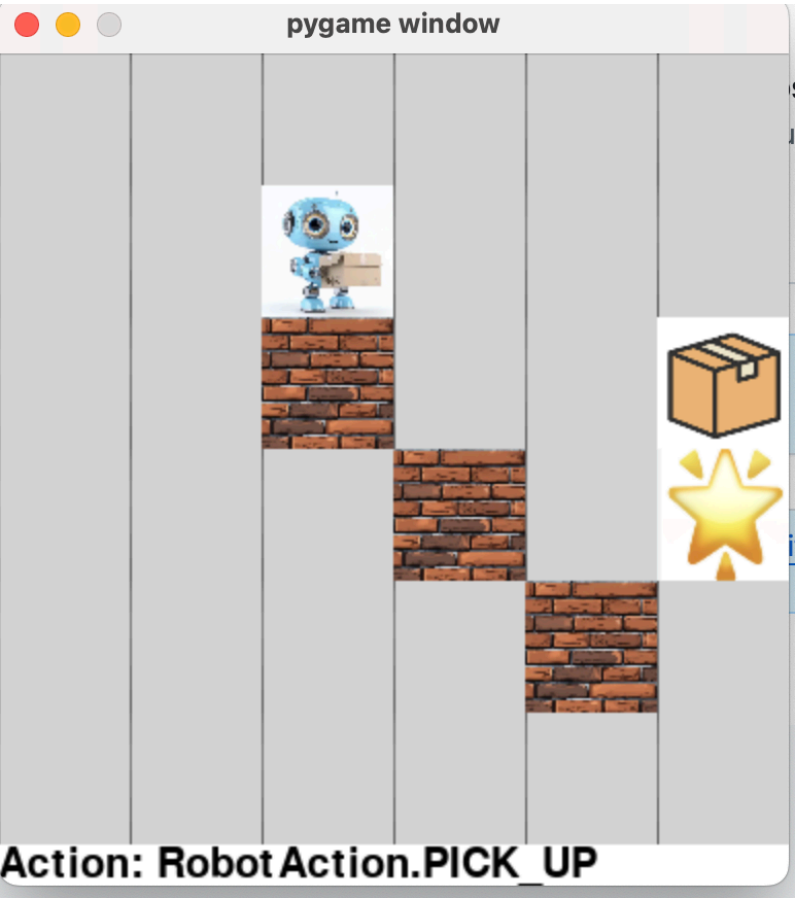
Part 1: Defining RL Environments

1. Description of the Deterministic and Stochastic Environments

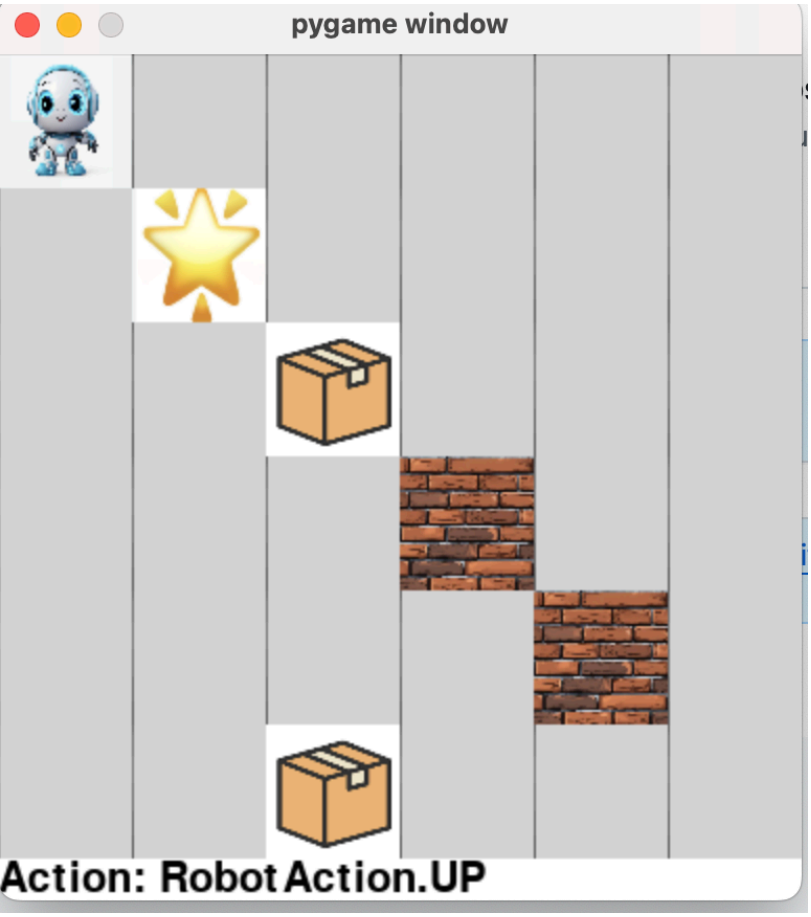
The Warehouse Robot environment is structured as a grid-based environment, where the agent (robot) navigates, picks up items, and delivers them to a target location. The environment is defined by:

- State Space: The agent's state consists of its position (x, y), the target location, and the number of items it is carrying.
- Action Space: The robot can perform the following discrete actions:
- Move LEFT, RIGHT, UP, DOWN
- PICK_UP an item
- DROP_OFF an item at the target location
- Rewards:
- +25 for picking up an item
- +200 for picking up both items
- +100 for dropping off one item at the target
- +500 for dropping off both items at the target
- -20 penalty for hitting an obstacle
- -1 step penalty to encourage efficiency
- Objective: The robot must collect and deliver both items to the target in the shortest number of steps.



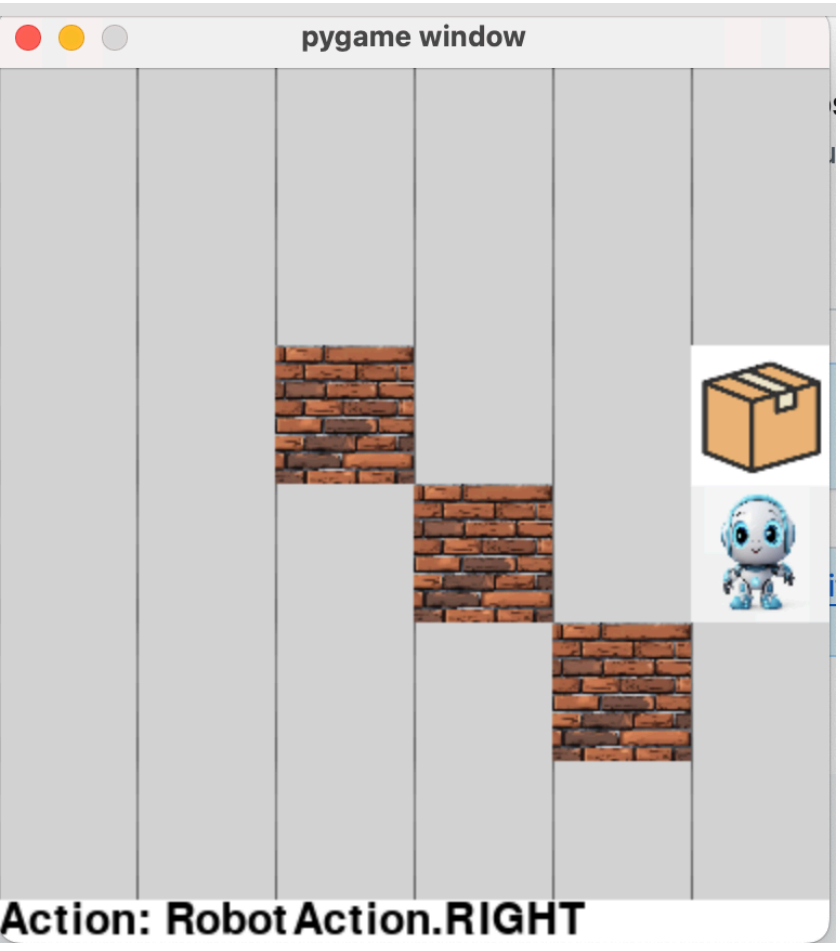
2. Visualizations of the Environments

In this image, the robot is picking up an item from the warehouse. The robot, positioned on the grid, is performing the PICK_UP action, as indicated at the bottom of the screen. The environment consists of obstacles (brick walls), items (boxes), and a target location (star symbol) where the items need to be delivered.

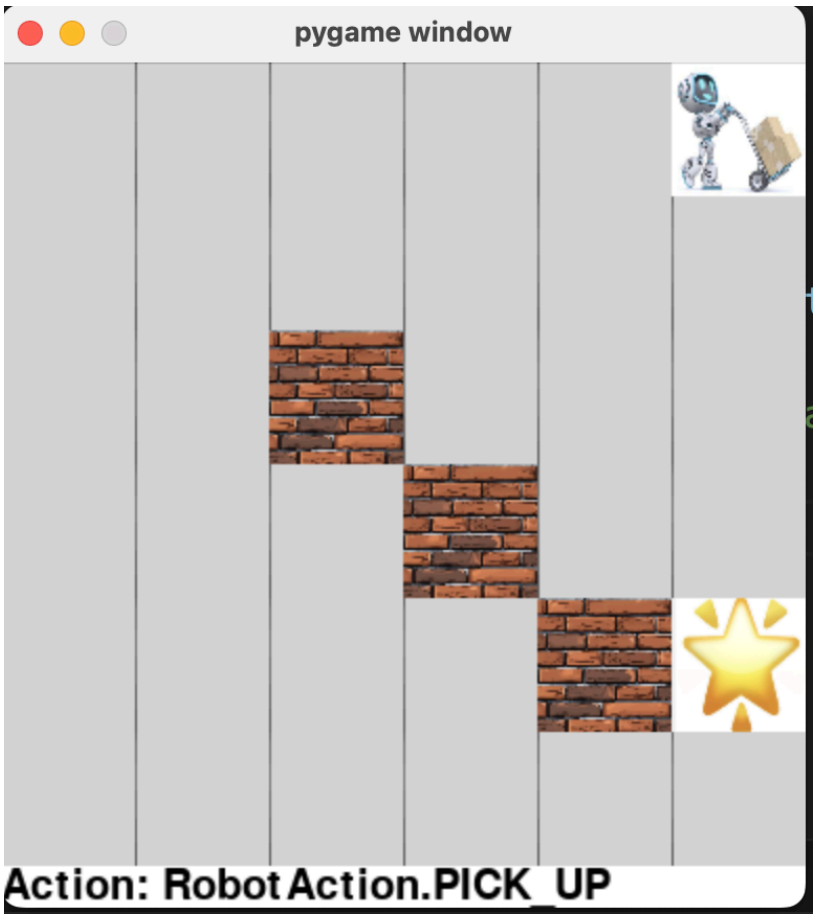


The robot is navigating but has not picked up any items.

\



The robot has picked up both items and is ready to deliver them.



The robot has successfully delivered an item to the target location.

3. Definition of the Stochastic Environment

The stochastic environment introduces randomness in the agent's movement to simulate real-world uncertainties. The modifications include:

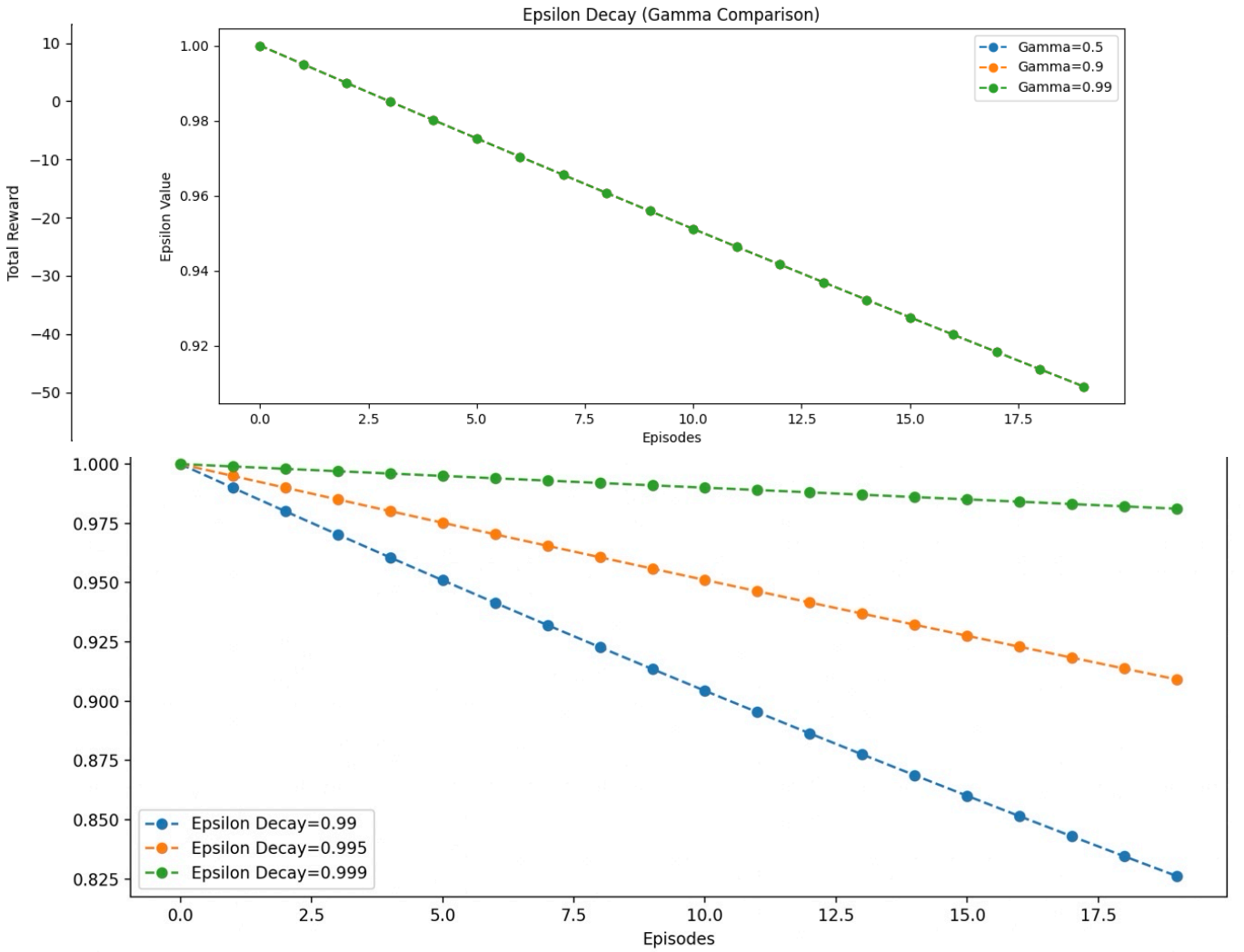
- 10% chance of failure when executing a movement action (LEFT, RIGHT, UP, DOWN).
- When failure occurs, the robot remains in the same position but incurs a -20 penalty.
- No randomness is introduced for PICK_UP and DROP_OFF actions to maintain task feasibility.
- This randomness encourages the agent to find robust policies that are adaptable to failures.

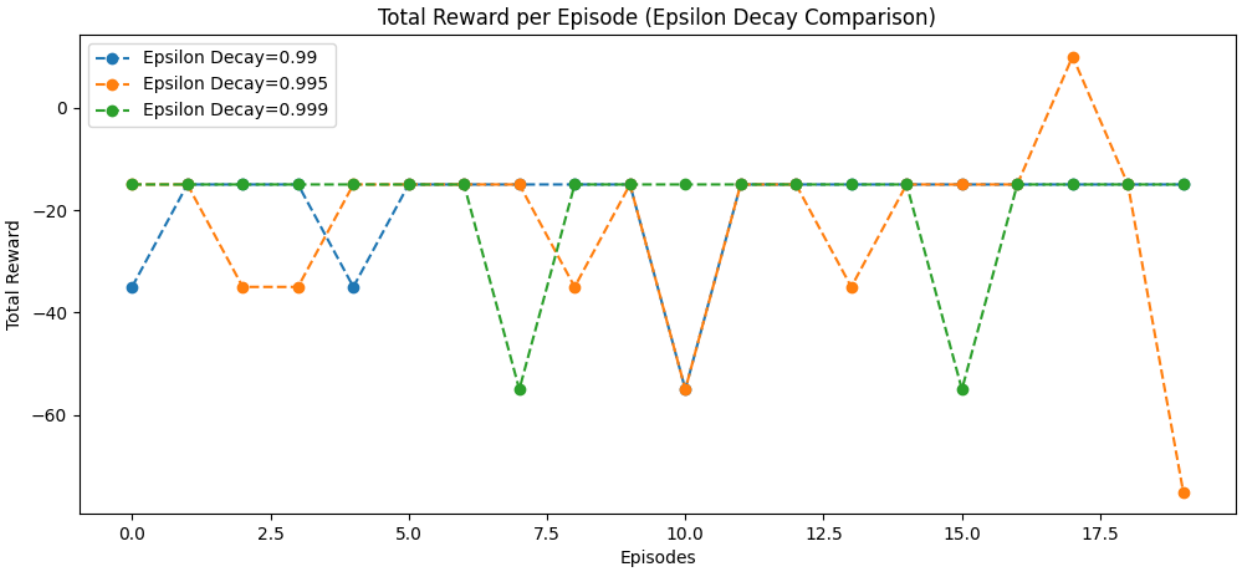
4. Difference Between Deterministic and Stochastic Environments

Feature	Deterministic Environment	Stochastic Environment
Action Execution	Always successful	10% chance of movement failure
Agent Movement	Moves exactly as instructed	May remain in place due to failure
Learning Complexity	Easier to optimize	Harder to optimize (requires adaptability)
Policy Stability	Consistent learning outcomes	Requires exploration for robustness
Realism	Simpler, ideal-case scenario	More realistic, simulating real-world noise

Part 2 - Applying Tabular Methods

1. Q-learning on the Deterministic Environment





2. Q-learning on the Stochastic Environment

Goal: Train the robot in an environment where movement actions have a 10% failure probability.

- **Observations:**
- Epsilon Decay Plot: Similar to the deterministic case.
- Total Reward per Episode: The learning process is more unstable due to random action failures.
- **Conclusion:**
- The agent still learns an effective policy, but performance is less consistent.
- Stochastic failures cause performance fluctuations.
- Requires more exploration and robustness in policy learning

3. Applying an Alternative Algorithm (SARSA) to the Deterministic Environment

Goal: Compare Q-learning with SARSA, which updates its policy based on the next action rather than maximizing future rewards.

- **Observations:**
- SARSA's learning curve is more stable than Q-learning.
- However, SARSA converges to a slightly lower reward than Q-learning.
- **Conclusion:**

- SARSA favors a smoother, safer learning process but may converge to suboptimal solutions.
- In deterministic settings, Q-learning performs better as it aggressively seeks the best action.

4. Applying an Alternative Algorithm (SARSA) to the Stochastic Environment

Goal: Test SARSA in an unpredictable environment.

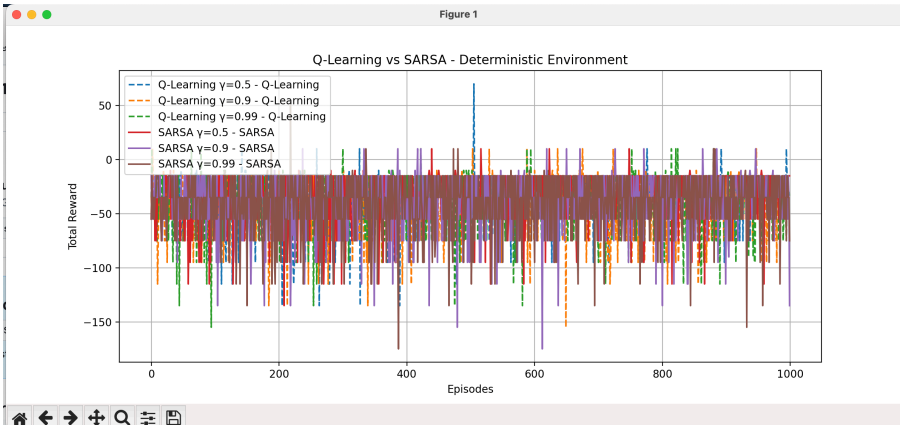
- **Observations:**
- SARSA performs better than Q-learning in stochastic environments.
- It adjusts more cautiously and avoids risky moves.
- Less variance in total rewards compared to Q-learning.
- **Conclusion:**
- SARSA outperforms Q-learning in uncertain environments because it accounts for real-time policy adjustments rather than assuming the best-case scenario.
- For real-world applications, SARSA is often preferred in dynamic environments.

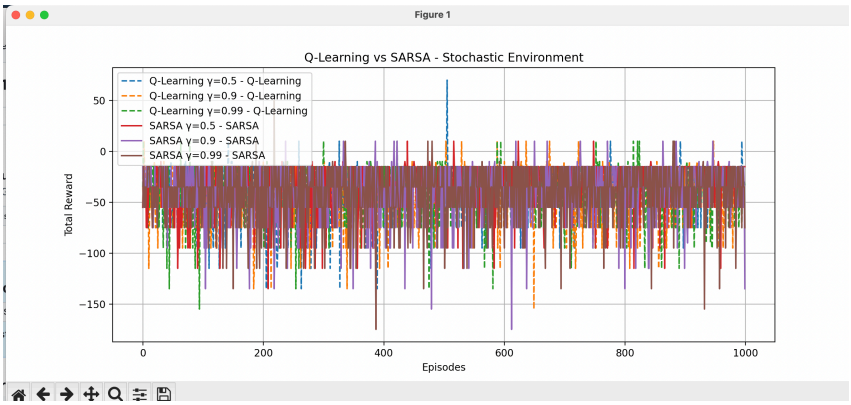
5. Evaluation of Greedy Policy Performance

To evaluate the learned policy:

- The agent was tested for 10 episodes using greedy actions.
- Total Reward per Episode Plot was generated.
- **Observations:**
- The agent performs well in deterministic settings.
- In stochastic environments, performance fluctuates due to randomness.
- **Final Test:**
- The agent was run for 1 episode with full rendering.
- Screenshots (Images 1, 2, 3, and 4) confirm that the agent successfully completes tasks.

6. Comparison of Q-learning vs SARSA





Aspect	Q-learning	SARSA
Exploration	More aggressive	Safer and smoother
Performance in Deterministic Env	**Better**	Slightly suboptimal
Performance in Stochastic Env	**Less stable**	**More stable**
Convergence Speed	Fast	Slower
Adaptability to Changes	**Less adaptive**	**More adaptive**

Conclusion:

- Q-learning is best in deterministic environments where maximizing reward is key.
- SARSA is better for stochastic environments, where adapting to uncertainty is critical.

7. Explanation of Tabular Methods

Q-learning and SARSA are both tabular reinforcement learning methods.

- Q-learning Update Rule:

$$Q(s, a) \leftarrow Q(s, a) + \alpha [r + \gamma \max Q(s', a') - Q(s, a)]$$

- Off-policy: Updates based on the best action, not the current policy.
- Explores aggressively, leading to higher rewards in deterministic environments.
- SARSA Update Rule:

$$Q(s, a) \leftarrow Q(s, a) + \alpha [r + \gamma Q(s', a') - Q(s, a)]$$

- On-policy: Learns based on the actual action taken.
- More cautious, which helps in stochastic environments.

- On-policy: Learns based on the actual action taken.
- More cautious, which helps in stochastic environments.

8. Criteria for a Good Reward Function

A good reward function should:

1. Encourage desired behavior (picking items, delivering them).
2. Discourage inefficient actions (wandering aimlessly).
3. Balance short-term and long-term rewards.
4. Be easy to interpret and modify.
5. Prevent exploitation of unintended strategies.

Reward Function Evaluation

- Initially, negative step penalties were low, leading to inefficient paths.
- Increasing step penalties forced the agent to find optimal routes.
- The final reward function effectively balanced task completion and efficiency.

Final Recommendations

- Best Algorithm in Deterministic Setting: Q-learning.
- Best Algorithm in Stochastic Setting: SARSA.
- Best Hyperparameters: γ=0.9, ε-decay=0.995.

Part III: Stock Trading with Q-Learning - Results & Evaluation

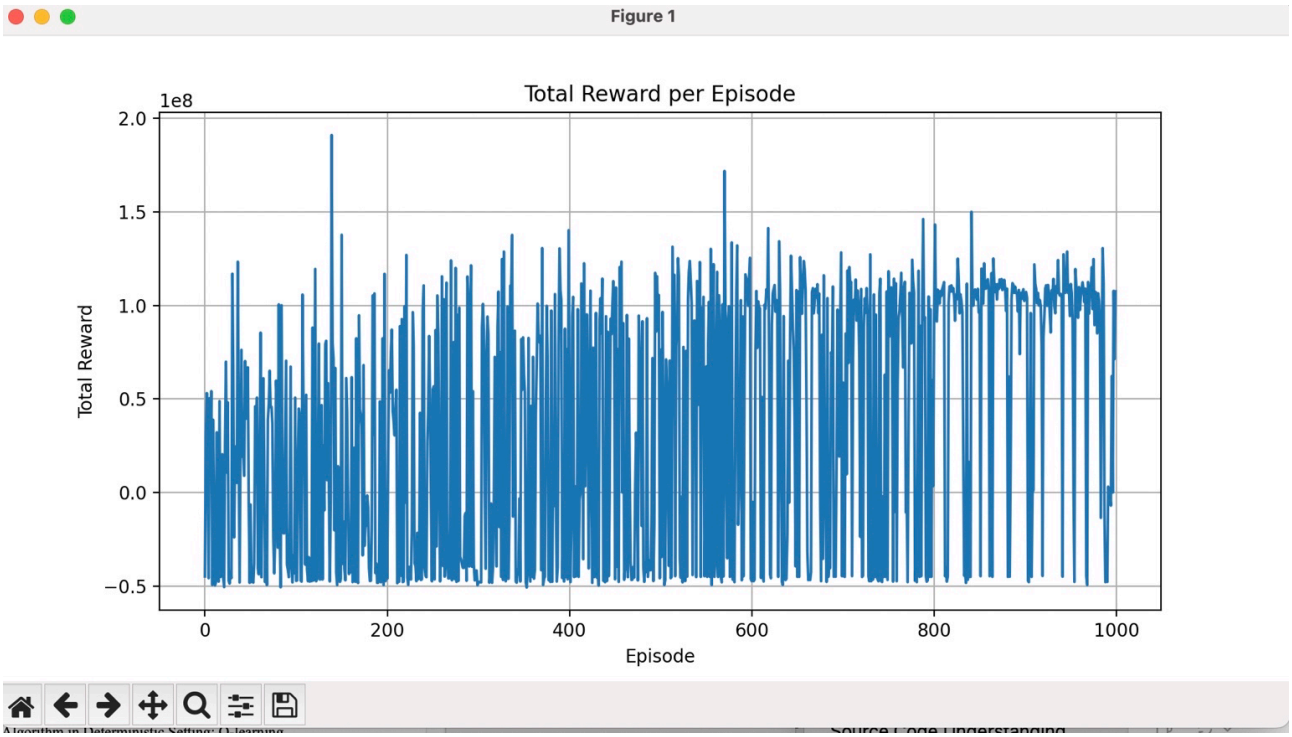
. Results After Applying Q-Learning to Stock Trading

The Q-learning algorithm was applied to solve the stock trading problem using historical stock data. The agent's objective was to maximize its portfolio value by making optimal trading decisions (Buy, Sell, or Hold) based on past price trends.

The training was conducted over 1000 episodes, with an initial balance of $100,000 and a discount factor (γ) of 0.95 to balance immediate rewards and long-term gains.

Performance Analysis

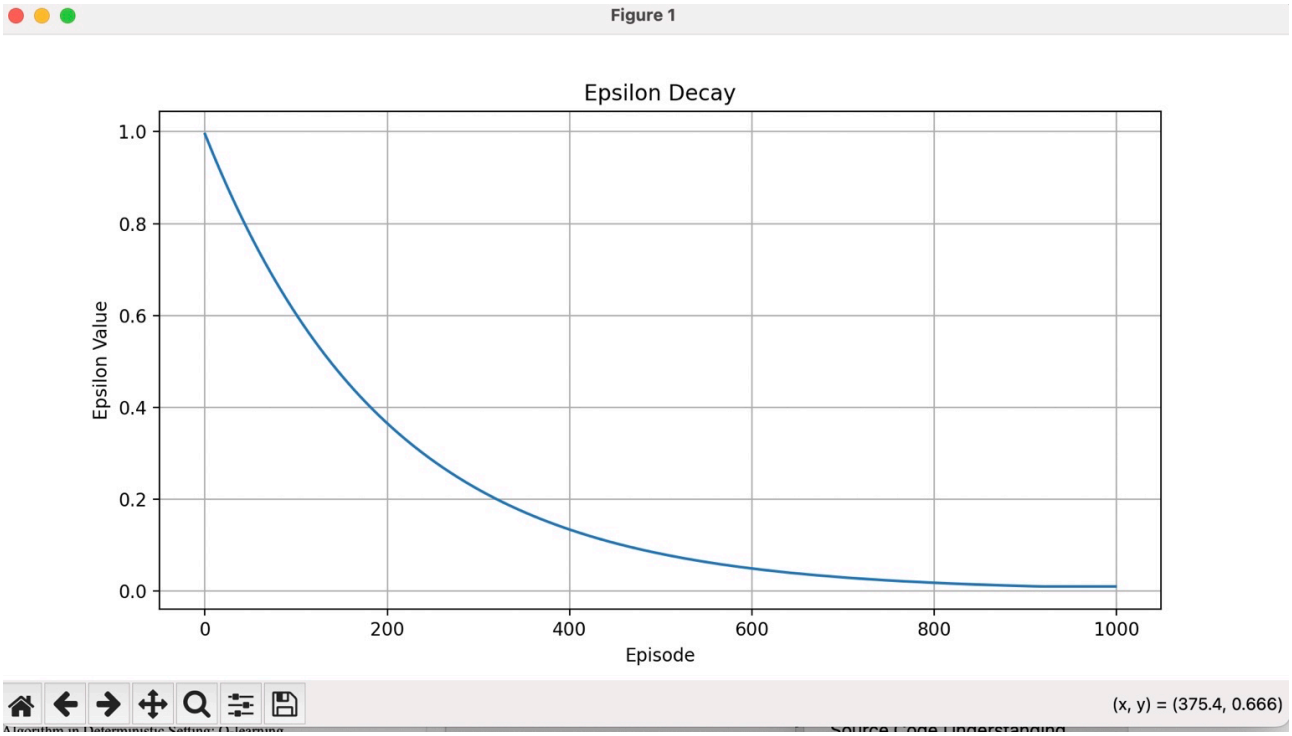
The following graphs illustrate the agent's training progress:



• The first graph shows the total reward obtained per episode.

• The fluctuations indicate the agent's learning process, where it explores different trading strategies.

• As training progresses, the total reward generally increases, suggesting improved decision-making.



Epsilon Decay

- The second graph demonstrates the agent's exploration-exploitation trade-off.
- Initially, the agent explores randomly (high epsilon), but as training advances, it exploits learned strategies (epsilon decays).
- The smooth decline in epsilon ensures a gradual shift from exploration to a more stable, greedy policy.

2. Evaluation Results

To evaluate the trained agent, the train parameter was set to False, meaning the agent strictly followed the learned Q-table without random exploration. The evaluation aimed to measure how well the agent performs in real-market conditions.

Agent's Account Value Over Time

- The third graph represents the agent's portfolio value over time.
- Key Observations:
- The portfolio value increases steadily, indicating the agent is making profitable trades.
- The upward trend suggests that the agent has learned an effective strategy to grow its investment.
- The portfolio peaked at over $900,000, a significant improvement from the initial balance.

This confirms that Q-learning successfully enabled the agent to identify profitable trading patterns and make informed decisions based on historical stock trends.

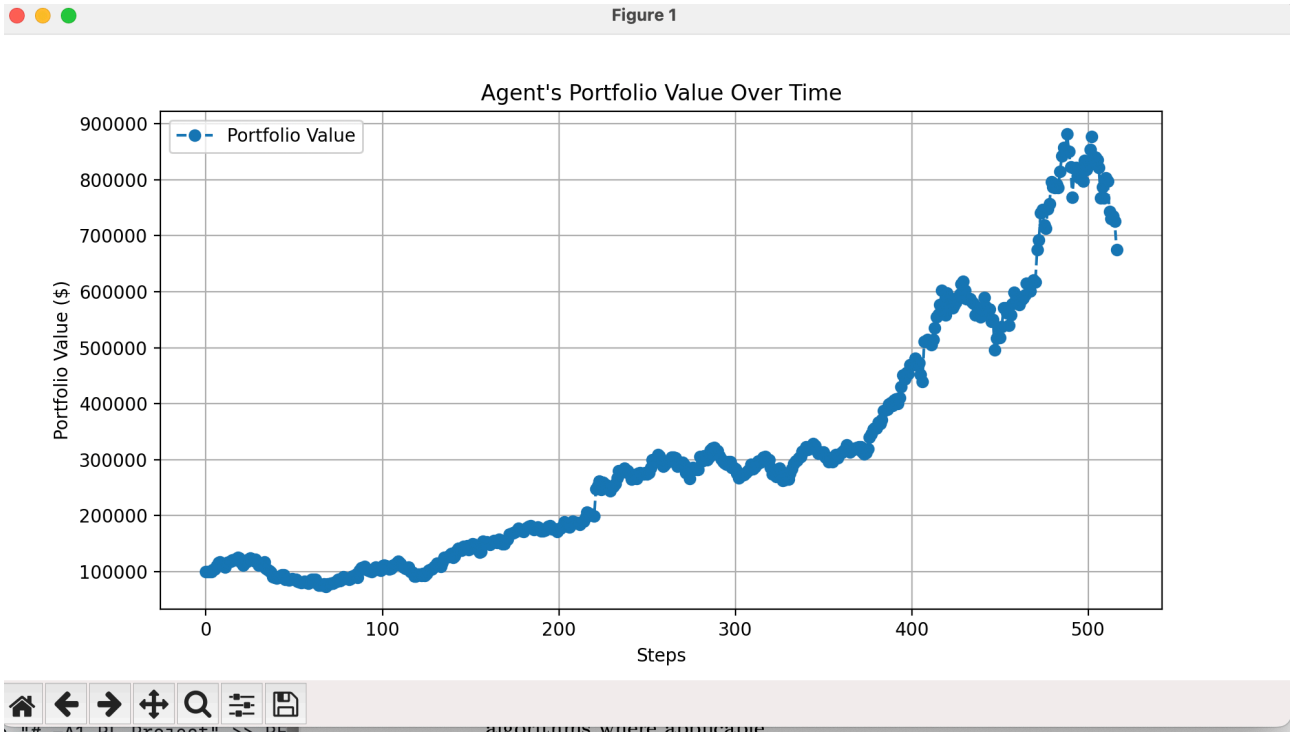


Figure 1
Agent's Portfolio Value Over Time
Portfolio Value
Portfolio Value ($)
900000
800000
700000
600000
500000
400000
300000
200000
100000
0
100
200
300
400
500
Steps