Assignment-3

Defining and Solving Reinforcement Learning Task

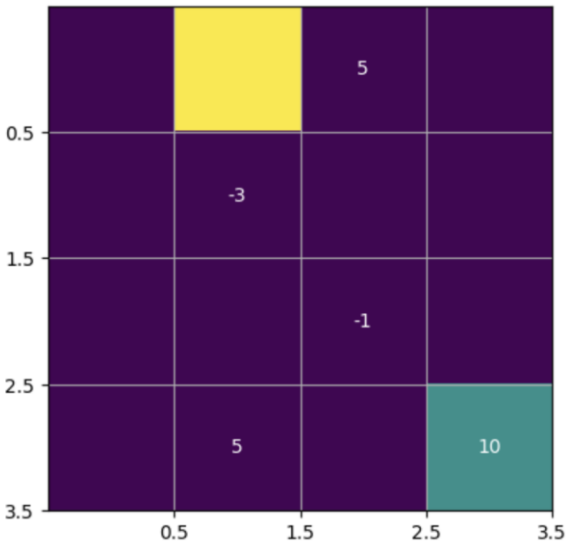
Part-1

1.) Environment Configuration:

- We created a (4, 4) grid environment with an **Observation Space of 16**.
- **Observation Space Positions:**
  {(0,0), (0,1), (0,2), (0,3), (1,0), (1,1), (1,2), (1,3), (2,0), (2,1), (2,2), (2,3), (3,0), (3,1), (3,2), (3,3)}
- **Agent Starting Position:** (0,0)
- **Goal Position:** (3,3)
- **Action Set:** 4 Actions ({0: Down, 1: Up, 2: Right, 3: Left})
- **Rewards:**
  - We have Placed Positive rewards in 3 Positions: {(0,2): +5, (3,1): +5, (3,3): +10}
  - We have placed Negative rewards in 2 Positions: {(1,1): -3, (2,2): -1}

2.) Visualization of our Environment:

- We represented our Agent and Goal Positions in different colors and, we have also added the rewards in the grids.



3.) Safety of our Environment:

- We have restricted the grid environment to an observation space of 16. Restricting the observation space of the environment ensures that the Agent's movements are restricted to a well-defined area reducing the risk of undesirable consequences outside scope of our grid environment.
- We have used a function 'np.clip' to prevent the agent from moving outside our grid environment. The 'np.clip' function we used ensures that our agent stays within our defined grid environment.
- We are resetting the state of environment at the beginning of each episode. This step ensures that the changes made in the previous interactions do not interfere with the next episode.
- We are terminating the workflow based on the maximum number of timestamps or when the agent reaches the goal position. By doing this we can ensure that our agent does not run into an infinite loop.
- We have placed rewards in some positions of our grid environment to control the agent's movements. This will ensure that our agent does not perform any unsafe actions in the search of high rewards.

PART 2 &3

1.) The Tabular methods used to solve the problems are:

- SARSA
- Q-Learning

Both SARSA and Q-learning can be described as tabular methods because they maintain a table of Q-values to estimate the optimal value of each state-action pair.

SARSA: SARSA is an on-policy algorithm which means that while learning the optimal policy it uses the current estimate of the optimal policy to generate the behavior.

SARSA converges to an optimal policy if all state-action pairs are visited an infinite number of times and the policy converges in the limit to the greedy policy ($\varepsilon = 1$ t).

SARSA Update Rule: $Q(S, A) \leftarrow Q(S, A) + \alpha [R + \gamma Q(S', A') - Q(S, A)]$

Advantages:

- On-Policy (This is beneficial when the agent is required to learn in an environment where it cannot easily explore or when the agent needs to learn a policy that maximizes a certain metric, such as safety or fairness.).
- Converges for on-policy learning.
- Suitable for problems with stochastic policies.

Disadvantages:

- Can be slow to converge (SARSA may converge slower than Q-learning because it updates its estimates based on the current policy, which can lead to suboptimal learning in the early stages of training).
- On-policy learning can be less efficient (on-policy learning can be less effective than off-policy because the agent needs to balance exploration and exploitation).

Q-Learning: Q-learning, on the other hand, is an off-policy method that updates the Q-value of the state-action pair using the maximum Q-value of the next state-action pair.

In Q-learning the learned action-value function, Q directly approximates the optimal action-value function, independent of the policy being followed.

Q-Learning Update Rule: $Q(S, A) \leftarrow Q(S, A) + \alpha [R + \gamma \max (Q(S', a')) - Q(S, A)]$

Advantages:

- Off-policy (This is beneficial when the agent needs to learn in an environment where exploration is important or when the agent needs to learn a policy that maximizes a certain metric, such as reward or speed).
- Can learn from experience generated by any policy.
- Converges to the optimal policy.

Disadvantages:

- May overestimate the value of some state-action pairs.
- Unsuitable for problems with stochastic policies (It updates its estimates based on the maximum Q-value of the next state-action pair, which can lead to suboptimal learning when exploration is necessary).

Key Features of Tabular Methods:

- **They are model-free:** While using tabular methods the user is not required to have any prior knowledge of the underlying dynamics regarding the environment**.** They can simply rely on experience to update their value estimates.

- **They are on-policy or off-policy:** Tabular methods can be either on-policy or off-policy, depending on whether they update the value of the policy being followed or the optimal policy.
- **They are simple and computationally efficient:** Tabular methods are relatively simple and easy to implement, making them a popular choice for small-scale problems with discrete state and action spaces.

2.) Results

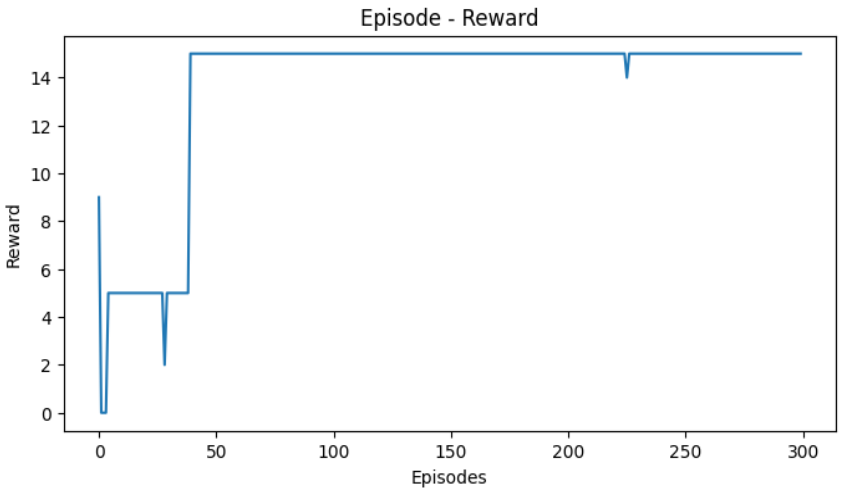
Applying SARSA to solve the environment defined in Part-1:

Hyperparameters:

- Alpha = 0.27
- Gamma = 0.97
- Epsilon = 0.89
- Decay_Factor = 0.9
- Number of Episodes (num_Episodes) = 300

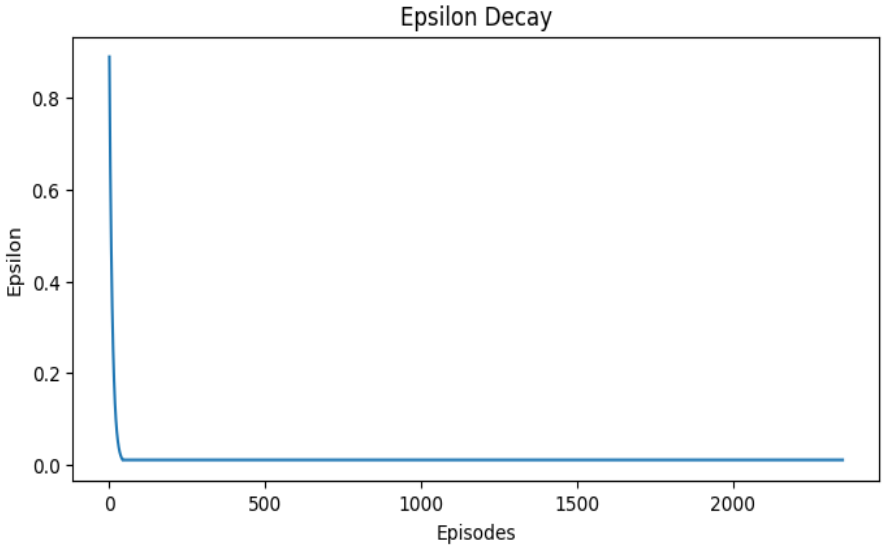
Visualizations:

- **Rewards per Episode**



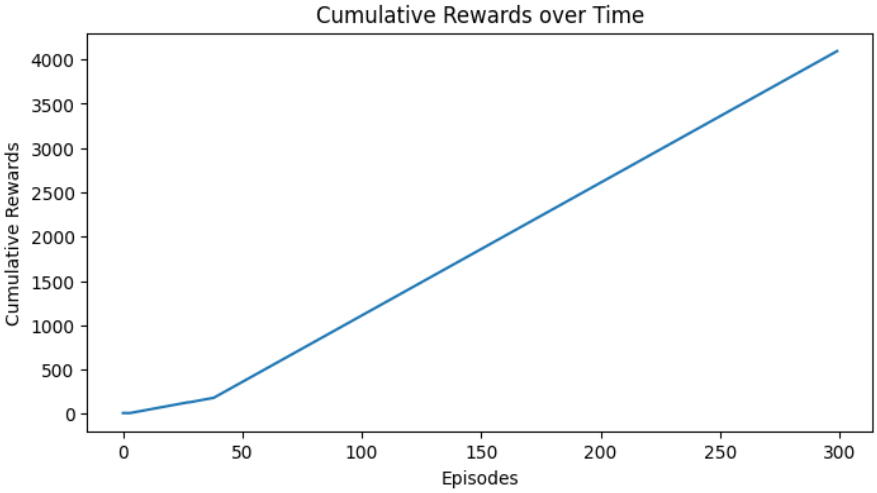
We can observe that the agent has successfully reached the highest reward point, that is 14. It has Explored in the over different reward values in the Initial episodes and reached the final reward point after the 50^{th} episode and remained constant for the rest of the episodes.

- **Epsilon Decay per Episode:**



Initially, we started with an epsilon value of 0.89 and as we can observe the epsilon decay value has decreased over the initial episodes to 0.01 and remained constant over the remaining episodes.

- **Cumulative Reward per Episode:**



Cumulative Rewards over Time is the sum of rewards obtained after each episode. The sum of all the reward points of each episode is greater than 4000 reward points according to our graph.

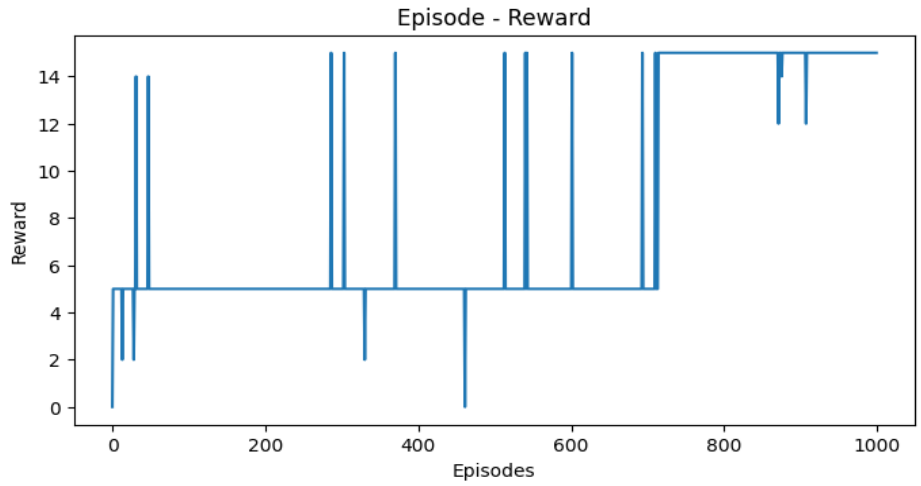
Applying Q-learning to solve the Stochastic Environment defined in Part-1:

Hyperparameters:

- Alpha = 0.1
- Gamma = 0.9
- Epsilon = 0.5
- Decay_Factor = 0.01
- Number of Episodes (num_Episodes) = 1000

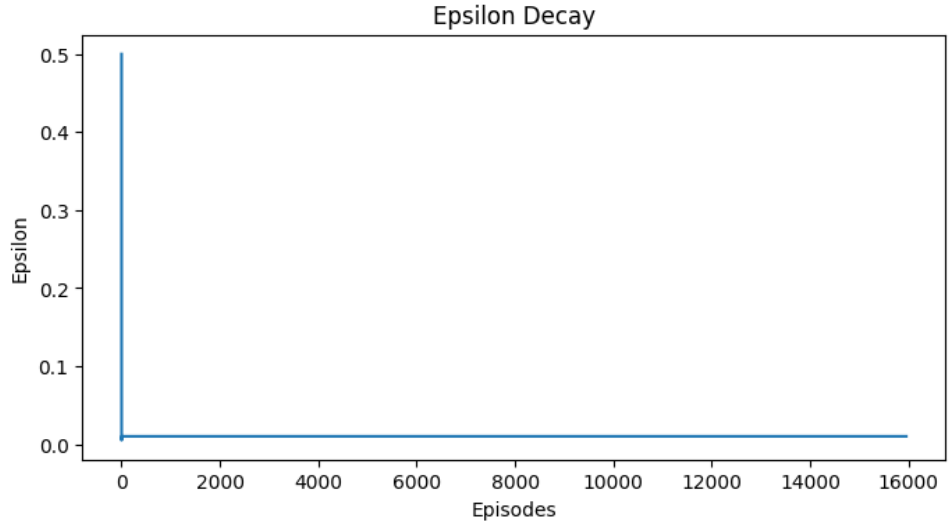
Visualizations:

- **Rewards per Episode:**



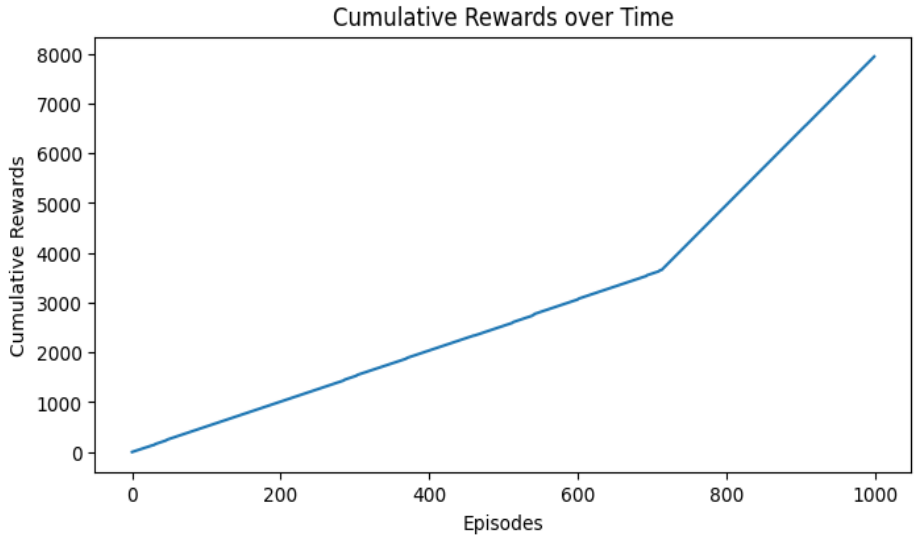
Unlike SARSA, in Q-Learning the Agent Explored a lot over the initial episodes and the agent explored mainly on the positive rewards which is the key feature of Q-Learning Algorithm. Also, the agent reached the largest reward point, that is 14.

- **Epsilon decay per Episode:**



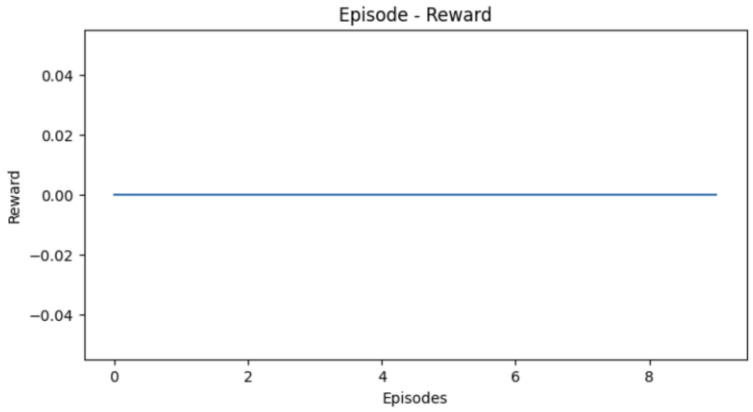
The Epsilon value started at 0.5 and while using this algorithm the epsilon value did not reduce gradually. It suddenly reached 0.01 in the initial timestamps only.

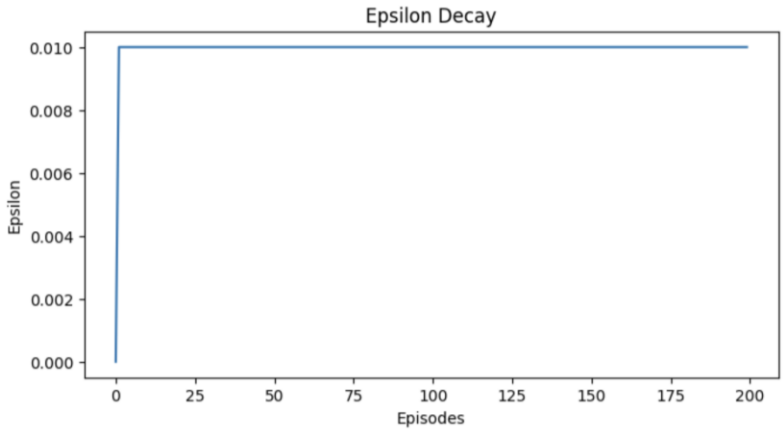
- **Cumulative Reward per Episode:**

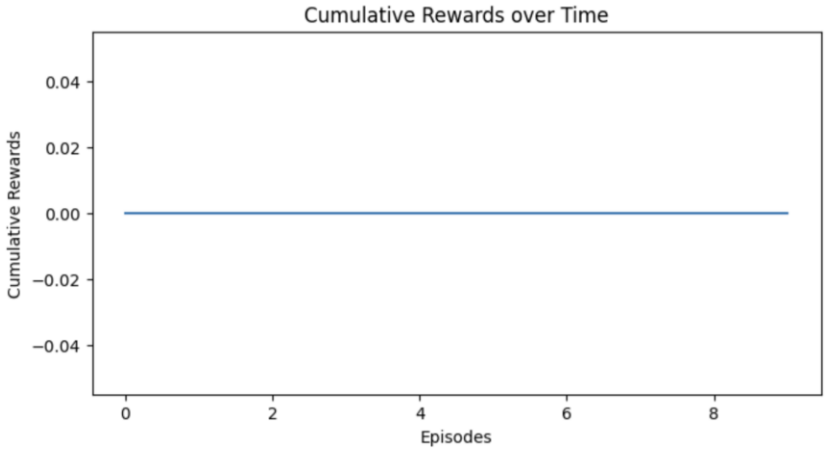


As we know in Q-Learning the agents explores more over the initial episodes and that too more on the positive rewards we have obtained 8000 Cumulative Reward Points which double the cumulative rewards obtained while using SARSA algorithm.

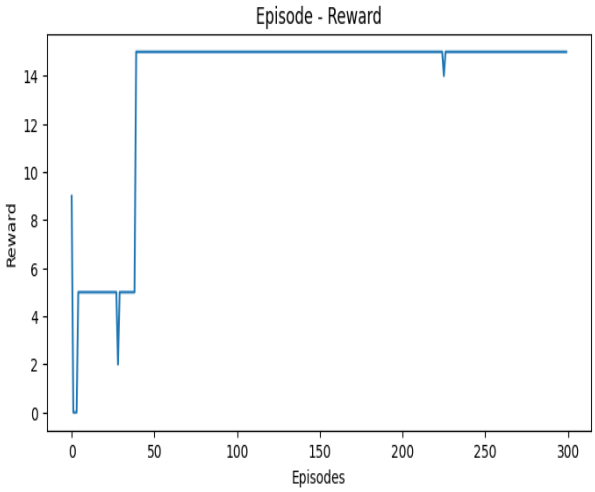
Plots for greedy actions:



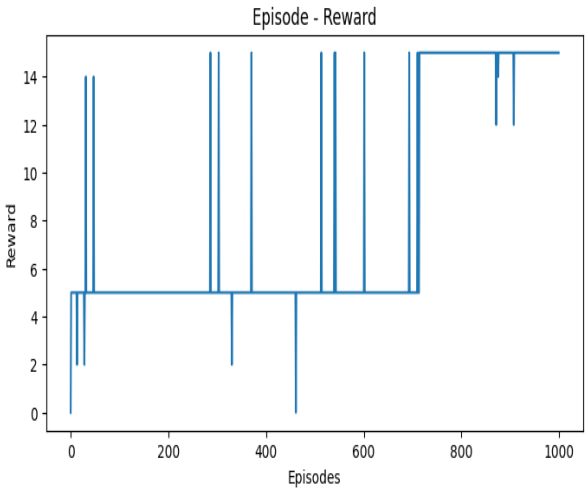




3.) Comparison of SARSA and Q-Learning on same environment:



SARSA



Q-Learning

The Agent has reached the highest reward point of 15 in both the SARSA and Q-Learning Algorithms but in Q-Learning the agent explored more on the positive rewards over the episodes compared to the SARSA algorithm. This can be clearly observed with the rewards graph of both the algorithms. The Cumulative Rewards are doubled while using Q-Learning algorithm which implies that the agent explored more on the positive rewards while using Q-learning.

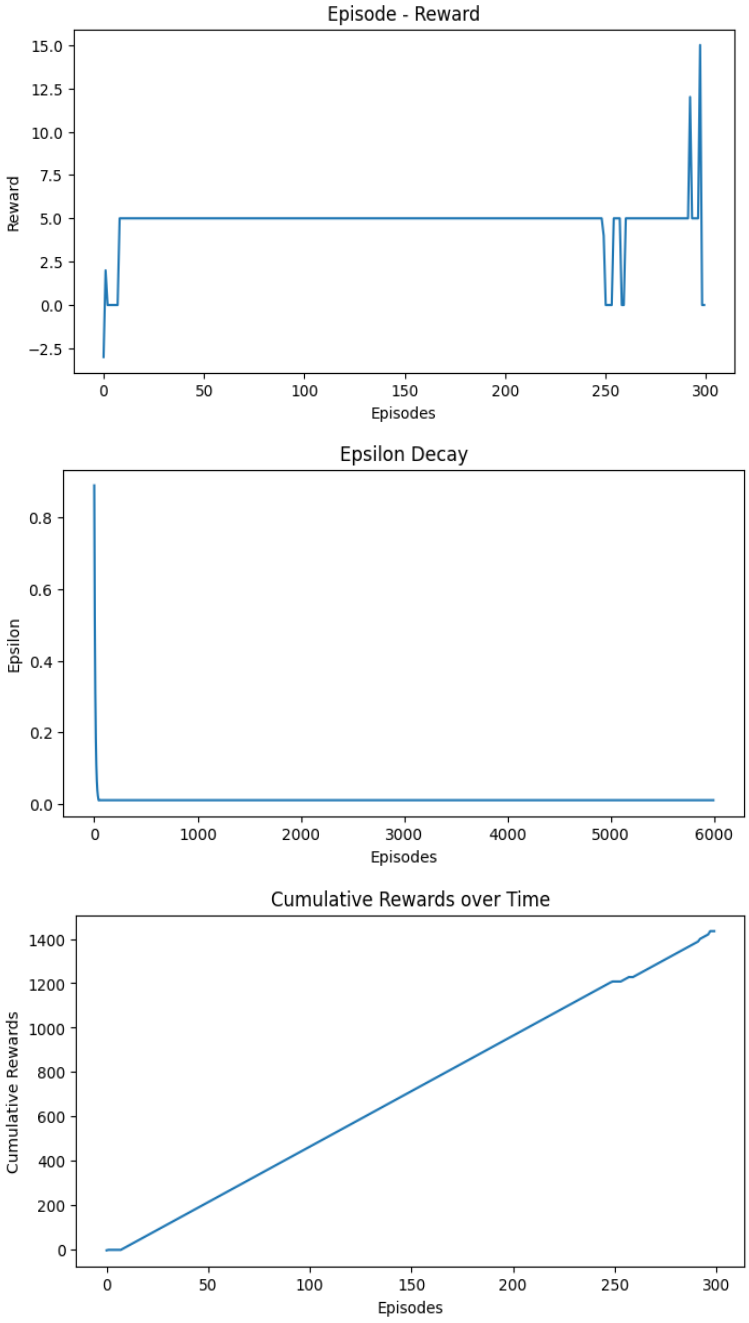
As we know unlike SARSA, Q-learning is an off-policy algorithm, which means that it learns the value of the optimal policy, regardless of the policy it is currently following. This can be beneficial when the agent needs to learn in an environment where exploration is important or when the agent needs to learn a policy that maximizes a certain metric, such as reward or speed.

4.) Hyperparameter Tuning:

Part-2(SARSA)

- **Setup-1:**

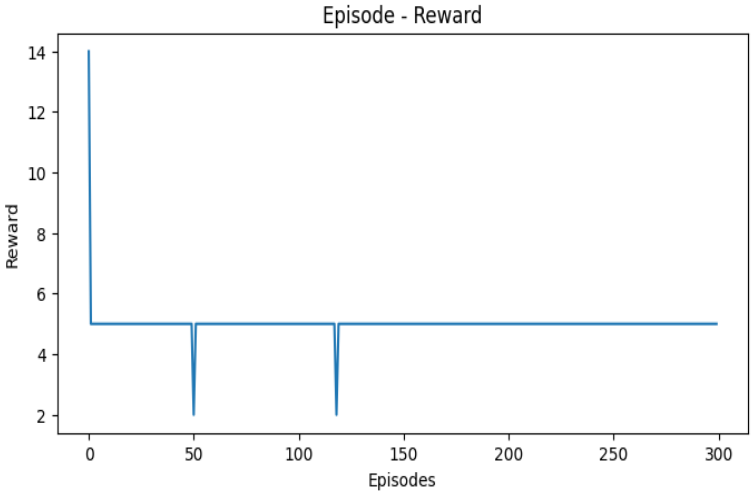
  Discount Factor: 0.99
  Epsilon Decay: 0.9

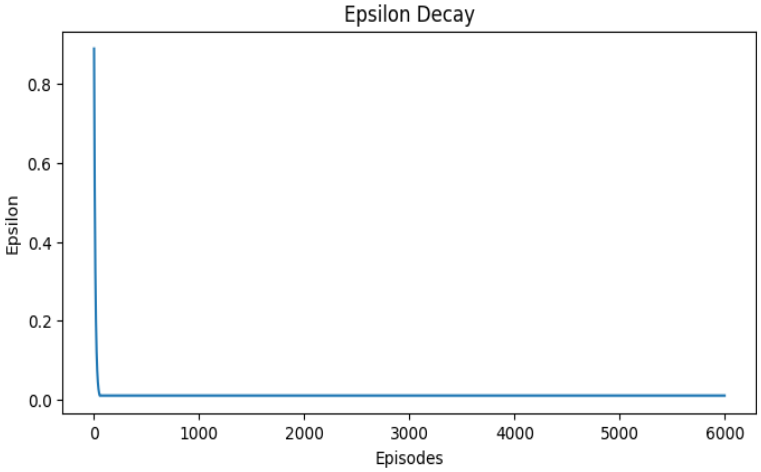


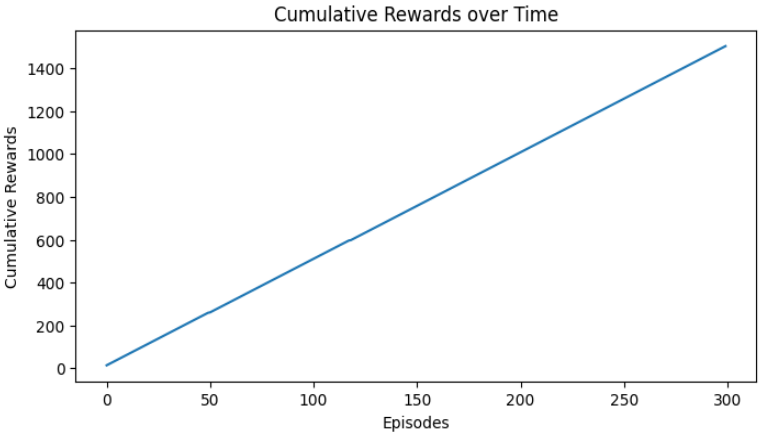
In this setup the model reached the maximum reward point before the final episode but has returned to 0 reward in the last episode. The Cumulative Reward points are also only 1400 which is quite less compared to our base SARSA algorithm. This depicts that the agent has not reached the goal state in the final episodes. This Implies that the agent did not explore more and did not learn on the positive rewards.

- **Setup-2:**

  Discount Factor: 0.95
  Epsilon Decay: 0.92



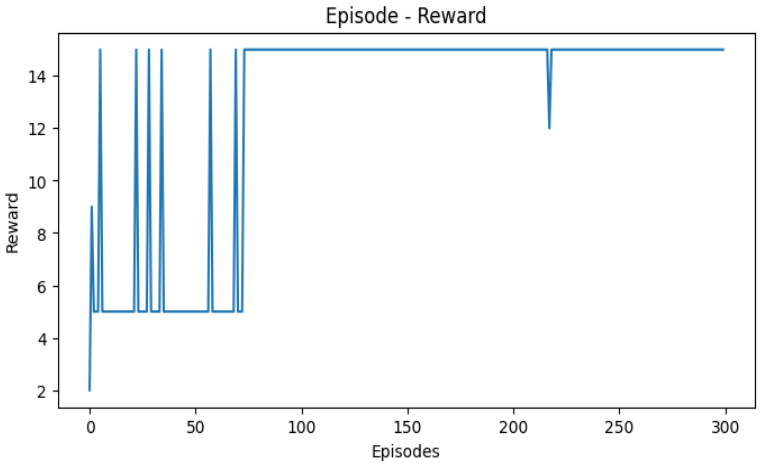


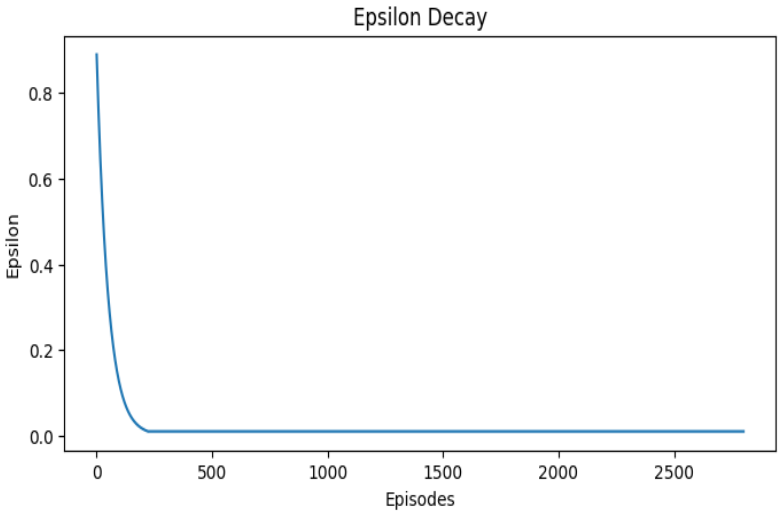


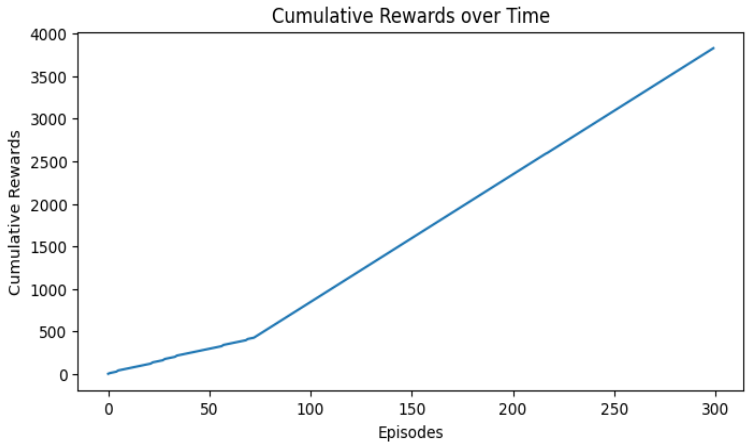
In this setup the model started from the largest reward point and ended up at 5 rewards in the last episode. This depicts that the agent has not reached the goal state in the final episodes. The Agent always Explored and learned on the negative rewards which is not as desired. The Cumulative rewards (1500) of this setup are somewhat like that of 1st setup.

- **Setup-3:**

Discount Factor: 0.96
Epsilon Decay: 0.98







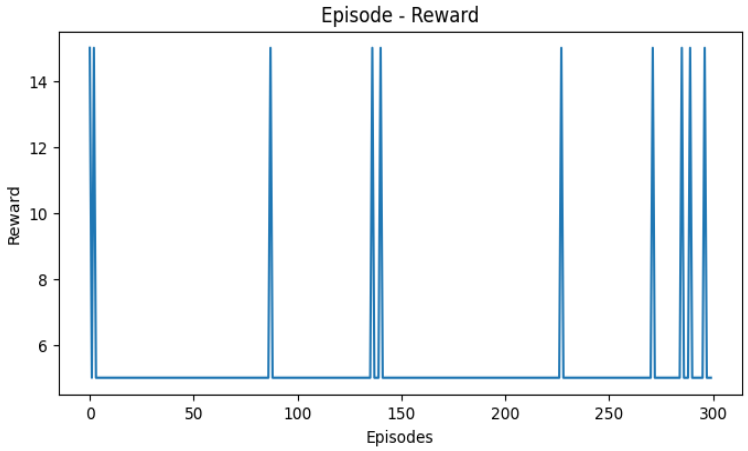
In our 3rd Setup the agent explored very well over the positive rewards in the initial episodes and reached the maximum reward point of 15. The Epsilon Decay also started from 0.98 and decreased gradually over the episodes. We have also obtained highest rewards using this setup, that is we have nearly obtained 4000 cumulative rewards. This implies that the Agent learned more from positive rewards which is expected. This is our Best Setup out of all the setups applied in SARSA algorithm.

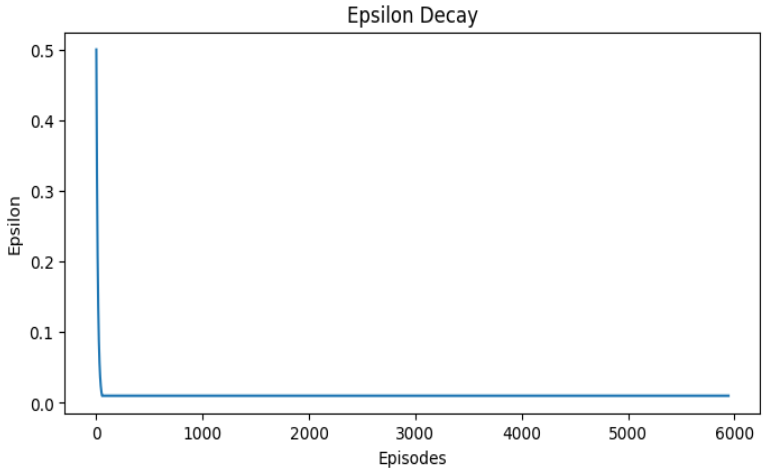
Best Setup:

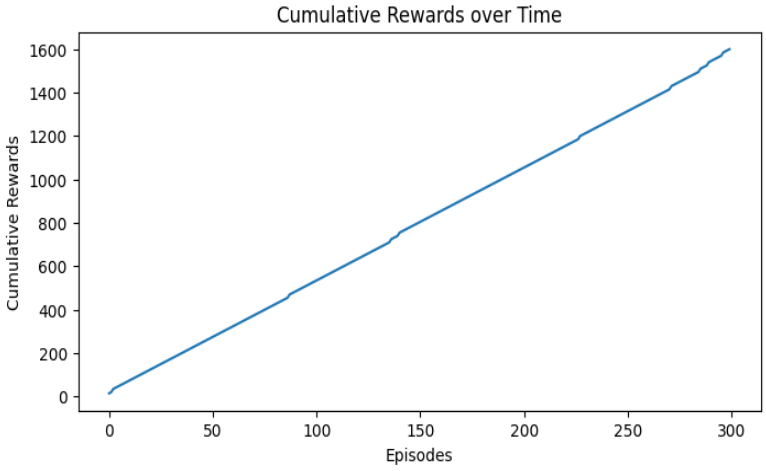
When we tuned different hyperparameters we can observe that our 3rd Setup (Discount Factor: 0.96, Epsilon Decay: 0.98) produced best results. The agent started accumulating positive rewards over time and reached the highest reward (15) greater than compared to all the other 3 setups.

Part-3 (Q-Learning)

- **Setup-1:**
  Discount Factor: 0.97
  Epsilon Decay: 0.93

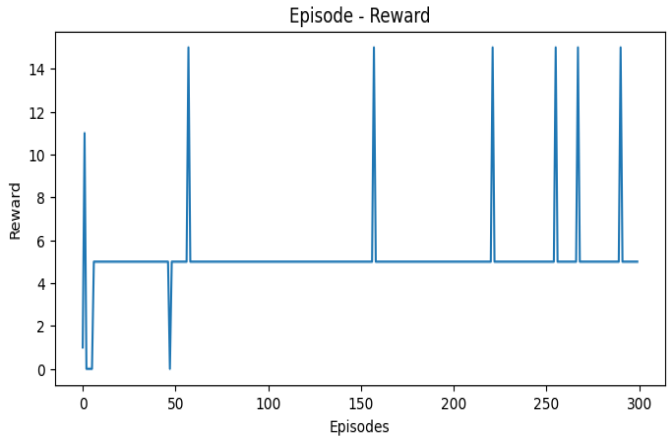


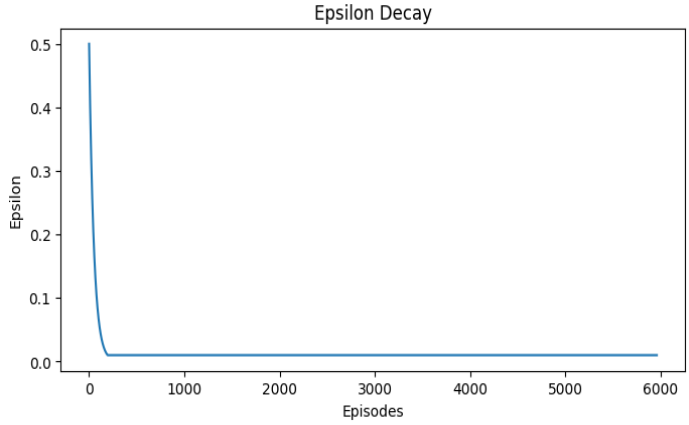


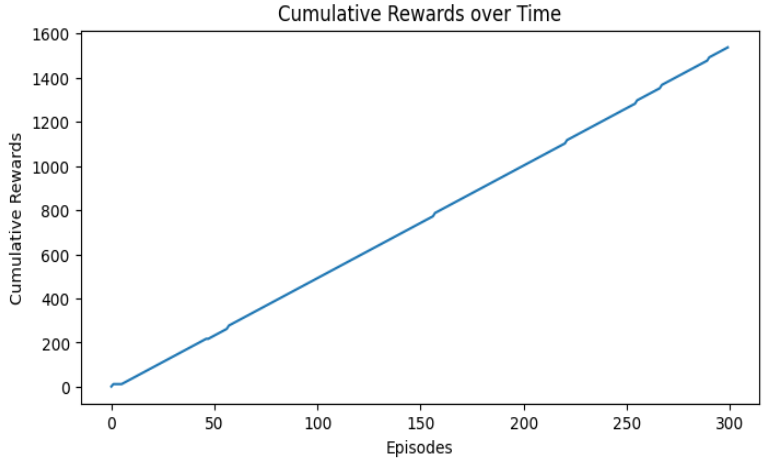


In this setup the agent explored throughout all the episodes and reached the maximum reward point in some episodes but finally ended up at 5 rewards in the last episode which is not as expected. This depicts that the agent has not reached the goal state in the final episodes. The cumulative Rewards are also only around 1600 which is very less especially in Q-Learning Algorithm.

- **Setup-2:**
  Discount Factor: 0.98
  Epsilon Decay: 0.98

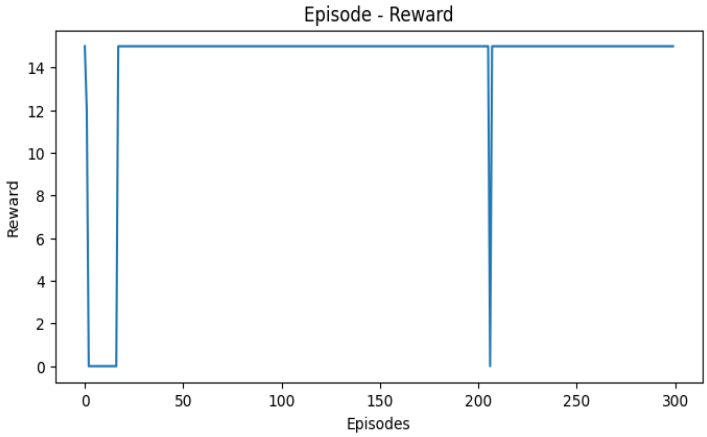


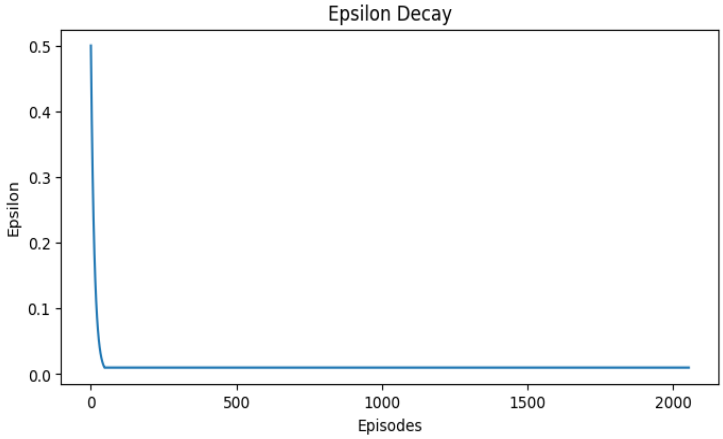


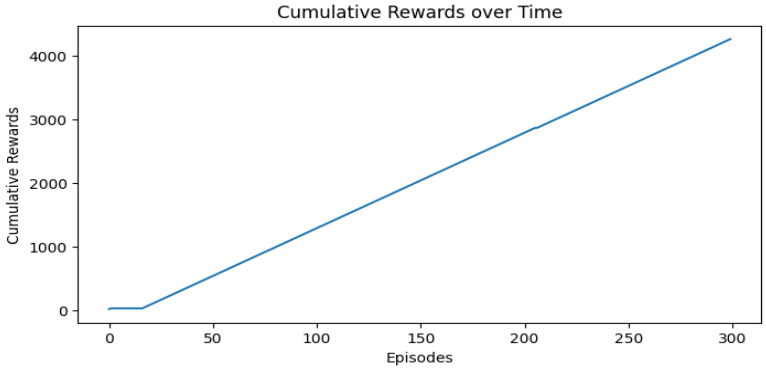


Like the first setup the agent reached the maximum reward point over some episodes but finally ended up at 5 rewards in the last episode which is not expected. This depicts that the agent has not reached the goal state in the final episodes. And the cumulative rewards are around 1600 which is less for a Q-Learning Algorithm.

- **Steup-3:**
  Discount Factor: 0.97
  Epsilon Decay: 0.92







In our 3rd Setup the Agent reached the highest reward point in the final episodes and in this setup, we have obtained cumulative rewards more than 4000 which is as desired. Therefore, this is our best setup compared to all the other setups that we used in our Q-Learning Algorithm.

Best Setup of Part-3:

When tuning different hyperparameters our 3rd Setup (Discount Factor: 0.97, Epsilon Decay: 0.92) reached the largest reward point, that is 15.

Contribution Table:

Team Member	Assignment Part	Contribution (%)
Sai Teja Sankarneni	All Parts	50%
Vinitha Vudhayagiri	All Parts	50%

References:

- https://piazza.com/class_profile/get_resource/ldezxxu3kf25wf/lgdwil4uvs96ct
- https://www.kaggle.com/code/nagakiranreddy/reinforcement-learning-sarsa-on-grid-world-env
- https://towardsdatascience.com/create-your-own-reinforcement-learning-environment-beb12f4151ef
- https://towardsdatascience.com/reinforcement-learning-temporal-difference-sarsa-q-learning-expected-sarsa-on-python-9fecfda7467e
- https://www.geeksforgeeks.org/sarsa-reinforcement-learning/
- https://www.learndatasci.com/tutorials/reinforcement-q-learning-scratch-python-openai-gym/