Report on Reinforcement Learning (RL) Algorithm Used

Introduction:

This report outlines the implementation of a Reinforcement Learning (RL) algorithm to optimize the investment strategy based on price drops in the NiftyBEES stock. The objective is to decide when to invest, considering historical price movements, with the aim of maximizing returns.

RL Algorithm Used:

The algorithm implemented here is a **Q-learning** based approach. Q-learning is a model-free, off-policy reinforcement learning algorithm that aims to learn the value of action-state pairs through an iterative process.

Why Q-learning?

- **Model-Free**: Q-learning does not require a model of the environment, which makes it suitable for this problem where the underlying stock market is unpredictable.
- **Off-Policy**: It learns the optimal policy independently of the actions taken, meaning it can adapt to any actions and adjust based on rewards.
- **Simplicity**: Q-learning is relatively simple to implement and understand compared to other RL methods, such as policy gradient methods or deep Q-networks.
- **Exploration vs Exploitation**: Q-learning effectively balances exploration of new investment opportunities and exploitation of the known best investment strategy.

How Q-learning Works in This Context:

- **States**: The state in this problem is represented by the price of NiftyBEES and the percentage drop from the highest price encountered.
- **Actions**: The possible actions are:
    - Invest ₹1,00,000 at a 10% drop.
    - Invest ₹2,00,000 at a 25% drop.
    - Do nothing (hold).
- **Rewards**: The reward is based on the return generated from the investment. If the investment leads to a higher price in the future, the agent gets a positive reward, i.e., the returns from the investment.

- **Q-table**: The Q-table stores the expected return for each state-action pair. The algorithm updates this table iteratively based on the rewards received after taking actions.
- **Learning Process**: Q-learning updates the Q-value for each state-action pair according to the formula:

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

where:

  - Q(s,a) is the current Q-value for the state-action pair.
  - α alpha is the learning rate.
  - r is the immediate reward from taking action a
  - γ is the discount factor.
  - $\max_a Q(s',a')$ is the maximum Q-value for the next state s′.

How it Works:

1. **Initial Setup**: The agent starts with no prior knowledge of the market. The Q-table is initialized with zeros.
2. **Exploration and Learning**: The agent begins by exploring the environment (stock prices). For each price drop, it evaluates whether to invest or hold based on the Q-values stored for each action. Over time, the agent learns the best action to take at different price levels.
3. **Action Selection**: The agent selects actions based on the Q-values and a balance between exploration (trying new actions) and exploitation (choosing the best-known action).
4. **Reward Update**: After an action is taken (i.e., an investment is made), the reward is calculated based on the returns from that action, and the Q-table is updated to reflect this.
5. **Convergence**: Over time, the agent's Q-table converges, and it starts making more informed investment decisions.

Conclusion:

Q-learning offers an effective way to optimize an investment strategy based on historical data, with the agent progressively learning the best times to invest based on observed price movements. Although the environment here (stock market) is inherently uncertain, Q-learning helps in navigating this uncertainty by learning from past experiences, ultimately maximizing long-term returns.