**Lab Exercise :1**

Implement Multi-Armed Bandit Problem from Scratch in Python

**Manager:**

"We have four different ad slots on our website: the top banner, sidebar, footer, and pop-up. Each of these slots has different characteristics in terms of visibility and user engagement. Right now, we’re unsure which slot will give us the highest click-through rate (CTR) for our ads, and we don't want to rely on gut feeling alone. What approach can we take to maximize the number of clicks we get from these ads?"

**Data Scientist:**

"We can approach this using a **Multi-Armed Bandit** strategy. The idea is similar to choosing the best slot machine to play in a casino where each ad slot is an arm of the bandit. Over time, we’ll balance between **exploring** ad slots that we haven’t tried much and **exploiting** the ad slots that have shown the best performance so far."

**Manager:**

"That makes sense. But I’m still unclear about how we can explore all ad slots without losing potential clicks by sticking to the wrong slots for too long. Can you explain that part a bit more?"

**Data Scientist:**

"Absolutely. We can use an **Epsilon-Greedy** algorithm for this. Here’s how it works:

* We’ll start by randomly assigning impressions to the different ad slots to get some initial feedback on their performance.
* At each step, there’s a small probability, say 10%, where we will explore a random slot. For the remaining 90%, we’ll exploit the ad slot that currently has the highest estimated CTR based on the feedback we’ve already collected. This approach ensures that we won’t get stuck on one slot too early, and we’ll still test other slots periodically to see if they perform better."

**Manager:**

"I see. So the idea is to keep testing all the ad slots to some degree, but focus more on the ones that show better performance over time. How do we actually **update** our understanding of how each ad slot performs? For example, after an ad is displayed on the sidebar, how do we use that information to learn whether the sidebar is good or not?"

**Data Scientist:**

"Good question. For every ad impression shown, we receive feedback in the form of whether the user clicked on the ad or not. This feedback is binary: **click** (reward = 1) or **no click** (reward = 0).

* Each time we get feedback, we update the estimated CTR for that ad slot. We do this incrementally using the average reward (clicks) received so far for that slot.
* Over time, the ad slots with better CTRs will naturally accumulate more clicks, and the algorithm will tend to choose them more often."

**Manager:**

"What happens if one of the ad slots changes performance over time? For example, what if our pop-up ads become less effective as users get annoyed with them?"

**Data Scientist:**

"That’s an important consideration. Since we are using **Epsilon-Greedy**, we are constantly doing a little bit of exploration. So, even if one ad slot starts underperforming, the algorithm will keep checking other slots occasionally. If it finds that another ad slot is performing better, it will naturally shift more impressions toward the better-performing one. In this way, the strategy adapts to changing user behavior over time."

**Manager:**

"I understand now. But how long would it take for the model to learn which ad slot is the best one?"

**Data Scientist:**

"That depends on the number of impressions we get and the differences in performance between the ad slots. The more impressions we gather, the faster we’ll learn. If one ad slot is significantly better than the others, we will discover that relatively quickly. However, if the performance differences between the slots are small, it may take longer for the algorithm to decide."

**Manager:**

"Okay, that sounds good. Can you summarize how we would implement this strategy? What steps do we need to take?"

**Data Scientist:**

"Sure! Here’s a step-by-step breakdown:

1. **Define the Problem**: We have four ad slots (top banner, sidebar, footer, and pop-up) and we want to maximize the number of clicks.
2. **Set Up Epsilon-Greedy Algorithm**: We’ll initialize with an equal probability of showing each ad slot, but over time, we’ll use a 90-10 split between exploiting the best-performing slot and exploring other slots.
3. **Run the Simulation**: For each impression, we:
   * Select an ad slot using the epsilon-greedy algorithm.
   * Show the ad in that slot and observe whether the user clicks on it.
   * Update the estimated CTR for that slot based on the observed result.
4. **Adjust Over Time**: As the algorithm collects more data, it will become better at selecting the most effective ad slot, but it will continue to explore occasionally to adapt to any changes."

**Manager:**

"Great! I like the idea of this adaptive approach. Can you implement this in Python and show me how it performs after, say, 1000 ad impressions?"

Based on the above context write a program in python using Armed Bandit Problem strategy.