CSE 574 - Introduction to Machine Learning

Assignment 4: Reinforcement Learning

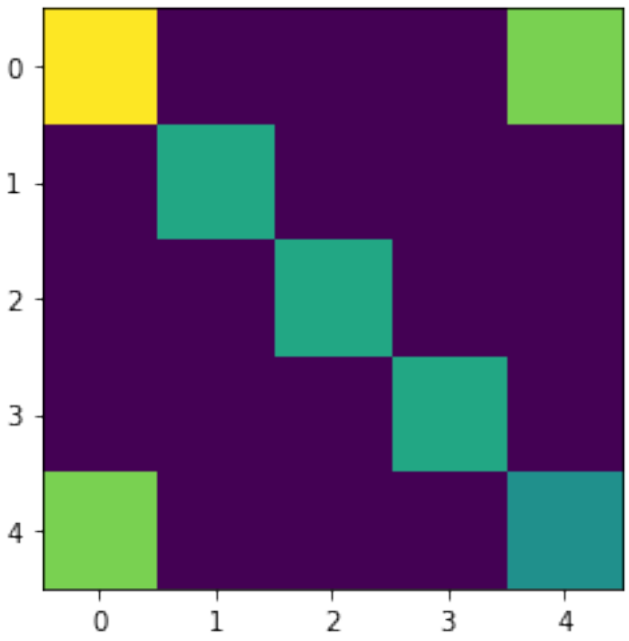
Digvijay Patil (50428152)
Pratik Malani (50416266)

Part 1:

1. Describe the environment that you defined. Provide a set of actions, states, rewards, main objective, etc.

Actions – {0: Down, 1: Up, 2: Right, 3: Left}
States – {Initial - [0,0], Cheese - [1,1], Jetpack - [2,2], Chocolate - [3,3], Coyote - [4,0], Gun - [0,4], Goal - [4,4]}
Rewards – {-1,-10,-5,5,15,10,100}
Main Objective: To get the Agent (Road Runner) to the final destination (Tunnel) avoiding obstacles that results in harm.

2. Provide visualization of your environment.



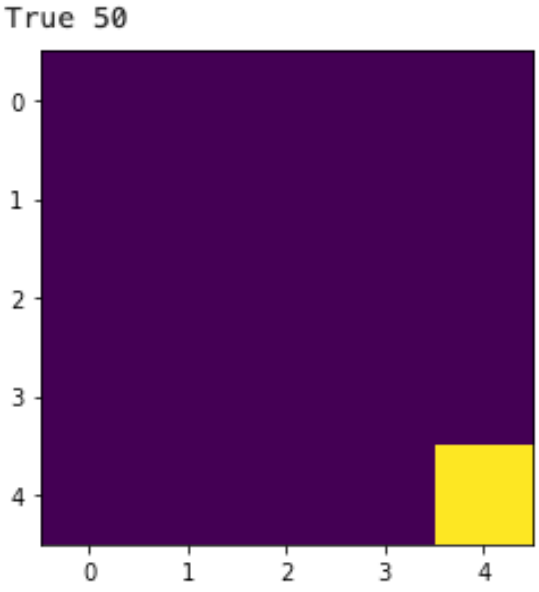
3. Safety in AI : Write a brief review explaining how you ensure the safety of your environment.

The observation model is used for simulation purposes. The safe-RL algorithm derives a policy in a fully observable environment.
Such mathematical formulation provides a principle frame- work to model multiple agents interacting, uncertain behaviors, and limited perception of the environment. However, solving for the optimal strategy to navigate in such an environment would be intractable due to an exponential increase in the number of states with the number of agents considered. For this reason, we will first focus on a sub-task in a simpler scenario, using offline methods. In such environments we turn a test model to see which likely steps provides an optimal outcome by looking at it as a subtask. With this recursive approach, we can in turn ensure safety by solving the bigger problem.

Part 2:

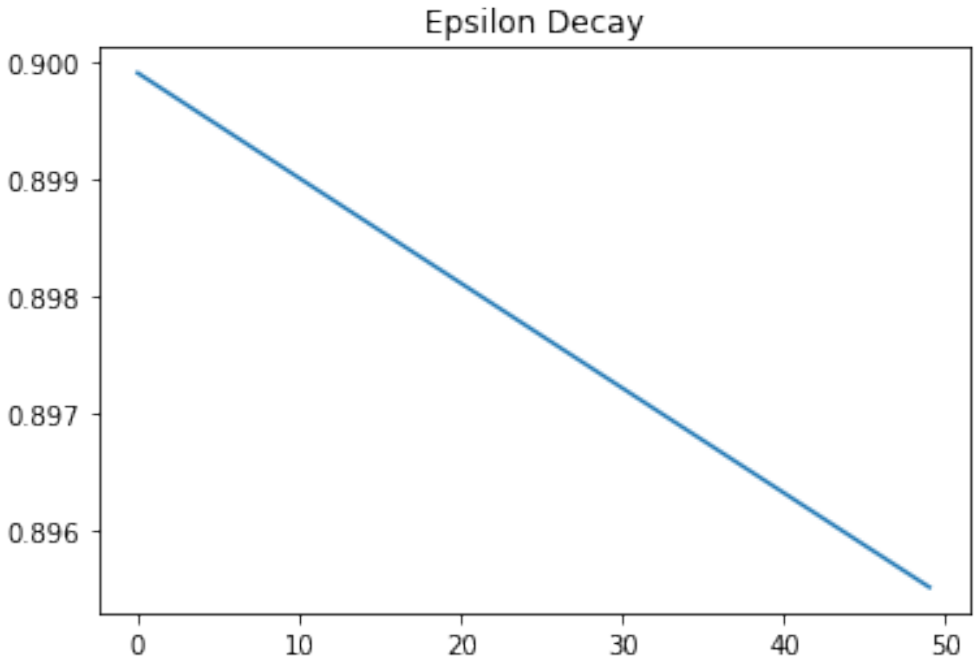
1. **Show and discuss the results after applying SARSA to solve the environment defined in Part I.**

We ran the SARSA Algorithm for 50 episodes, epsilon value at 0.9, discount factor at 0.9 and learning rate 0.01. On the first run, the total rewards accumulated was 724 and the Performance was 14.48 rewards per episode.

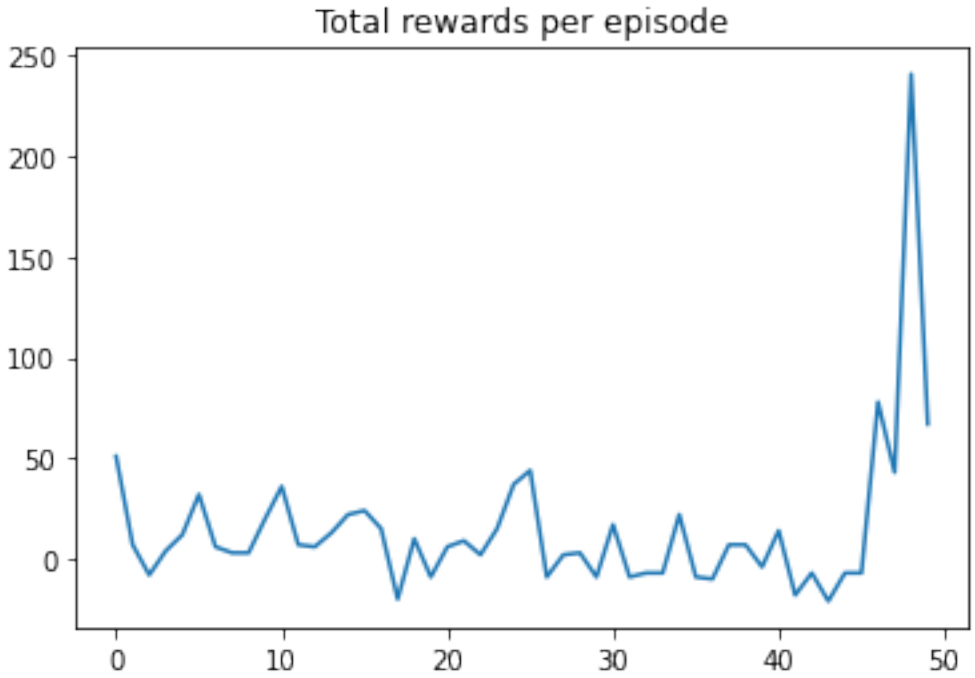



```
Total Rewards -  724
Performance - Total Rewards/Number of Episodes 14.48
```

2. Provide a plot for epsilon decay.

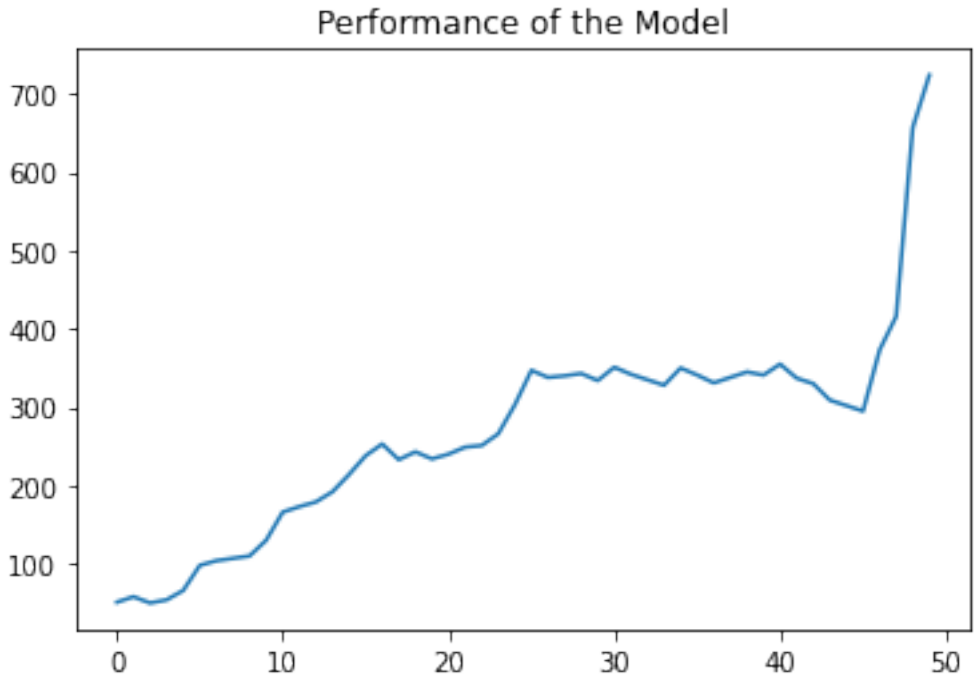


3. Provide the evaluation results. Run your environment for at least 10 episodes, where the agent chooses only greedy actions from the learnt policy. Plot should include the total reward per episode.



4. Give your interpretation of the results.

The performance of the model moves in an upward linear fashion where the agent is consistently learning the right path with maximum rewards. At the 45th episode, it takes a steep improvement to find the optimal path.



5. Briefly explain these tabular methods: SARSA and Q-learning. Provide their update functions and key features.

SARSA and Q-learning are two reinforcement learning methods that do not require model knowledge, only observed rewards from many experiment runs. Unlike MC which we need to wait until the end of an episode to update the state-action value function Q(s,a) SARSA and Q-learning make the update after each step.

In both cases the policy followed by the agent ϵ-greedy which is important for exploration.

In both methods, during each episode, from a current state s we take an action a from s to another new state s', observing the reward r. The action a is taken following the current ϵ-greedy policy. Now we update Q(s,a).

In SARSA, this is done by choosing another action a' following the same current policy above and using r+γQ(s′,a′) as target.

Sarsa (on-policy TD control) for estimating $Q \approx q_*$

Initialize $Q(s,a), \forall s \in \mathcal{S}, a \in \mathcal{A}(s)$, arbitrarily, and $Q(\textit{terminal-state}, \cdot) = 0$
Repeat (for each episode):
  Initialize S
  Choose A from S using policy derived from Q (e.g., ϵ-greedy)
  Repeat (for each step of episode):
    Take action A, observe R, S'
    Choose A' from S' using policy derived from Q (e.g., ϵ-greedy)
    $Q(S,A) \leftarrow Q(S,A) + \alpha\big[R + \gamma Q(S',A') - Q(S,A)\big]$
    $S \leftarrow S'$; $A \leftarrow A'$;
  until S is terminal

In Q-learning, this is done by choosing the **greedy action a'** i.e the action that maximize the Q-value function at the new state Q(s', a):

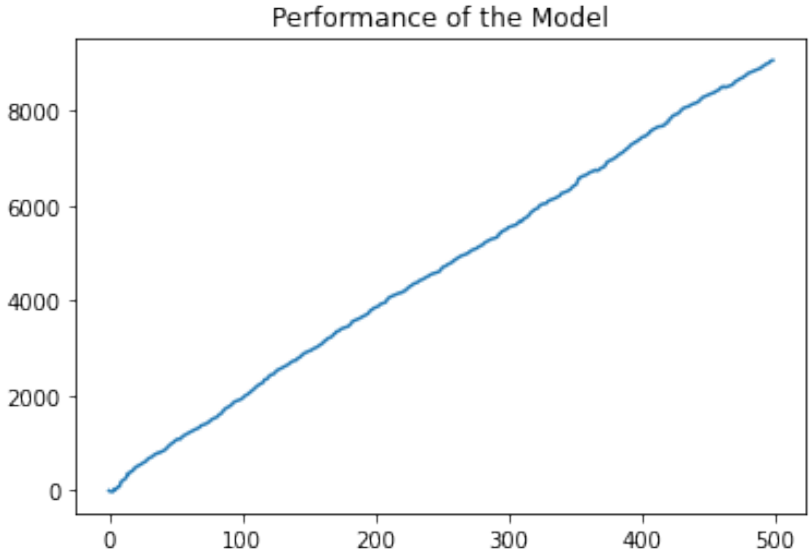
Q-learning (off-policy TD control) for estimating $\pi \approx \pi_*$

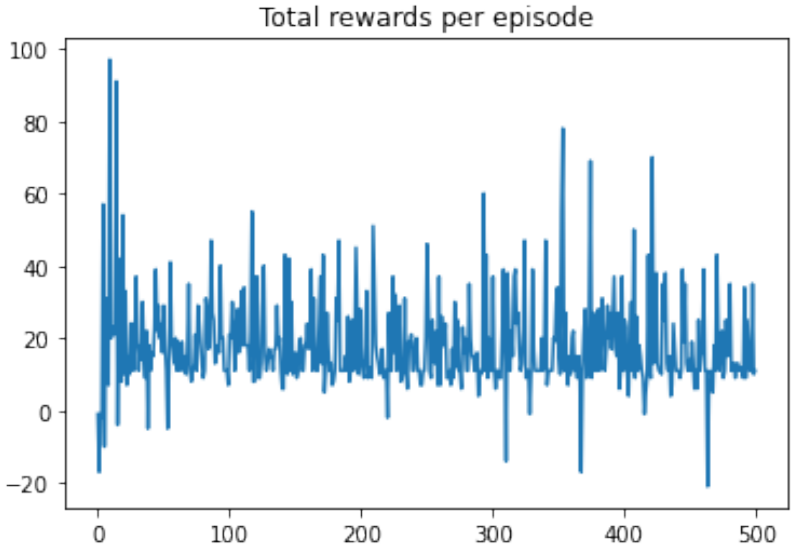
Initialize $Q(s,a), \forall s \in \mathcal{S}, a \in \mathcal{A}(s)$, arbitrarily, and $Q(\textit{terminal-state}, \cdot) = 0$
Repeat (for each episode):
  Initialize S
  Repeat (for each step of episode):
    Choose A from S using policy derived from Q (e.g., ϵ-greedy)
    Take action A, observe R, S'
    $Q(S,A) \leftarrow Q(S,A) + \alpha\big[R + \gamma \max_a Q(S',a) - Q(S,A)\big]$
    $S \leftarrow S'$
  until S is terminal

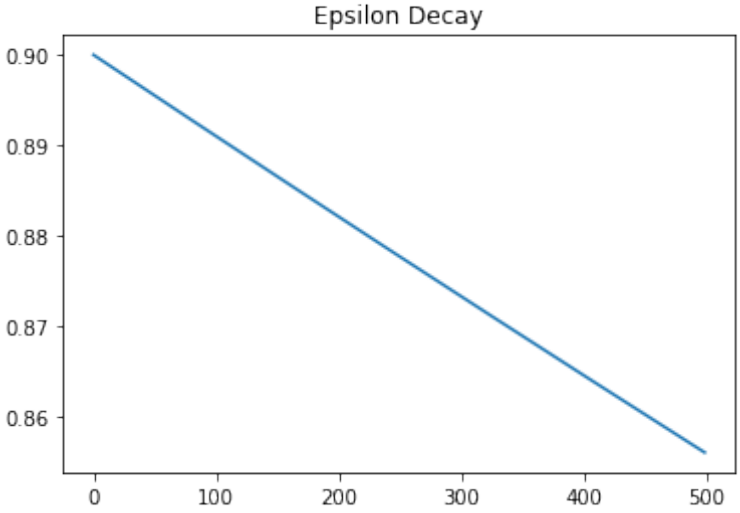
6. Provide the analysis after tuning. at least two hyperparameters from the list above. Try at least 3 different values for each of the parameters that you choose. Provide the reward graphs and your explanation for each of the results. In total you should have at least 6 graphs and your explanations. Make your suggestion on the most efficient hyperparameters values for your problem setup.

The 2 parameters tuned are No of Episodes and Decay Rate:

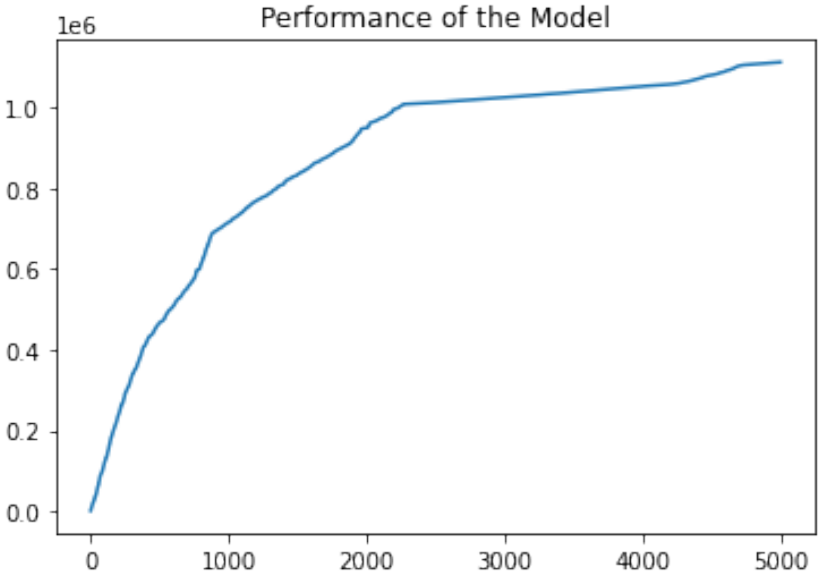
Setup 1: No of episodes - 50, Decay Rate: 0.9999 given above.
Setup 2: No of episodes - 500, Decay Rate: 0.9999:

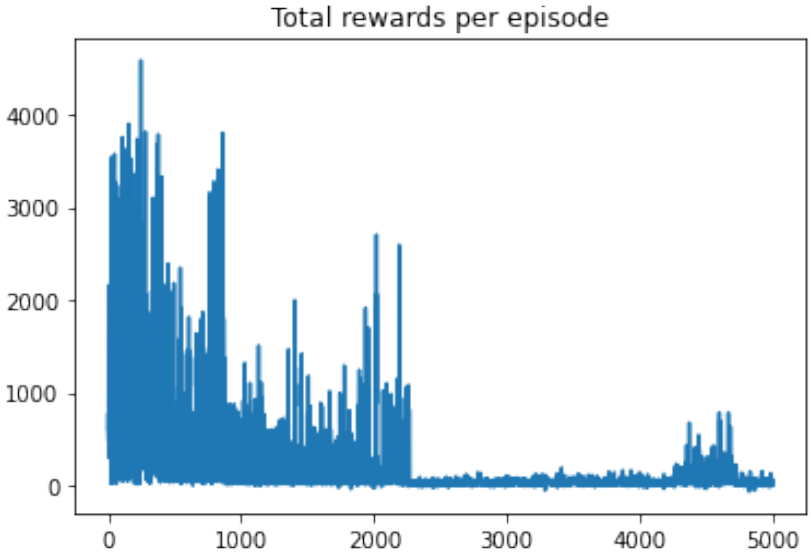


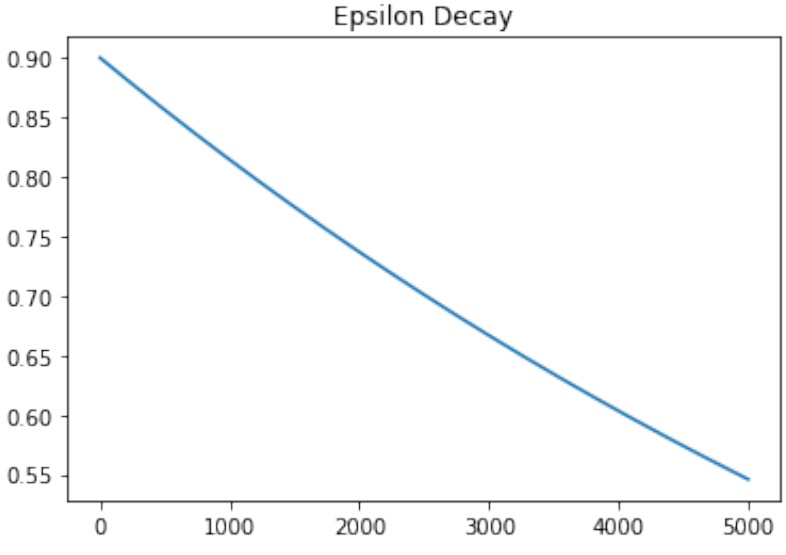




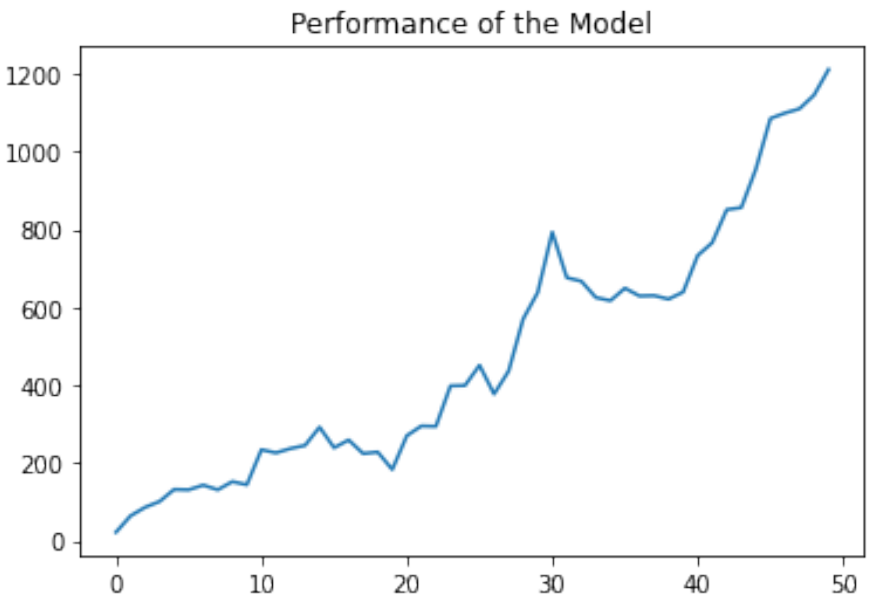
Setup 3: No of episodes - 5000, Decay Rate: 0.9999:

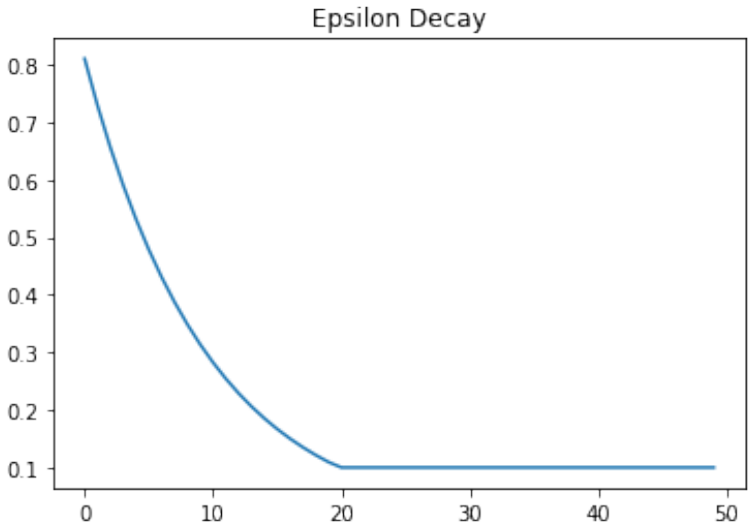




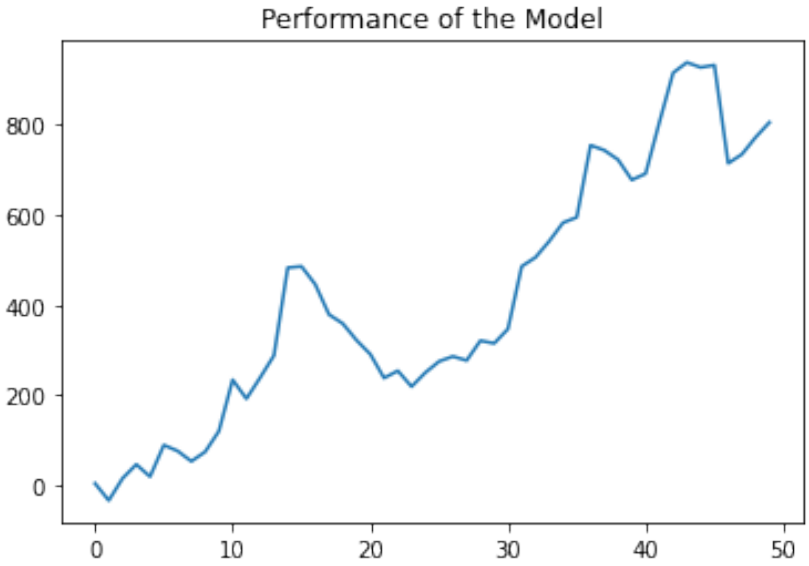


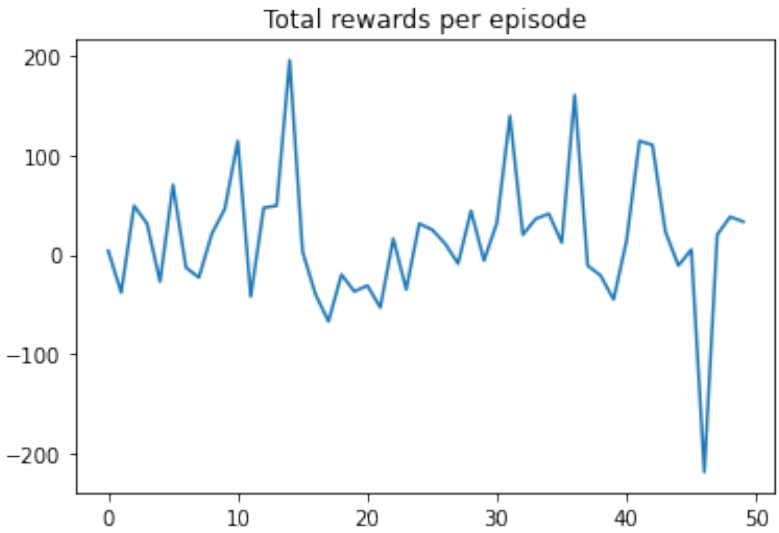
Setup 4: No of episodes - 50, Decay Rate: 0.9:

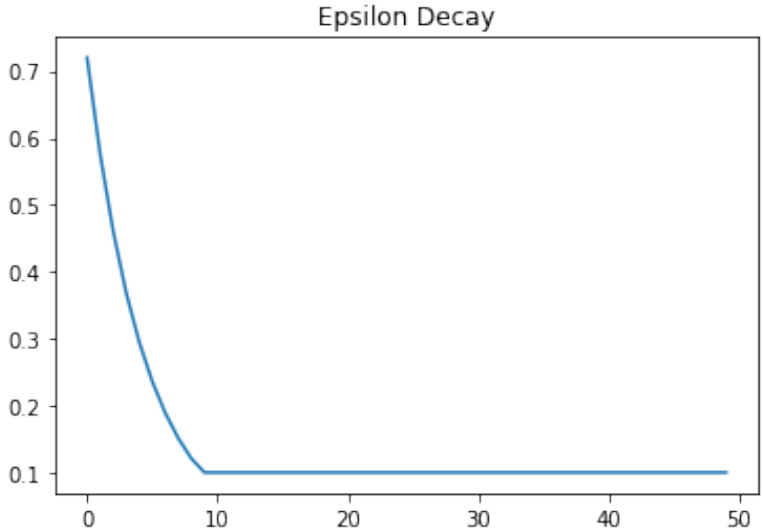




Setup 5: No of episodes - 50, Decay Rate: 0.8:







Comments: Decay rate didn't seem to have much of an effect in performance probably because episodes were low.
When number of episodes increased, model gave an upward linear improvement. When no of episodes was 5000, improvement stagnated as that was the maximum a model could perform.

Team Contribution:

Digvijay Patil	Pratik Malani
50%	50%
SARSA Implementation	Creating RL Environment
Parameter Tuning	Report and Parameter Tuning

References:
https://arxiv.org/pdf/1904.11483.pdf
Fall 2021: CSE 546 Reinforcement Learning Assignment 1.
https://en.wikipedia.org/wiki/State–action–reward–state–action
https://www.geeksforgeeks.org/sarsa-reinforcement-learning/