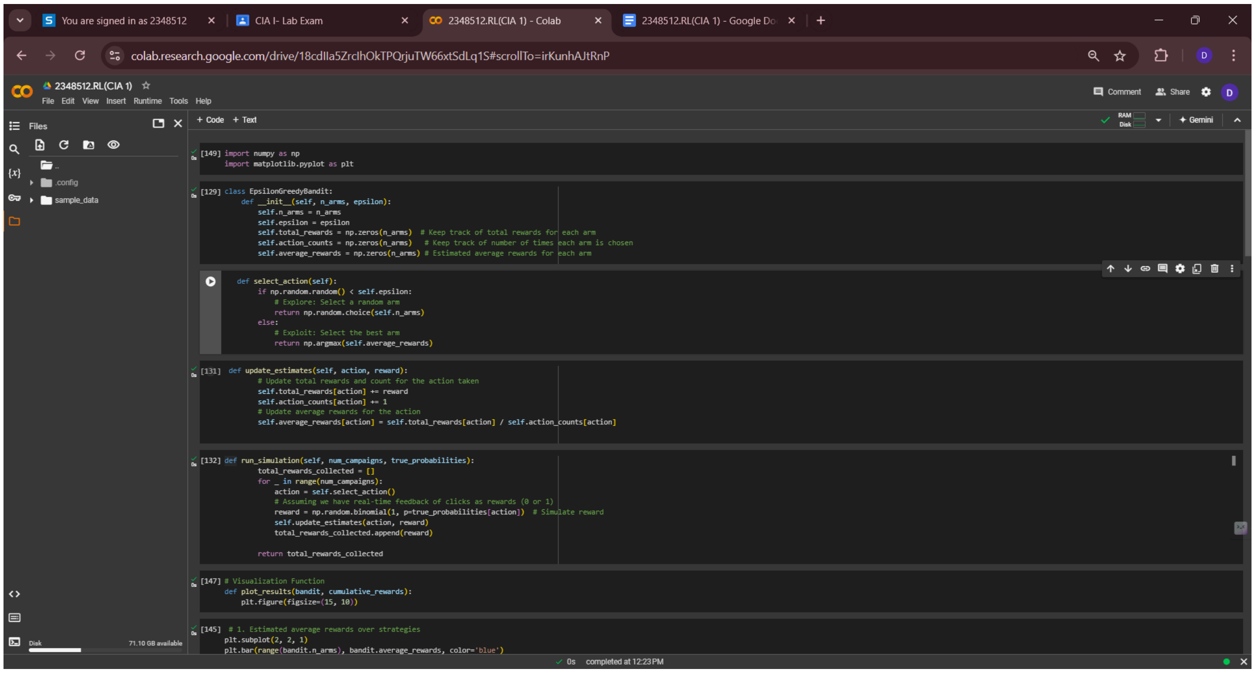
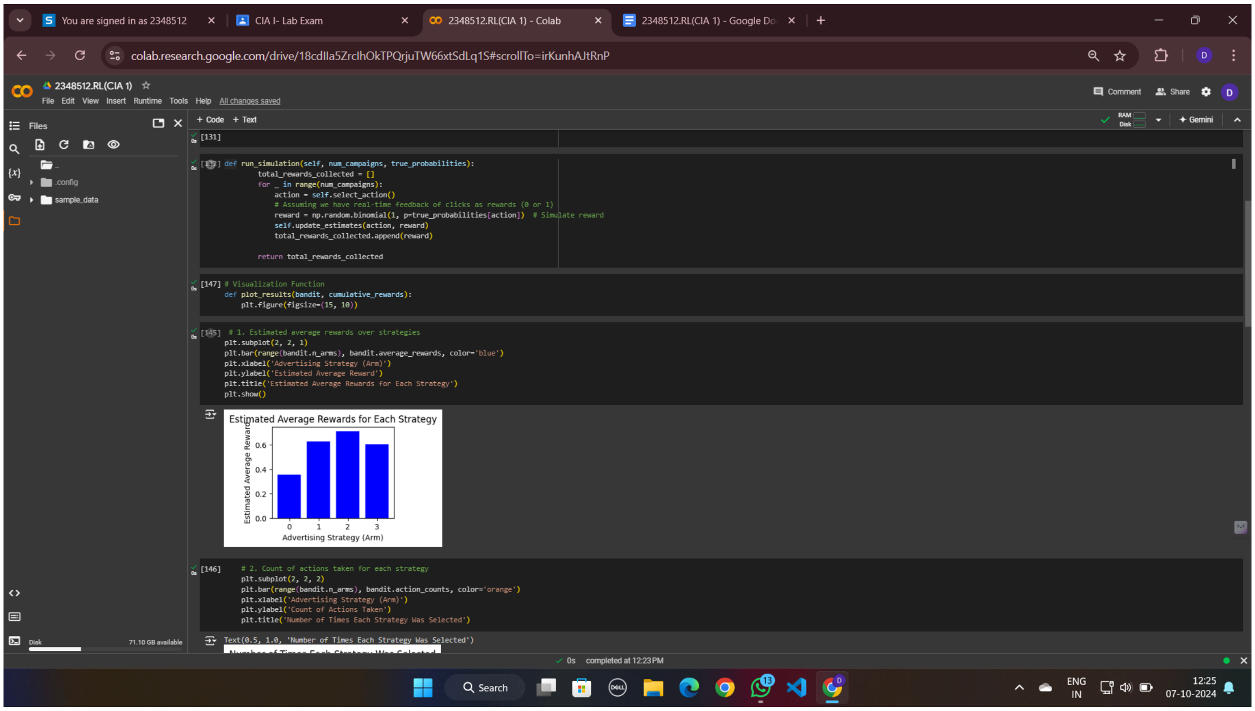
Question1:

Python implementation of the algorithm:

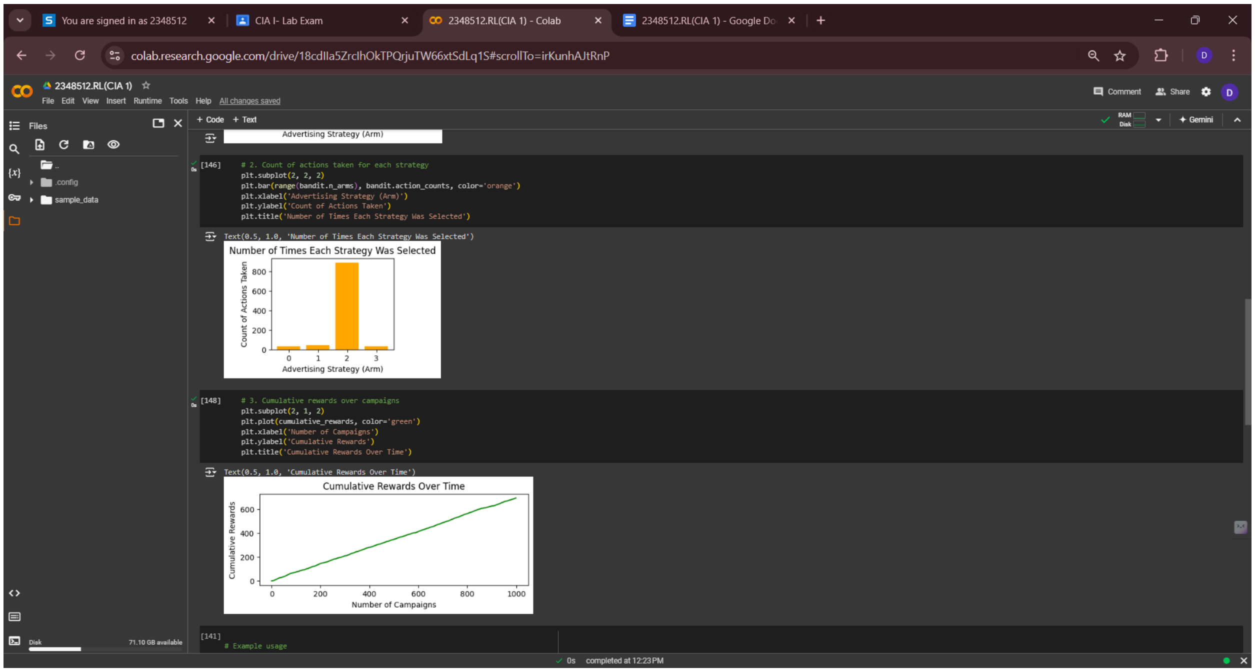

You are signed in as 2348512
CIA I- Lab Exam
2348512.RL(CIA 1) - Colab
2348512.RL(CIA 1) - Google Do
colab.research.google.com/drive/18cdIla5ZrclhOkTPQrjuTW66xtSdLq1S#scrollTo=irKunhAJtRnP
2348512.RL(CIA 1)
File Edit View Insert Runtime Tools Help
Comment Share
Files
+ Code + Text
RAM Disk
Gemini
.config
sample_data
Disk 71.10 GB available
0s completed at 12:23 PM

```
[149] import numpy as np
      import matplotlib.pyplot as plt

[129] class EpsilonGreedyBandit:
          def __init__(self, n_arms, epsilon):
              self.n_arms = n_arms
              self.epsilon = epsilon
              self.total_rewards = np.zeros(n_arms)  # Keep track of total rewards for each arm
              self.action_counts = np.zeros(n_arms)   # Keep track of number of times each arm is chosen
              self.average_rewards = np.zeros(n_arms) # Estimated average rewards for each arm

        def select_action(self):
            if np.random.random() < self.epsilon:
                # Explore: Select a random arm
                return np.random.choice(self.n_arms)
            else:
                # Exploit: Select the best arm
                return np.argmax(self.average_rewards)

[131]  def update_estimates(self, action, reward):
              # Update total rewards and count for the action taken
              self.total_rewards[action] += reward
              self.action_counts[action] += 1
              # Update average rewards for the action
              self.average_rewards[action] = self.total_rewards[action] / self.action_counts[action]

[132] def run_simulation(self, num_campaigns, true_probabilities):
              total_rewards_collected = []
              for _ in range(num_campaigns):
                  action = self.select_action()
                  # Assuming we have real-time feedback of clicks as rewards (0 or 1)
                  reward = np.random.binomial(1, p=true_probabilities[action])  # Simulate reward
                  self.update_estimates(action, reward)
                  total_rewards_collected.append(reward)

              return total_rewards_collected

[147] # Visualization Function
      def plot_results(bandit, cumulative_rewards):
          plt.figure(figsize=(15, 10))

[145]  # 1. Estimated average rewards over strategies
      plt.subplot(2, 2, 1)
      plt.bar(range(bandit.n_arms), bandit.average_rewards, color='blue')
```

Average Rewards over strategies:


You are signed in as 2348512
CIA I- Lab Exam
2348512.RL(CIA 1) - Colab
2348512.RL(CIA 1) - Google Do
colab.research.google.com/drive/18cdIla5ZrclhOkTPQrjuTW66xtSdLq1S#scrollTo=irKunhAJtRnP
2348512.RL(CIA 1)
File Edit View Insert Runtime Tools Help All changes saved
Comment Share
Files
+ Code + Text
Gemini
.config
sample_data
Disk 71.10 GB available
Estimated Average Rewards for Each Strategy
Estimated Average Reward
0.0
0.2
0.4
0.6
0
1
2
3
Advertising Strategy (Arm)
0s completed at 12:23 PM
Search
ENG IN
12:25
07-10-2024

```
[131]

[132] def run_simulation(self, num_campaigns, true_probabilities):
              total_rewards_collected = []
              for _ in range(num_campaigns):
                  action = self.select_action()
                  # Assuming we have real-time feedback of clicks as rewards (0 or 1)
                  reward = np.random.binomial(1, p=true_probabilities[action])  # Simulate reward
                  self.update_estimates(action, reward)
                  total_rewards_collected.append(reward)

              return total_rewards_collected

[147] # Visualization Function
      def plot_results(bandit, cumulative_rewards):
          plt.figure(figsize=(15, 10))

[145]  # 1. Estimated average rewards over strategies
      plt.subplot(2, 2, 1)
      plt.bar(range(bandit.n_arms), bandit.average_rewards, color='blue')
      plt.xlabel('Advertising Strategy (Arm)')
      plt.ylabel('Estimated Average Reward')
      plt.title('Estimated Average Rewards for Each Strategy')
      plt.show()

[146]     # 2. Count of actions taken for each strategy
          plt.subplot(2, 2, 2)
          plt.bar(range(bandit.n_arms), bandit.action_counts, color='orange')
          plt.xlabel('Advertising Strategy (Arm)')
          plt.ylabel('Count of Actions Taken')
          plt.title('Number of Times Each Strategy Was Selected')

Text(0.5, 1.0, 'Number of Times Each Strategy Was Selected')
```


You are signed in as 2348512
CIA I- Lab Exam
2348512.RL(CIA 1) - Colab
2348512.RL(CIA 1) - Google Do
colab.research.google.com/drive/18cdlla5ZrclhOkTPQrjuTW66xtSdLq1S#scrollTo=irKunhAJtRnP
2348512.RL(CIA 1)
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
Files
.config
sample_data
+ Code
+ Text
RAM
Disk
Gemini
Advertising Strategy (Arm)
Number of Times Each Strategy Was Selected
Count of Actions Taken
800
600
400
200
0
0
1
2
3
Advertising Strategy (Arm)
Cumulative Rewards Over Time
Cumulative Rewards
600
400
200
0
0
200
400
600
800
1000
Number of Campaigns
Disk
71.10 GB available

```
[146]   # 2. Count of actions taken for each strategy
        plt.subplot(2, 2, 2)
        plt.bar(range(bandit.n_arms), bandit.action_counts, color='orange')
        plt.xlabel('Advertising Strategy (Arm)')
        plt.ylabel('Count of Actions Taken')
        plt.title('Number of Times Each Strategy Was Selected')
```

```
Text(0.5, 1.0, 'Number of Times Each Strategy Was Selected')
```

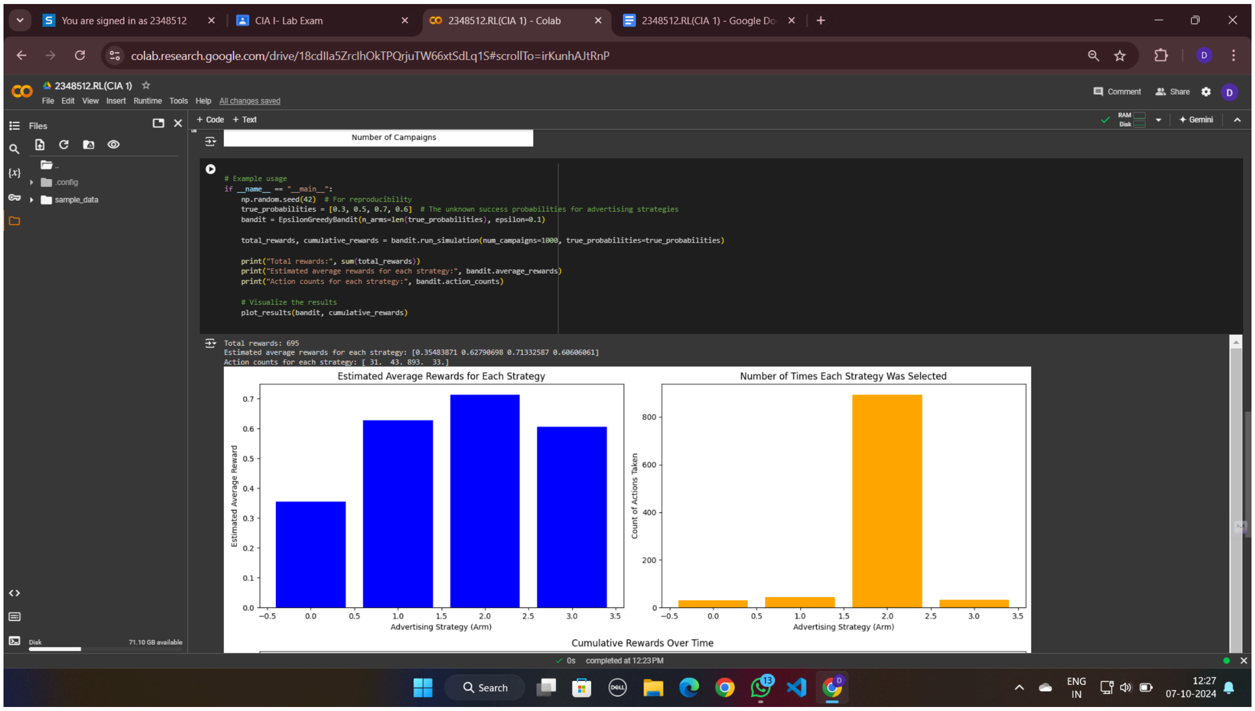
```
[148]   # 3. Cumulative rewards over campaigns
        plt.subplot(2, 1, 2)
        plt.plot(cumulative_rewards, color='green')
        plt.xlabel('Number of Campaigns')
        plt.ylabel('Cumulative Rewards')
        plt.title('Cumulative Rewards Over Time')
```

```
Text(0.5, 1.0, 'Cumulative Rewards Over Time')
```

```
[141]
    # Example usage
```

✓ 0s completed at 12:23 PM

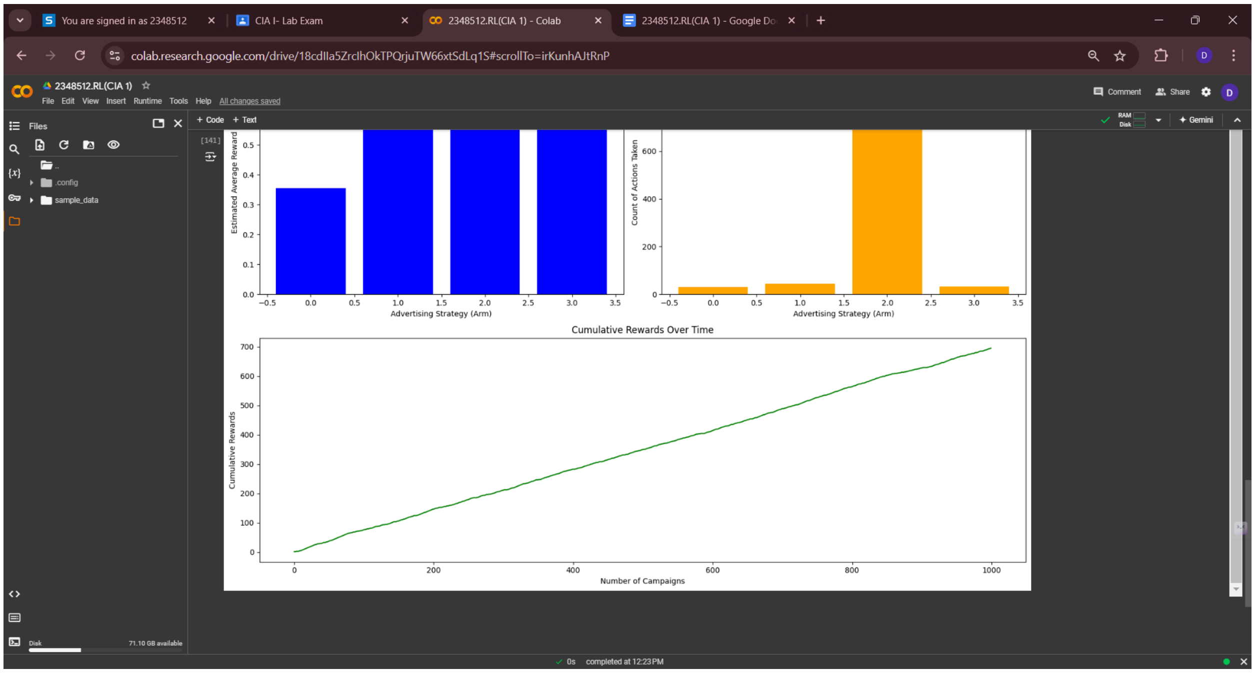
Show how the algorithm helps to choose the best-performing advertising strategy over time:



```
# Example usage
if __name__ == "__main__":
    np.random.seed(42)  # For reproducibility
    true_probabilities = [0.3, 0.5, 0.7, 0.6]  # The unknown success probabilities for advertising strategies
    bandit = EpsilonGreedyBandit(n_arms=len(true_probabilities), epsilon=0.1)

    total_rewards, cumulative_rewards = bandit.run_simulation(num_campaigns=1000, true_probabilities=true_probabilities)

    print("Total rewards:", sum(total_rewards))
    print("Estimated average rewards for each strategy:", bandit.average_rewards)
    print("Action counts for each strategy:", bandit.action_counts)

    # Visualize the results
    plot_results(bandit, cumulative_rewards)
```

```
Total rewards: 695
Estimated average rewards for each strategy: [0.35483871 0.62790698 0.71332587 0.60606061]
Action counts for each strategy: [ 31.  43. 893.  33.]
```

You are signed in as 2348512
CIA I- Lab Exam
2348512.RL(CIA 1) - Colab
2348512.RL(CIA 1) - Google Do
colab.research.google.com/drive/18cdlla5ZrclhOkTPQrjuTW66xtSdLq1S#scrollTo=irKunhAJtRnP
2348512.RL(CIA 1)
File Edit View Insert Runtime Tools Help All changes saved
Comment Share Gemini
Files
+ Code + Text
Number of Campaigns
Estimated Average Rewards for Each Strategy
Estimated Average Reward
Advertising Strategy (Arm)
Number of Times Each Strategy Was Selected
Count of Actions Taken
Cumulative Rewards Over Time
Disk 71.10 GB available
0s completed at 12:23 PM
Search
ENG IN
12:27
07-10-2024



You are signed in as 2348512
CIA I- Lab Exam
2348512.RL(CIA 1) - Colab
2348512.RL(CIA 1) - Google Do
colab.research.google.com/drive/18cdlla5ZrclhOkTPQrjuTW66xtSdLq1S#scrollTo=irKunhAJtRnP
2348512.RL(CIA 1)
File Edit View Insert Runtime Tools Help All changes saved
[141]
Estimated Average Reward
0.0
0.1
0.2
0.3
0.4
0.5
−0.5 0.0 0.5 1.0 1.5 2.0 2.5 3.0 3.5
Advertising Strategy (Arm)
Count of Actions Taken
0
200
400
600
Cumulative Rewards Over Time
Cumulative Rewards
0
100
200
300
400
500
600
700
0 200 400 600 800 1000
Number of Campaigns
Disk 71.10 GB available
0s completed at 12:23 PM