Marketing Analytics | Homework 2 | David Aslanyan

Step 1 - Load the libraries

In [12]:
```
from abc import ABC, abstractmethod
import numpy as np
from loguru import logger
import pandas as pd
import matplotlib.pyplot as plt

class Bandit(ABC):
    """Abstract class for the Bandit algorithms (EpsilonGreedy, ThompsonSamp

    @abstractmethod
    def __init__(self, true_reward_probabilities):
        pass

    @abstractmethod
    def __repr__(self):
        pass

    @abstractmethod
    def pull(self):
        pass

    @abstractmethod
    def update(self):
        pass

    @abstractmethod
    def experiment(self):
        pass

    @abstractmethod
    def report(self):
        pass
```

Step 2 - Define the visualizations

In [13]:
```
class Visualization:

    def plot_rewards(self, epsilon_rewards, ts_rewards):
        plt.figure(figsize=(10, 6))
        plt.plot(epsilon_rewards, label="EpsilonGreedy Cumulative Rewards")
        plt.plot(ts_rewards, label="ThompsonSampling Cumulative Rewards")
        plt.xlabel('Trials')
        plt.ylabel('Cumulative Reward')
        plt.title('Comparison of Cumulative Rewards')
        plt.legend()
        plt.show()

    def plot_rewards_and_regrets(self, epsilon_rewards, ts_rewards, epsilon_
        plt.figure(figsize=(10, 6))
        plt.plot(epsilon_rewards, label="EpsilonGreedy Cumulative Rewards")
        plt.plot(ts_rewards, label="ThompsonSampling Cumulative Rewards")
```

```
        plt.plot(epsilon_regrets, label="EpsilonGreedy Cumulative Regrets",
        plt.plot(ts_regrets, label="ThompsonSampling Cumulative Regrets", li
        plt.xlabel('Trials')
        plt.ylabel('Cumulative Value')
        plt.title('Comparison of Cumulative Rewards and Regrets')
        plt.legend()
        plt.show()
```

Step 3 - Creating Epsilon-Greedy class

In [14]:
```
class EpsilonGreedy(Bandit):
    def __init__(self, true_reward_probabilities, epsilon=1.0, decay_rate=0.
        self.true_reward_probabilities = true_reward_probabilities
        self.epsilon = epsilon
        self.decay_rate = decay_rate
        self.num_arms = num_arms
        self.arm_pull_counts = np.zeros(num_arms)
        self.arm_value_estimates = np.zeros(num_arms)

    def __repr__(self):
        return f"EpsilonGreedy(epsilon={self.epsilon}, estimated_values={se

    def pull(self):
        if np.random.rand() < self.epsilon:
            arm_chosen = np.random.randint(self.num_arms)
        else:
            arm_chosen = np.argmax(self.arm_value_estimates)
        reward = np.random.binomial(1, self.true_reward_probabilities[arm_ch
        return arm_chosen, reward

    def update(self, arm_chosen, reward):
        self.arm_pull_counts[arm_chosen] += 1
        self.arm_value_estimates[arm_chosen] += (reward - self.arm_value_est

    def experiment(self, num_trials):
        total_reward = 0
        cumulative_rewards = []
        cumulative_regrets = []
        for trial in range(num_trials):
            arm_chosen, reward = self.pull()
            self.update(arm_chosen, reward)
            total_reward += reward
            cumulative_rewards.append(total_reward)
            current_regret = np.max(self.true_reward_probabilities) - self.a
            cumulative_regrets.append(np.sum(current_regret))
            self.epsilon *= self.decay_rate
        return cumulative_rewards, cumulative_regrets

    def report(self):
        avg_reward = np.mean(self.arm_value_estimates)
        avg_regret = np.sum(np.max(self.true_reward_probabilities) - self.ar
        logger.info(f"EpsilonGreedy - Average Estimated Reward: {avg_reward:
        logger.info(f"EpsilonGreedy - Total Regret: {avg_regret:.4f}")
        return avg_reward, avg_regret
```

Step 4 - Creating Thompson-Sampling class

In [15]:
```
class ThompsonSampling(Bandit):
    def __init__(self, true_reward_probabilities, num_arms=4):
        self.true_reward_probabilities = true_reward_probabilities
```

```
        self.num_arms = num_arms
        self.alpha = np.ones(num_arms)
        self.beta = np.ones(num_arms)

    def __repr__(self):
        return f"ThompsonSampling(alpha={self.alpha}, beta={self.beta})"

    def pull(self):
        theta_samples = np.random.beta(self.alpha, self.beta)
        arm_chosen = np.argmax(theta_samples)
        reward = np.random.binomial(1, self.true_reward_probabilities[arm_ch
        return arm_chosen, reward

    def update(self, arm_chosen, reward):
        if reward == 1:
            self.alpha[arm_chosen] += 1
        else:
            self.beta[arm_chosen] += 1

    def experiment(self, num_trials):
        total_reward = 0
        cumulative_rewards = []
        cumulative_regrets = []
        for trial in range(num_trials):
            arm_chosen, reward = self.pull()
            self.update(arm_chosen, reward)
            total_reward += reward
            cumulative_rewards.append(total_reward)
            current_regret = np.max(self.true_reward_probabilities) - (self
            cumulative_regrets.append(np.sum(current_regret))
        return cumulative_rewards, cumulative_regrets

    def report(self):
        avg_reward = np.mean(self.alpha / (self.alpha + self.beta))
        avg_regret = np.sum(np.max(self.true_reward_probabilities) - (self.a
        logger.info(f"ThompsonSampling - Average Estimated Reward: {avg_rewa
        logger.info(f"ThompsonSampling - Total Regret: {avg_regret:.4f}")
        return avg_reward, avg_regret
```

Step 5 - Visualizing the results

In [16]:

```
def comparison(true_reward_probabilities, num_trials=20000):
    epsilon_bandit = EpsilonGreedy(true_reward_probabilities)
    ts_bandit = ThompsonSampling(true_reward_probabilities)

    epsilon_rewards, epsilon_regrets = epsilon_bandit.experiment(num_trials)
    ts_rewards, ts_regrets = ts_bandit.experiment(num_trials)

    epsilon_bandit.report()
    ts_bandit.report()

    vis = Visualization()
    vis.plot_rewards(epsilon_rewards, ts_rewards)
    vis.plot_rewards_and_regrets(epsilon_rewards, ts_rewards, epsilon_regret

    results_df = pd.DataFrame({
        'Bandit': ['EpsilonGreedy'] * len(epsilon_rewards) + ['ThompsonSamp'
        'Reward': epsilon_rewards + ts_rewards,
        'Algorithm': ['EpsilonGreedy'] * len(epsilon_rewards) + ['ThompsonSa
    })
    results_df.to_csv('bandit_experiment_results.csv', index=False)
```

```
if __name__ == "__main__":
    true_reward_probabilities = [0.1, 0.2, 0.3, 0.4]
    comparison(true_reward_probabilities)
```

```
2025-03-28 11:49:14.601 | INFO     | __main__:report:42 - EpsilonGreedy - Average Estimated Reward: 0.2525
2025-03-28 11:49:14.601 | INFO     | __main__:report:43 - EpsilonGreedy - Total Regret: 0.5899
2025-03-28 11:49:14.602 | INFO     | __main__:report:39 - ThompsonSampling - Average Estimated Reward: 0.2713
2025-03-28 11:49:14.603 | INFO     | __main__:report:40 - ThompsonSampling - Total Regret: 0.5149
```