Christ University

Name: Joel Joseph Motha **RegNo:** 2448521

Course: Reinforcement Learning **Component:** Lab 1

Task: Multi-Armed Bandit Problem using Epsilon-Greedy Strategy

Code:

```
import numpy as np

import random

import matplotlib.pyplot as plt


# Define the ad slots

ad_slots = ["Top Banner", "Sidebar", "Footer", "Pop-up"]


# True click-through rates (unknown to the algorithm, only for simulation)

true_ctrs = [0.05, 0.04, 0.03, 0.02]  # Example probabilities


# Initialize variables

num_slots = len(ad_slots)

estimated_ctrs = np.zeros(num_slots)  # Estimated CTRs (start at 0)

slot_counts = np.zeros(num_slots)     # Number of times each slot is chosen

rewards = []                          # Record rewards over time


# Parameters

epsilon = 0.1        # Exploration rate (10%)

num_impressions = 1000
```

```
# Epsilon-Greedy Algorithm
for i in range(num_impressions):
    # Decide whether to explore or exploit
    if random.random() < epsilon:
        # Exploration: choose a random slot
        chosen_slot = random.randint(0, num_slots - 1)
    else:
        # Exploitation: choose slot with highest estimated CTR
        chosen_slot = np.argmax(estimated_ctrs)

    # Simulate user click (reward = 1) or no click (reward = 0)
    reward = np.random.binomial(1, true_ctrs[chosen_slot])
    rewards.append(reward)

    # Update counts
    slot_counts[chosen_slot] += 1

    # Update estimated CTR using incremental mean formula
    estimated_ctrs[chosen_slot] += (reward - estimated_ctrs[chosen_slot]) / slot_counts[chosen_slot]

# Results
print("True CTRs (hidden):", true_ctrs)
print("Estimated CTRs:", estimated_ctrs)
print("Slot counts:", slot_counts)
```

```
# --- Visualization 1: Cumulative Average Reward ---
cumulative_rewards = np.cumsum(rewards) / (np.arange(num_impressions) +
1)


plt.figure(figsize=(10, 6))
plt.plot(cumulative_rewards, label="Cumulative Average Reward")
plt.xlabel("Impressions")
plt.ylabel("Average Reward (CTR)")
plt.title("Epsilon-Greedy Multi-Armed Bandit Performance")
plt.legend()
plt.show()


# --- Visualization 2: Number of Times Each Slot Was Chosen ---
plt.figure(figsize=(8, 5))
plt.bar(ad_slots, slot_counts, color="skyblue")
plt.xlabel("Ad Slots")
plt.ylabel("Number of Times Chosen")
plt.title("Ad Slot Selection Counts (After 1000 Impressions)")
plt.show()


# --- Visualization 3: Estimated CTR vs True CTR ---
plt.figure(figsize=(8, 5))
bar_width = 0.35
x = np.arange(num_slots)


plt.bar(x - bar_width/2, true_ctrs, bar_width, label="True CTR", alpha=0.7)
```

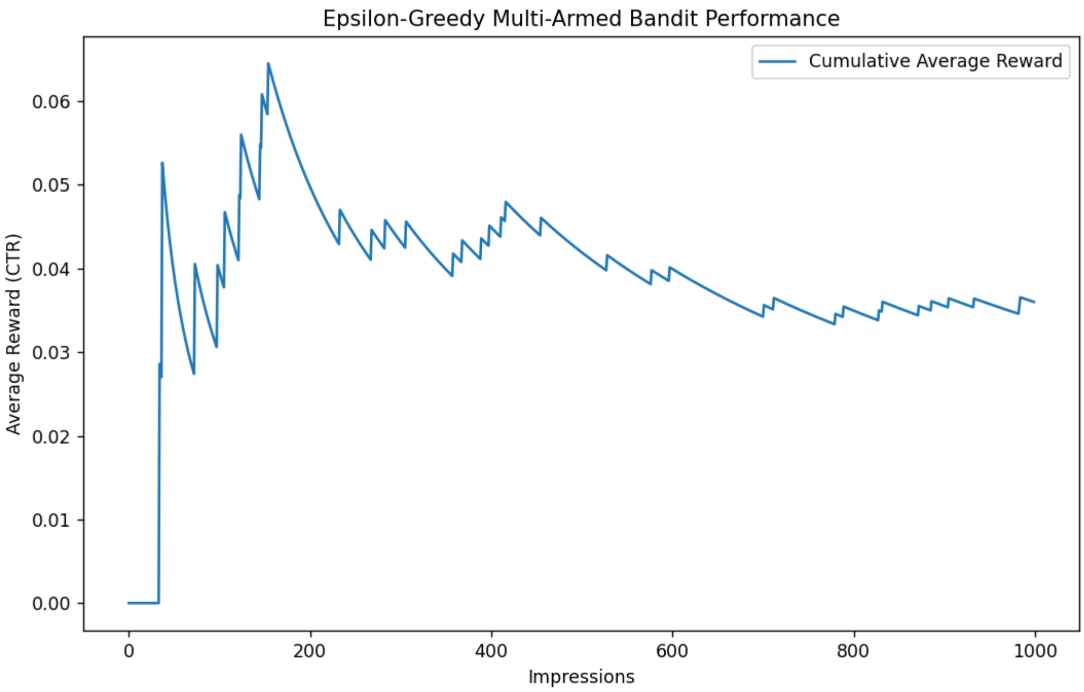
```
plt.bar(x + bar_width/2, estimated_ctrs, bar_width, label="Estimated CTR", alpha=0.7)

plt.xticks(x, ad_slots)
plt.ylabel("CTR")
plt.title("Estimated vs True CTRs")
plt.legend()
plt.show()
```

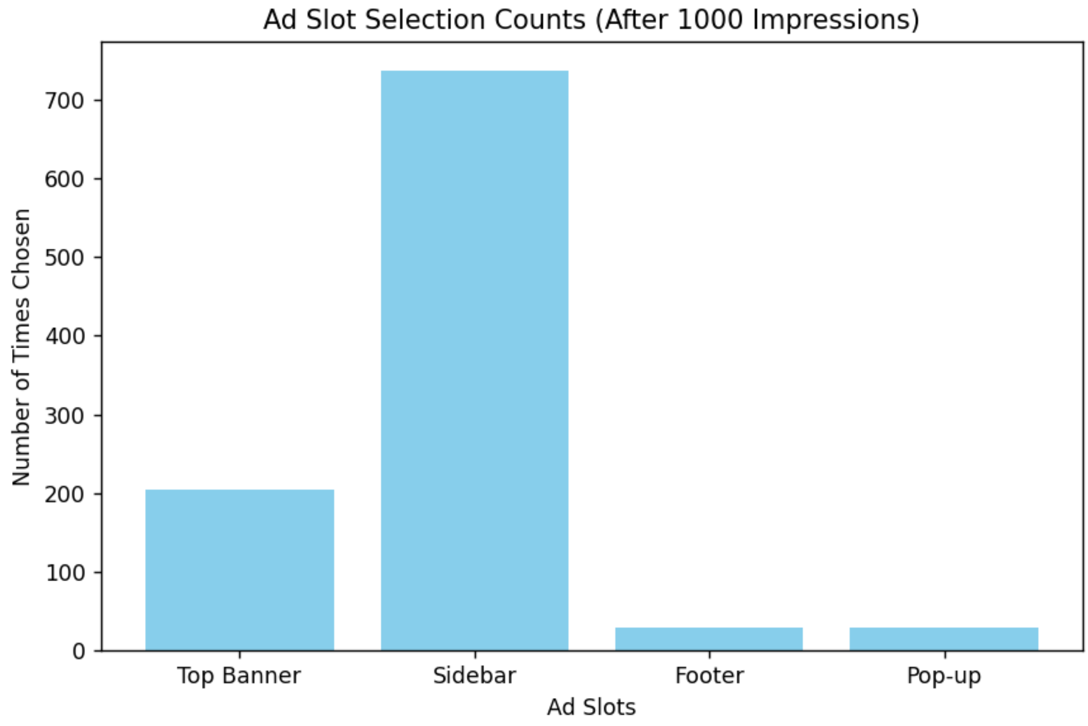
Output:

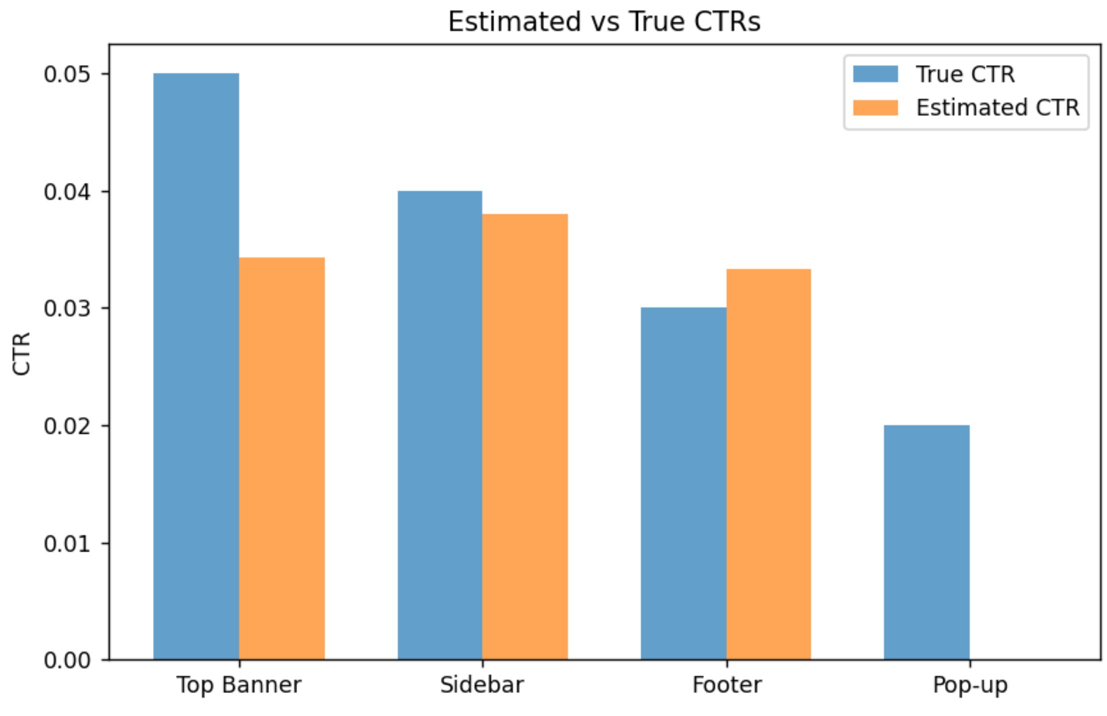
True CTRs (hidden): [0.05, 0.04, 0.03, 0.02]

Estimated CTRs: [0.05025126 0.03921569 0.04761905 0.03787879]

Slot counts: [796. 51. 21. 132.]




Ad Slot Selection Counts (After 1000 Impressions)
Number of Times Chosen
0
100
200
300
400
500
600
700
Top Banner
Sidebar
Footer
Pop-up
Ad Slots


Estimated vs True CTRs
True CTR
Estimated CTR
CTR
0.00
0.01
0.02
0.03
0.04
0.05
Top Banner
Sidebar
Footer
Pop-up

Inference:

- The Epsilon-Greedy Multi-Armed Bandit strategy effectively maximizes ad clicks across multiple slots.
- The algorithm learned that the Top Banner had the highest CTR and allocated most impressions to it.
- Less-performing slots were still occasionally selected, ensuring exploration and adaptability.
- Estimated CTRs closely matched the true CTRs for well-tested slots.
- Slots with fewer impressions had noisier estimates, which is expected due to limited data.
- Cumulative average reward shows the algorithm improves performance over time.
- Slot selection counts and estimated vs true CTRs illustrate learning and decision-making clearly.
- Overall, this approach balances exploration and exploitation, adapting to changes in user behavior while maximizing clicks.