3, 0), Move: R, Score: -1

Position: (3, 1), Move: R, Score: -1

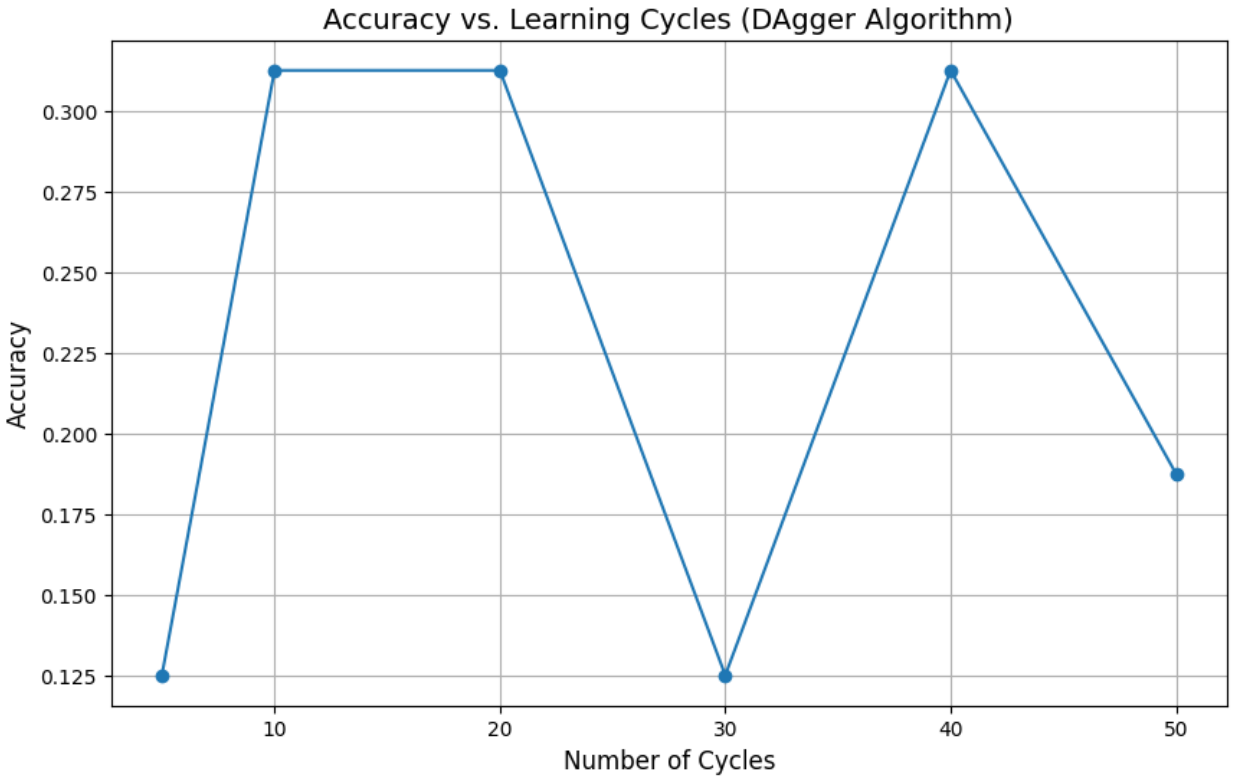
Position: (3, 2), Move: R, Score: -1

Position: (2, 2), Move: R, Score: -5

Position: (3, 2), Move: R, Score: -1

Position: (3, 3), Move: None, Score: 100

2) I have implemented the Dagger algorithm as required in Part B, Sections. I set the blend factor (βi) to 0, meaning the learned policy does not use the expert policy directly but instead learns from collected experience. I choose a Decision Tree classifier to train on the collected dataset, where each state is labeled with the corresponding action from the trajectory. The classifier is trained as more data is collected, and it predicts actions for all states in the MDP. The learned policy is updated based on these predictions. I also ensured that all MDP states were included in the test set, following the project instructions. Finally, I conducted experiments with different values of N = {5, 10, 20, 30, 40, 50} and plotted accuracy to analyze the learning performance over multiple iterations.



The graph shows that accuracy does not always increase as N gets bigger. At N = 5, accuracy starts low but improves at N = 10 and 20. Then, at N = 30, accuracy drops a lot, meaning the model did not learn well at that point. At N = 40, accuracy improves again, but at N = 50, it goes down. This means that more training does not always make the model better. Sometimes, new training data may confuse the model instead of helping it learn. The results show that training helps, but accuracy can go up and down depending on how well the model learns from the data.