Reinforcement Learning - Final Project

DSM – BSE

Oliver Tausendschön

Marvin Ernst

Timothy Cassel

Contents

July 2, 2025

Contextual Bandits from Real and Synthetic Data: Online Learning and Offline Policy Evaluation

1. Introduction

In this project, we investigate the performance of contextual bandit algorithms in realistic environments using logged data. Our main objective is to understand how popular algorithms such as Upper Confidence Bound (UCB) and Thompson Sampling behave when deployed without prior training - that is, in a cold-start setting where they begin learning purely from logged historical interactions. To achieve this, we use the Open Bandit Pipeline (OBP) dataset, which provides a rich set of logged interactions from a large-scale online fashion platform. This dataset is specifically designed for off-policy evaluation (OPE) and contextual bandit experimentation. It allows us to assess how bandit algorithms perform in settings where reward feedback is only available for actions taken by a behavior policy, making the evaluation of new policies inherently challenging.

Initially, we aimed to perform true online learning with this real-world data, simulating how a bandit algorithm would behave if deployed live on the platform. However, we soon discovered the practical and methodological limitations of this approach. In particular, selective updates - only updating the algorithm when the logged action matches the one selected by the learning algorithm - can introduce severe bias. This realization led us to separate our analysis into two distinct parts: First, a logged bandit replay experiment that mimics online learning using selective updates; and second, fully offline policy evaluation using OPE techniques, where the evaluation policy remains fixed and all learning is done prior to deployment.

Throughout the project, we encountered several challenges, including high computational costs, difficulty scaling certain algorithms to the full dataset (especially with 80 arms and over 1 million logged rounds), and issues related to off-policy learning such as correctly specifying propensity scores and reward models. Nevertheless, we present a comprehensive analysis of contextual bandit algorithms across multiple experimental setups, both synthetic and real, with a focus on robustness, performance comparison, and practical implementation concerns.

The remainder of this report is structured as follows. In Section 2, we describe the OBP and our preprocessing steps. Section 3 presents our online simulation using logged bandit replay. Section 4 introduces synthetic benchmarking to evaluate algorithms under controlled

conditions. In Section 5, we perform true offline policy evaluation using standard OPE estimators. Finally, Section 6 concludes with a discussion of insights and potential future directions.

2. Open Bandits Pipeline

To follow our goal of carrying out our empirical evaluation of contextual bandit algorithms in realistic offline settings, we rely on the Open Bandit Pipeline. OBP is an open-source framework developed by CyberAgent, Inc. in collaboration with academic researchers to support reproducible research in offline reinforcement learning and contextual bandits. It provides standardized tools for logging bandit feedback, implementing policy learning methods, and performing rigorous off-policy evaluation (OPE). Crucially, OBP includes both a modular Python library for algorithm development and a collection of benchmark datasets derived from real-world user interaction logs.

The dataset used in this project comes from the ZOZOTOWN fashion e-commerce platform, one of Japan's largest online fashion retailers. It simulates a recommendation environment where a policy selects an item to recommend based on user context, and a binary reward (click or no click) is observed. The logging policy employed to generate this dataset is a uniformly random policy over 80 items, allowing each action to be selected with equal probability. This uniformity is particularly valuable for off-policy evaluation, as it reduces bias from historical policy decisions and ensures broader coverage of the action space. In total, the dataset consists of over 1.3 million logged interactions, each consisting of a context vector, the selected action, the observed reward, and the corresponding propensity score.

Each context is represented by a high-dimensional binary vector capturing user features and browsing behavior. The action is the index of the recommended item (ranging from 0 to 79), and the reward is a binary indicator of whether the user clicked on the item. The propensity score - the probability with which the action was chosen - is constant at $1/80 = 0.0125$ due to the uniform behavior policy. These elements closely mirror the canonical setup of stochastic contextual bandits and allow us to focus entirely on the learning and evaluation aspects without having to model complex or biased logging policies.

The OBP dataset provides a rigorous testbed for exploring key questions in contextual bandit learning: How well do algorithms like LinUCB or Thompson Sampling perform under sparse rewards and large action spaces? Can off-policy evaluation methods reliably estimate policy performance in the presence of high variance and limited feedback? And how does computational cost scale with dataset size and model complexity? By working with logged interaction data rather than synthetic simulations, we aim to answer these questions under conditions that are closer to deployment in industrial recommendation systems.

We further benefit from OBP's extensive software support, which includes built-in implementations of many standard bandit algorithms, regression-based reward models, and multiple OPE estimators such as Inverse Propensity Weighting (IPW), Direct Method (DM), and Dou-

bly Robust (DR). The framework also includes diagnostic tools for checking reward sparsity, propensity score stability, and model fit quality. These tools allow for careful experimentation and model validation in a setting where ground-truth counterfactual outcomes are not observed.

3. *Logged Bandit Replay* with Selective Updating

Before conducting formal offline policy evaluation, we first explored a more intuitive and straightforward approach: simulating the deployment of contextual bandit algorithms on logged data through what we refer to as "logged bandit replay." The motivation behind this setup was to approximate how online learning algorithms such as LinUCB and Thompson Sampling would behave if directly deployed in a real-world setting, learning incrementally from user interactions over time. Since the Open Bandit Pipeline dataset provides logged historical data but no opportunity to collect new observations, we simulate online learning by iterating over the dataset sequentially and updating the learning algorithm's internal parameters only when the action it would have selected matches the action logged by the behavior policy.

This "selective updating" procedure introduces an inherent asymmetry: only those contexts for which the algorithm agrees with the historical action are used for training. The remaining rounds are ignored entirely, as they provide no reward feedback for the algorithm's own decisions. While this enables a form of online learning over static data, it introduces significant bias. The algorithm is effectively conditioning its updates on a non-random subset of the data - those that align with its current policy - which undermines the validity of the evaluation. As a result, this method does not satisfy the assumptions of off-policy evaluation, nor does it provide an unbiased estimate of a policy's long-run performance.

Nevertheless, we do not present this approach as a sound or rigorous evaluation method, but rather as an experimental attempt driven by our initial curiosity. It served more as a personal learning experience than a diagnostic tool. While the method arose from our idea to test online learning algorithms directly on real-world data, we ultimately recognized that this setup can be misleading if not properly constrained or interpreted. In contrast, we later turned to offline policy evaluation as a more principled and robust framework. We want to emphasize upfront that the experiments in this section are illustrative rather than conclusive and highlight the risks of naively applying online learning methods to logged data. These insights motivate the need for more rigorous and statistically grounded techniques when working in offline settings, instead of trying to simulate online learning based on logged data.

3.1. Data Preprocessing

Recall that the original Open Bandit Pipeline dataset comprises over 1.37 million logged interactions between users and items, where each interaction includes a high-dimensional, binary-encoded context vector, a selected action among 80 possible arms, and a binary reward indicating whether the user clicked the item. While this scale provides a rich experimental envi-

ronment, it also poses challenges for reliable learning and evaluation: the reward signals are "extremely" sparse, i.e., rewards only appear in 0.35% of the logged data[1], and the dimensionality of the context space is unnecessarily high, with many features being nearly always inactive. These characteristics introduce variance and slow down both learning and evaluation. As a result, we performed a series of preprocessing steps to obtain a reduced yet representative dataset for our initial exploration of contextual bandit algorithms.

As Figure 1 illustrates, many context features in the original dataset are rarely activated. For instance, several features are active in less than 5% of the samples, while a few are activated in over 30% of the cases (feature 1 in more than 50%). This skewed activation prompted us to remove sparse features from the context matrix as a first preprocessing step. Specifically, we dropped any feature that was active in fewer than 1% of the total samples. This simple yet effective filtering heuristic eliminated noise-prone or uninformative dimensions and reduced the feature space from 26 to 18 binary context features.

Furthermore, as shown in Figure 2, although the original context vectors are 26-dimensional, the vast majority of samples contain only between 2 and 4 active features. This highlights the sparsity of the representation and the limited number of informative features available per observation. To further denoise and compress the context vectors, we applied Principal Component Analysis (PCA), retaining 99% of the variance. This transformation reduced the context dimensionality to 12 continuous components, providing a more compact and informative input representation while preserving most of the original signal structure.

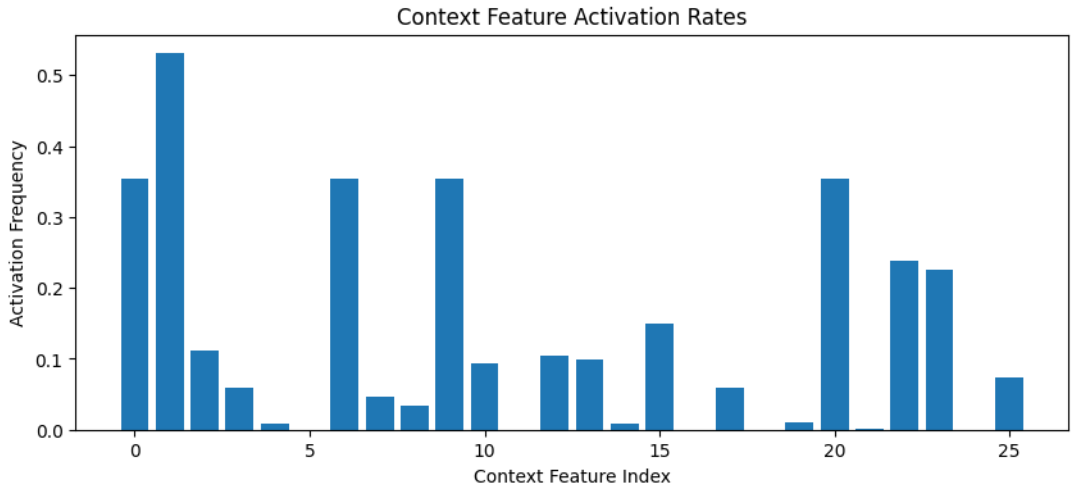


Figure 1: Feature-wise activation frequency of the 26 binary context features before preprocessing. Feature 1 appears in over 50% of samples, while many others are nearly inactive.

[1]This very sparse reward setting is especially problematic with our "naive" approach of only updating when the selected arm matches the logged arm, as we will show. Note, however, we had also tried with updating always based on the logged rewards, we found this to be even more biased though, since this is not what the algorithm actually would have observed.

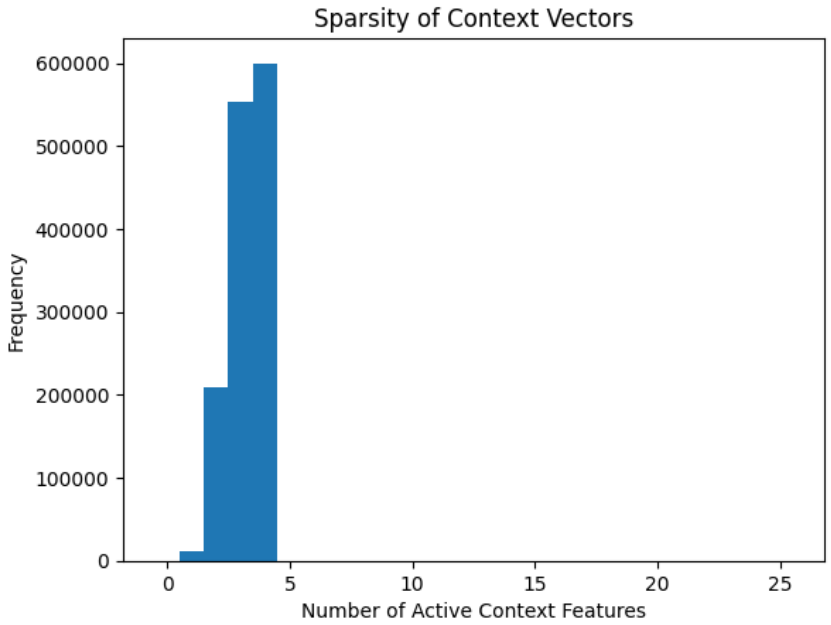


Figure 2: Histogram of the number of active features per context vector. Despite the original dimensionality, most samples have only 2 to 4 nonzero entries.

Next, we focused on simplifying the action space. The original dataset includes 80 possible arms, making it challenging to learn which action is optimal in a given context - especially under conditions of extreme reward sparsity. To address this, we constructed a filtered version of the dataset that retains only the three most frequently selected actions under the uniformly random logging policy. Although the logging policy selects actions uniformly at random, some arms appear slightly more often due to sampling variability. In fact, by selecting the three most frequently logged actions, we obtain approximately 2,500 more samples than would be expected from choosing three arms at random (i.e., three times the average number of samples per arm).[2] (These top arms appear more frequently in the logged data and therefore have greater overlap with any evaluation policy, increasing the likelihood of observing matching actions - particularly important for IPS and DR estimators.)

After filtering, the final dataset used for the experiments in this section consists of 53,988 samples. Each sample includes a 12-dimensional context vector, one of three re-indexed actions, the corresponding binary reward, and a constant propensity score of $1/3$. This focused subset strikes a practical balance: it preserves the structure of the contextual bandit setting while enabling faster and more reliable experimentation. While we also explored other configurations - including the full 80-arm dataset and a version restricted to the five most- and least-clicked arms - we restrict our attention in this section to the top-3-arm version for clarity and consistency.

[2]Note, however, that average click-through rates across all arms (marginal to context) were relatively uniform.

3.2. Bandit Algorithms

The algorithms studied in this section are standard contextual bandit algorithms designed for online learning - specifically, Linear UCB (LinUCB) and Linear Thompson Sampling (LinTS). These methods are based on the assumption that the expected reward of each action is a linear function of the observed context.[3] All algorithms in this section rely on ridge regression to estimate the expected reward for each arm as a linear function of the context. While the binary nature of the reward variable might suggest that a logistic model would be more appropriate, linear models remain common in contextual bandit literature due to their simplicity and efficiency. Notably, the original OBP paper by Saito et al., 2020 also employed linear models, even in binary reward settings.

3.2.1 Ridge Regression

We begin by formalizing the linear contextual bandit assumption. Let K denote the number of available arms. Each arm $a \in \{1,\dots,K\}$ is associated with an unknown parameter vector $\theta_a \in \mathbb{R}^d$, where d is the dimensionality of the context. At each round t, the algorithm observes a context vector $x_t \in \mathbb{R}^d$, and the expected reward of choosing arm a is given by

$$\mathbb{E}[r_t \mid a_t = a, x_t] = x_t^\top \theta_a.$$

Since θ_a is unknown, we estimate it using ridge regression. For each arm a, we maintain a regularized design matrix A_a and a cumulative reward-weighted context vector b_a, defined as

$$A_a = \lambda I + \sum_{s:a_s=a} x_s x_s^\top, \qquad b_a = \sum_{s:a_s=a} r_s x_s.$$

The ridge regression estimate for the arm's reward parameter is then

$$\hat{\theta}_a = A_a^{-1} b_a.$$

Here, A_a serves as a regularized covariance matrix that accumulates second-order statistics of the observed contexts for arm a, while b_a accumulates the corresponding rewards. The hyperparameter $\lambda > 0$ controls the degree of regularization, incorporating a prior on the weight magnitude and stabilizing the estimates when data is limited.

[3] As mentioned before, our primary goal was to examine how these algorithms perform when deployed in a cold-start setting, i.e., without pretraining or access to a simulator. Despite this goal, we emphasize again that applying online learning algorithms directly to logged data - as done here via selective replay - is not methodologically sound for policy evaluation.

3.2.2 Linear UCB

The LinUCB algorithm balances exploitation of high-reward actions with exploration of uncertain ones. For each arm a and observed context x_t, LinUCB computes an upper confidence bound score:

$$p_a(x_t) = x_t^\top \hat{\theta}_a + \alpha \cdot \sqrt{x_t^\top A_a^{-1} x_t},$$

where the first term encourages exploitation of actions with high predicted reward, and the second term encourages exploration of actions with high uncertainty. The parameter $\alpha > 0$ controls this trade-off. The arm selected at round t is given by

$$a_t = \arg\max_a \ p_a(x_t).$$

If the selected arm a_t matches the logged action a_t^{logged}, the algorithm updates its statistics as follows:

$$A_{a_t} \leftarrow A_{a_t} + x_t x_t^\top, \qquad b_{a_t} \leftarrow b_{a_t} + r_t x_t.$$

This ensures that only on-policy feedback is incorporated, which is consistent with our selective update strategy in this replay-based offline setup.

3.2.3 Linear Thompson Sampling

In contrast to LinUCB, which uses deterministic upper confidence bounds, Linear Thompson Sampling adopts a Bayesian approach by sampling from a posterior distribution over the parameters θ_a. At each round, we compute the ridge estimate $\hat{\theta}_a = A_a^{-1} b_a$, and then draw a sample from the posterior distribution:

$$\tilde{\theta}_a \sim \mathcal{N}(\hat{\theta}_a, \alpha^2 A_a^{-1}),$$

where $\alpha^2 A_a^{-1}$ serves as the covariance of the posterior distribution. The algorithm selects the arm that maximizes the sampled reward:

$$a_t = \arg\max_a \ x_t^\top \tilde{\theta}_a.$$

If the selected arm matches the logged action, we again perform a selective update:

$$A_{a_t} \leftarrow A_{a_t} + x_t x_t^\top, \qquad b_{a_t} \leftarrow b_{a_t} + r_t x_t.$$

The exploration-exploitation balance is governed by the parameter α, which scales the posterior variance. Larger values of α result in more exploratory behavior, especially in early rounds when uncertainty is high.

Together, these algorithms offer complementary strategies for learning from contextual interactions: LinUCB uses confidence intervals to guide deterministic exploration, while Thompson

Sampling introduces stochasticity through posterior sampling. While their use in a replay-based offline setting introduces bias and limits interpretability, they remain valuable tools for analyzing bandit dynamics under realistic logging constraints.

3.3. Evaluation Metrics

A common way to evaluate bandit algorithms is through the lens of *cumulative reward* or *cumulative regret*. In an ideal setting, where the reward distribution of each arm is known or can be simulated, one can measure how much reward an algorithm accumulates over time compared to an optimal policy. This approach allows direct computation of regret - the difference between the reward obtained by the algorithm and the reward of the best possible policy. However, such evaluation requires access to the ground truth reward distribution for all possible actions in all possible contexts, which is not available in our real-world dataset.

In our setting, we only observe the rewards for the actions that were selected by the behavior policy (i.e., the logging policy), which in this case is a uniformly random policy. Since we do not know the expected rewards for actions that were not chosen, traditional regret-based evaluation is infeasible. Instead, we rely on methods from *off-policy evaluation* (OPE), which are specifically designed to estimate the performance of new policies using logged bandit data.

A naive approach to evaluation in this context is to calculate the *cumulative logged reward* of the policy. This means simulating the policy on the logged data and summing up the rewards for rounds where the policy happens to choose the same action as the one logged. While this method is simple and reflects actual observed rewards, it leads to biased underestimates of performance. Most notably, it ignores all rounds in which the evaluation policy selects an action different from the one taken in the log, even if that alternative action might have led to a reward. This makes naive logged reward evaluation unreliable and overly pessimistic, especially when there is limited overlap between the evaluation policy and the logging policy.

To address this bias, we adopt *Inverse Propensity Scoring* (IPS). IPS corrects for the mismatch between the evaluation and logging policies by reweighting the observed rewards. Specifically, if the evaluation policy selects the same action as the logged policy, the observed reward is scaled by the inverse of the logging policy's propensity score, which is the probability of that action being selected under the behavior policy. For a uniform random logging policy with K actions, the propensity score is $1/K$, and the IPS weight is therefore K. This yields the unbiased estimator:

$$\hat{V}_{\text{IPS}} = \frac{1}{n} \sum_{i=1}^{n} \frac{\mathbb{I}\{a_i = \pi(x_i)\}}{p(a_i \mid x_i)} r_i$$

where $\pi(x)$ is the target policy, $p(a \mid x)$ is the propensity of the logging policy, and r_i is the observed reward. While IPS provides an unbiased estimator under correct propensity scores, it suffers from high variance, especially in cases with low overlap or sparse rewards. This behavior has been well documented in the contextual bandits literature (e.g., Wang et al., 2017).

To further enhance the stability and reliability of our offline evaluation, we adopt the *Dou-*

bly Robust (DR) estimator. DR combines the strengths of two complementary approaches: the *Direct Method* (DM), which uses a learned reward model to estimate expected rewards for any context-action pair, and the IPS. When the evaluation policy selects the same action as the logged one, the DR estimator adjusts the predicted reward using the observed outcome and the inverse of the propensity score. In cases without action overlap, it falls back on the model-based estimate. This hybrid structure makes DR robust: it remains consistent if either the reward model or the propensity model is well-specified, and it generally achieves lower variance than IPS alone. Formally, the DR estimator is given by:

$$\hat{V}_{\mathrm{DR}} = \hat{r}(x_i, \pi(x_i)) + \frac{\mathbb{I}\{\pi(x_i) = a_i\}}{p(a_i \mid x_i)} \big(r_i - \hat{r}(x_i, a_i)\big),$$

where $\hat{r}(x,a)$ is the predicted reward from our random forest model, $\pi(x)$ is the action chosen by the evaluation policy for context x, a_i and r_i are the logged action and observed binary reward, and $p(a_i \mid x_i)$ is the logging policy's propensity score. When $\pi(x_i) = a_i$, DR applies a correction to the predicted reward using the observed outcome; otherwise, it relies entirely on $\hat{r}(x_i, \pi(x_i))$.

In our implementation, the reward model is a random forest regressor trained on the logged data, predicting expected reward for any context-action pair. This allows DR to leverage both model generalization and empirical evidence. Under real-world conditions - especially with sparse rewards and low action overlap - DR has been shown to lower variance relative to IPS while also being more robust than the Direct Method alone Dudík et al., 2011; Su et al., 2020.

We leverage this DR estimator as our principal evaluation metric, combining the flexibility of a non-linear reward model with the statistical robustness of the DR framework to deliver reliable policy value estimates in challenging, low-overlap logged data environments.

3.4. Results

If not stated differently, we present our experimental results using the *Linear Upper Confidence Bound* (LinUCB) algorithm with an exploration parameter of $\alpha = 1.0$. We begin by analyzing the performance under a naive evaluation scheme before introducing more principled off-policy estimators and then presenting some average results.

Naive Evaluation

As a baseline approach, we simulate the learned LinUCB policy over the logged dataset and track the cumulative reward it accrues. However, since the dataset only contains feedback for actions selected by the behavior policy, this method can only assign reward to the evaluation policy if it selects the exact same action as the one logged at a given round. In practice, the overlap between LinUCB's chosen actions and the logged actions is limited, especially in the early learning phase and in high-dimensional action spaces.

Consequently, the resulting cumulative reward for LinUCB appears deceptively low, as

shown in Figure 3. In contrast, the random policy (i.e., the behavior policy used to generate the logged data) accumulates reward at a much faster rate, since it receives observed rewards in every round. This naive logged evaluation underestimates the performance of LinUCB and fails to account for the policy's potentially superior decisions in rounds where the chosen action does not match the logged one.

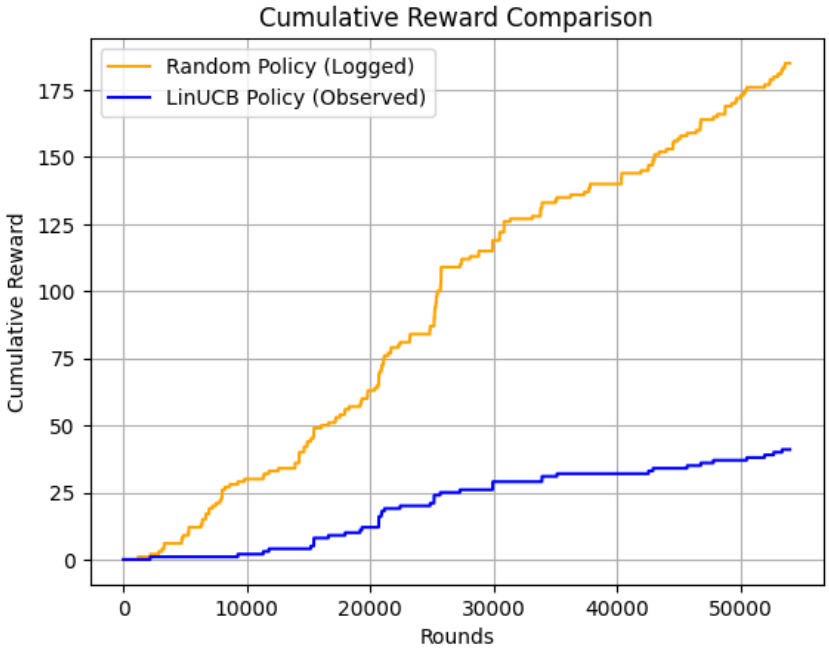


Figure 3: **Cumulative Reward Comparison under Naive Evaluation.** The blue line shows the cumulative reward observed by LinUCB only in rounds where its action matched the logged one. The orange line represents the reward accumulated by the behavior policy (uniformly random). Due to limited action overlap, this naive approach severely underestimates the evaluation policy's performance.

Inverse Propensity Scoring

Next, we applied IPS; specifically, if the evaluation policy π selects the same action as the logged action a_i under context x_i, we include the observed reward r_i, scaled by $1/p(a_i \mid x_i)$. Otherwise, the contribution is zero. However, this still relies exclusively on rounds where the evaluation and behavior policies align, meaning it suffers from high variance in settings with low action overlap. In our dataset, LinUCB frequently chooses actions different from those taken by the random logging policy, and the binary rewards are sparse. As a result, the cumulative IPS estimate for LinUCB remains substantially lower than the IPS-estimated reward of a uniformly random policy. This gap illustrates the inherent difficulty of offline evaluation in low-overlap, sparse-reward environments.

Figure 4 compares four curves: the logged reward of the random policy (orange), IPS-corrected reward for LinUCB (purple), IPS-corrected reward for the logging policy itself (gray), and IPS for a simulated random policy (green). While LinUCB's IPS performance shows some

improvement over logged naive evaluation, it remains far below that of the baseline policies due to limited matching opportunities.

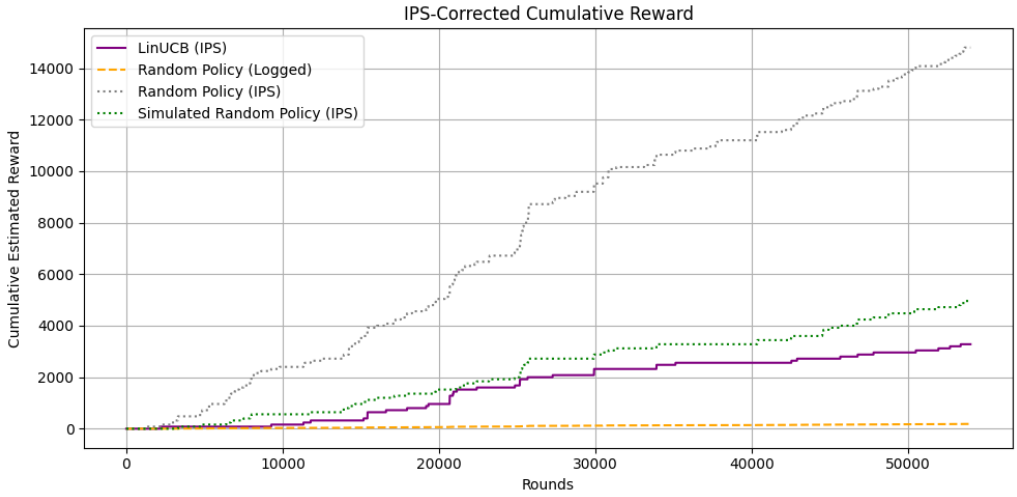


Figure 4: **IPS-Corrected Cumulative Reward.** Comparison of LinUCB (purple) against the logging policy (orange), the logging policy corrected via IPS (gray), and a simulated random policy (green). Due to limited action overlap and sparse rewards, LinUCB's IPS estimate remains lower despite learning a potentially better policy.

Doubly Robust Estimation

Finally, we evaluate the learned policy using the DR estimator. The DR estimates shown in Figure 5 are based on *non-clipped predictions*, allowing both model-based and corrected rewards to go beyond the $[0, 1]$ range. While this increases expressiveness, it may lead to negative cumulative rewards early on due to overcorrection—especially if the reward model inaccurately estimates low-probability actions.

In our results, the simulated random policy surprisingly outperforms LinUCB by a significant margin. This counterintuitive outcome arises because the simulated random policy explores uniformly and occasionally matches the logged actions that yielded positive rewards—these rare overlaps are strongly upweighted by the DR correction. In contrast, LinUCB tends to exploit early and may fail to match rewarding logged actions, particularly in the cold-start phase where model estimates are still uncertain.[4]

[4]*Note:* The cumulative sum of predicted rewards under the model (`r_hat`) was 189.01, closely aligned with the total number of observed rewards in the dataset (185). This indicates that the reward model was reasonably well-calibrated overall, but individual misestimations are still heavily affect the DR correction when combined with inverse propensity weights.

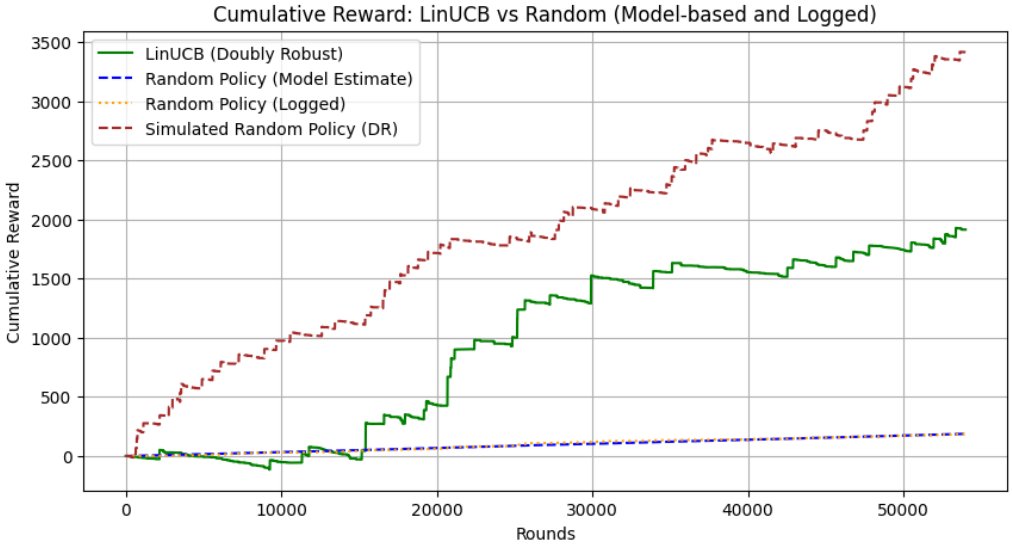


Figure 5: **Doubly Robust Evaluation of LinUCB and Random Policies.** The green curve shows the DR estimate for LinUCB. The red dashed curve corresponds to a simulated random policy under the same DR evaluation, while the blue and orange curves represent the model-based and logged estimates of the original random logging policy, respectively. Due to rare but highly upweighted matches, the random policy appears to outperform LinUCB.

LinUCB with Different Alpha Values

We now explore how LinUCB behaves under varying levels of exploration by changing the value of the hyperparameter α, which controls the exploration-exploitation tradeoff. Specifically, we compare four settings: $\alpha = 0.0, 0.1, 0.5$, and 1.0. A value of $\alpha = 0.0$ corresponds to a purely greedy policy with no exploration, while higher values of α introduce increasing levels of exploration by adding uncertainty-based bonuses to each arm's score. We evaluate each of these policies using a single run and estimate their performance using the DR estimator.

Figure 6 presents the cumulative DR-estimated rewards for each LinUCB policy. We observe that the greedy variant ($\alpha = 0.0$) achieves the best performance across the entire horizon. The light exploration setting ($\alpha = 0.1$) also performs strongly, closely tracking the greedy policy. As we increase exploration to $\alpha = 0.5$ and $\alpha = 1.0$, performance deteriorates noticeably. The heavily exploratory version ($\alpha = 1.0$) learns more slowly and fails to match the performance of even a simulated random policy. Interestingly, the policy with $\alpha = 0.5$ performs similarly to the simulated random baseline, which selects actions independently of context. This suggests that in this offline setting - characterized by sparse rewards and limited action overlap - aggressive exploration may harm learning. In contrast, the lower-α policies manage to outperform the random baseline, likely because they are better able to exploit the few high-reward opportunities that align with the logged data.

While these findings are based on a single simulation run, they suggest that low exploration may be preferable when overlap between the evaluation and logging policies is rare, and the signal-to-noise ratio is low. We explore this result further in the next section by averaging

across multiple shuffled evaluations.

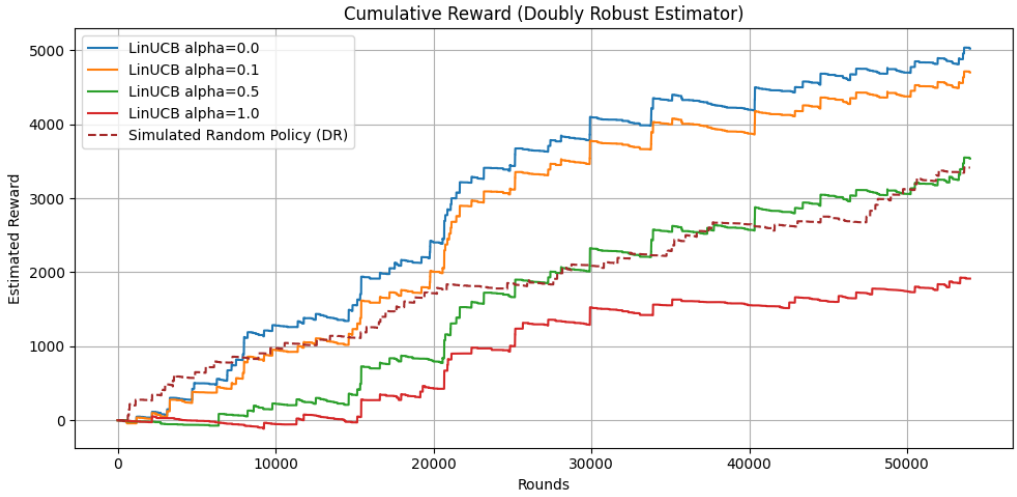


Figure 6: **Cumulative Doubly Robust Estimated Reward for Different LinUCB Exploration Parameters.** LinUCB with $\alpha = 0.0$ (purely greedy) performs best, followed closely by $\alpha = 0.1$. Higher exploration levels ($\alpha = 0.5$ and $\alpha = 1.0$) result in worse performance. The dashed red line represents a simulated random policy evaluated using DR.

Average Evaluation with Precomputed LinUCB Policies

We now evaluate how the different LinUCB variants perform when their action sequences are fixed in advance, but the evaluation is repeated across 30 different shuffled versions of the dataset. In each run, the logged contexts, actions, rewards, and propensity scores are randomly permuted, and a new reward model is trained. However, the policy actions generated by each LinUCB variant remain fixed across all runs. This setup allows us to assess the robustness of each policy under varying reward model fits and exposure to different parts of the data, while isolating the policy's own behavior from randomness in training.

Figure 7 shows the average cumulative reward estimated using the Doubly Robust (DR) estimator for three LinUCB variants with $\alpha = 0.0$, 0.5, and 1.0. As before, the $\alpha = 0.0$ policy - greedy without exploration - achieves the best performance. The variant with $\alpha = 0.5$ performs considerably worse, and $\alpha = 1.0$ yields the lowest cumulative reward of the three. This confirms earlier observations that excessive exploration can be detrimental in offline settings where the evaluation relies on limited action overlap and where reward signals are sparse.

The dashed brown curve in the plot represents the average DR performance of a simulated random policy, evaluated in the same way over the same shuffled datasets. Interestingly, the LinUCB policy with moderate exploration ($\alpha = 0.5$) performs comparably to this random baseline, while the high-exploration policy ($\alpha = 1.0$) performs worse. The greedy policy ($\alpha = 0.0$), in contrast, substantially outperforms the random baseline across the entire horizon.

These results underscore a key insight: in environments with sparse rewards and limited logging overlap, conservative exploration can be more effective than aggressive exploration. Since LinUCB with $\alpha = 0.0$ is purely exploitative, it benefits from early exploitation of predic-

tive context features, whereas more exploratory variants may waste valuable logged feedback on suboptimal or poorly aligned actions.

We emphasize that this analysis is based on fixed action sequences and varying reward models across runs. In the next section, we present true averages that recompute both model and policy in each run, providing an even more comprehensive evaluation.

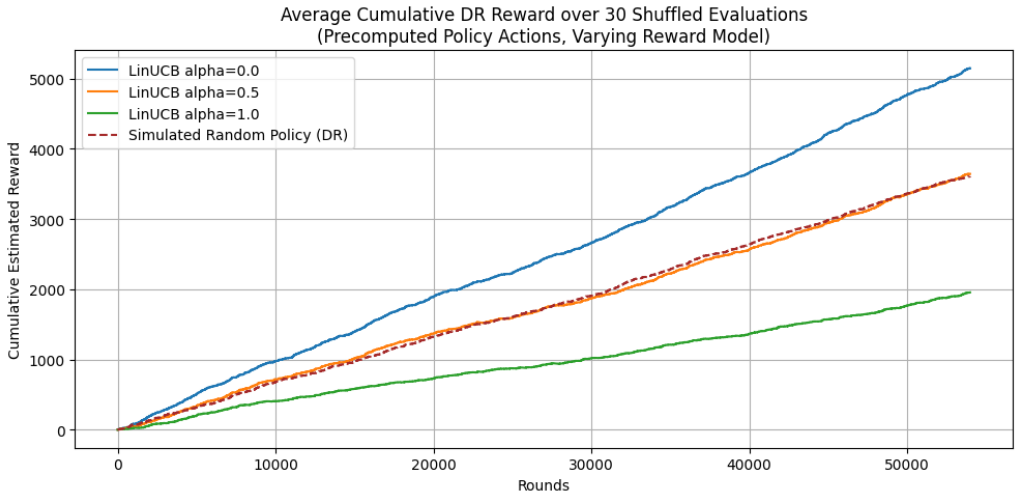


Figure 7: **Average cumulative DR-estimated reward over 30 shuffled evaluations.** LinUCB with $\alpha = 0.0$ (greedy) consistently outperforms the other variants. Moderate exploration ($\alpha = 0.5$) performs similarly to the random policy baseline, while heavy exploration ($\alpha = 1.0$) underperforms. These results highlight the value of conservative exploration in low-reward, low-overlap offline bandit settings.

True Averages Across Multiple Runs

In this final experiment, we conduct a robust evaluation of contextual bandit policies by averaging performance across multiple randomized runs. For each of the 30 runs, we reshuffle the dataset, retrain a new reward model using a random forest regressor, and recompute both the LinUCB and Thompson Sampling (TS) policies from scratch. This procedure ensures that the cumulative reward estimates are not biased by a specific data order or fixed model initialization, offering a more stable and generalizable comparison.

We evaluate LinUCB with $\alpha \in \{0.0, 0.1, 0.25\}$ and TS with $\alpha \in \{0.25, 0.5, 1.0\}$, alongside a simulated random policy (uniform arm selection), using the Doubly Robust (DR) estimator for each run. Unlike LinUCB, where $\alpha = 0$ corresponds to purely greedy behavior, the TS algorithm does not support $\alpha = 0$. Instead, the α parameter in TS controls the scale of the posterior variance—larger values lead to more exploration through greater sampling noise.

Figure 8 presents the average DR-corrected cumulative rewards. The results show that all LinUCB variants consistently outperform the random policy baseline, though by only a modest margin. The greedy LinUCB policy ($\alpha = 0.0$) achieves the highest cumulative reward, especially as the number of rounds increases. This suggests that under sparse reward conditions with moderately informative context features, a conservative approach may yield the most reli-

able results. For TS, the same pattern largely holds. The most exploratory variant with $\alpha = 1.0$ underperforms compared to both LinUCB and the simulated random policy, while more conservative TS configurations (e.g., $\alpha = 0.25$) perform comparably to the best UCB variants.

These findings highlight that, in offline learning scenarios with limited action overlap and sparse rewards, excessive exploration can degrade performance. Instead, stable and conservative strategies such as greedy LinUCB or low-variance TS can offer more dependable outcomes. Still, we emphasize that this result may not generalize to tasks with richer reward structures or larger context-action variability.

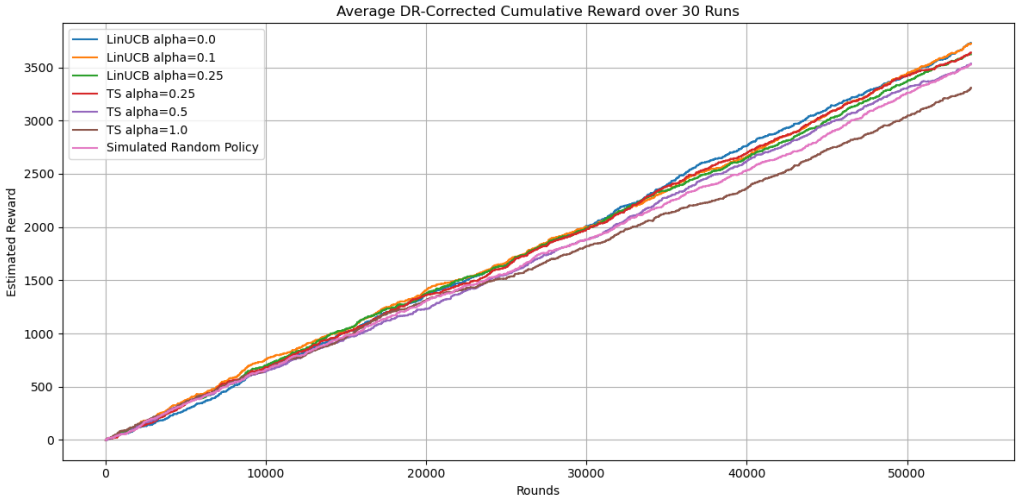


Figure 8: **Average DR-corrected cumulative reward over 30 runs.** LinUCB and Thompson Sampling (TS) variants with varying α values are evaluated and compared against a simulated random policy. Greedy LinUCB ($\alpha = 0.0$) achieves the best overall performance, while highly exploratory TS performs worst. Conservative strategies prove more effective in sparse offline settings.

In addition to varying the exploration factor α, we further investigate the role of the prior strength parameter λ in Thompson Sampling. Specifically, we fix $\alpha = 0.25$ - the best-performing TS configuration from the previous evaluation - and vary $\lambda \in \{0.1, 1.0, 10.0\}$ to assess how different levels of regularization influence learning.

Thompson Sampling maintains a Gaussian posterior over each arm's parameter vector. The posterior covariance is defined as $\Sigma_a = \alpha^2 A_a^{-1}$, where $A_a = \lambda I + \sum_{i \in \mathcal{D}_a} x_i x_i^\top$. Here, λ acts as a prior precision term, regulating how strongly the algorithm relies on prior assumptions. A smaller λ allows faster adaptation to data but introduces more noise, while a larger λ enforces conservative updates that reduce variance at the cost of responsiveness.

Figure 9 compares the average DR-corrected cumulative reward for the different λ values under fixed $\alpha = 0.25$, alongside the greedy LinUCB baseline ($\alpha = 0.0$) and the simulated random policy. Interestingly, the performance across all TS variants remains closely aligned regardless of λ, indicating that in this particular setting, reward sparsity and limited overlap dominate the influence of prior strength. No configuration significantly outperforms the others,

and all remain competitive with the best-performing LinUCB baseline.

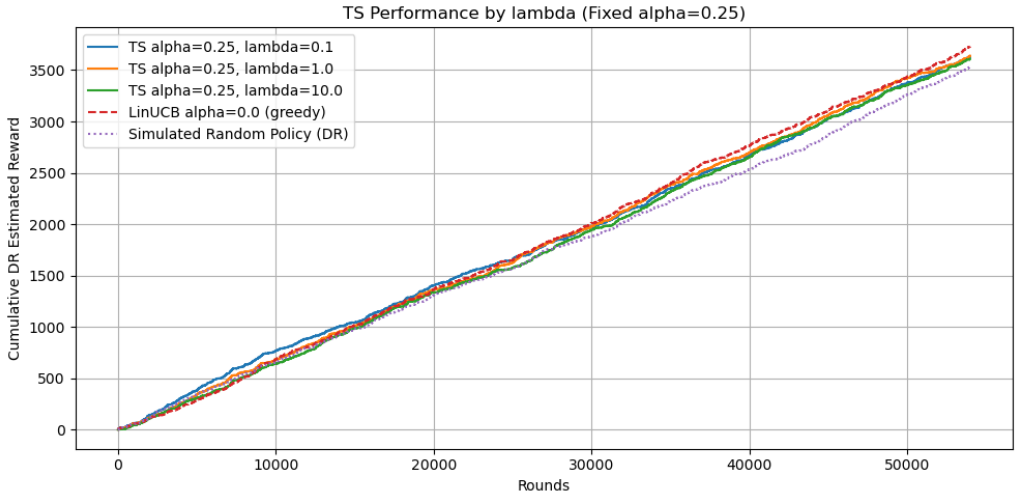


Figure 9: **Effect of Prior Strength λ on Thompson Sampling.** All TS variants share fixed $\alpha = 0.25$ and differ only in prior strength $\lambda \in \{0.1, 1.0, 10.0\}$. The results show minimal performance variation across configurations, suggesting that λ has limited impact under sparse-reward and low-overlap conditions.

4. Synthetic Benchmarking

Setup and Dataset

The goal of this section is to evaluate contextual bandit algorithms in a controlled synthetic environment where the ground-truth reward function is known. By using a synthetic enivornment, we can test our bandit algorithms, calculate cumulative reward and regret, and compare the results to the ground truth.

The synthetic dataset we generate has the following properties:

- Context Dimension: 5
- Arms/Actions: 5
- Rounds: 3,000

To simulate realistic binary outcomes like clicks, the reward function used is logistic sigmoid and is given by equation 1

$$r_t \sim \text{Bernoulli}(\sigma(x_t \theta_t)) \tag{1}$$

We can see that in this simple setting, the average reward for each arm is slightly variable, and hence exploitable by an efficient algorithm (see Figure 10)

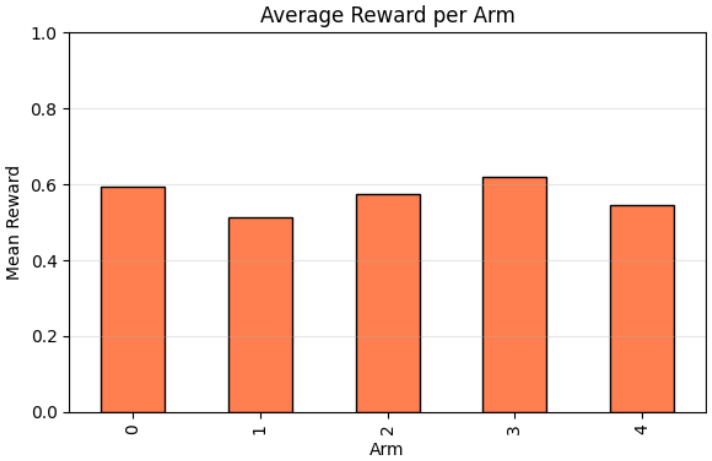


Figure 10: Average reward per arm in the synthetic environment

We evaluated the following algorithms:

- **LinUCB** with $\alpha \in 0.0, 0.1, 0.5$, where $\alpha = 0.0$ is purely greedy
- **LinTS (Linear Thompson Sampling)** with variance parameter $\alpha = 1.0$.
- **Linear ε-Greedy** with exploration rate $\varepsilon = 0.1$.
- **Random Policy** as a simple, non-learning baseline.

We tried offline evaluation, as well as online policy simulation.
We evaluated the policies using two standard offline evaluation methods:

- **Replay Method (RM):** A simple baseline that estimates performance using only those data points where the action chosen by the policy matches the logged action. This method is unbiased but often suffers from high variance due to limited matching samples.
- **Doubly Robust (DR):** A more advanced method that combines two sources of information: a learned reward model (which provides predictions for all actions) and importance weighting (which corrects for discrepancies between the policy and the logged data). The DR method remains accurate as long as either the reward model or the logging policy is estimated well, making it more reliable in practice.

We also tried online simulation, where we had each algorithm interact with the environment for 3000 rounds. This allowed us to track cumulative regret and reward that each algorithm managed to obtain. To increase robustness of our results, we averaged the results over 30 independent runs.

Results

Offline Evaluation

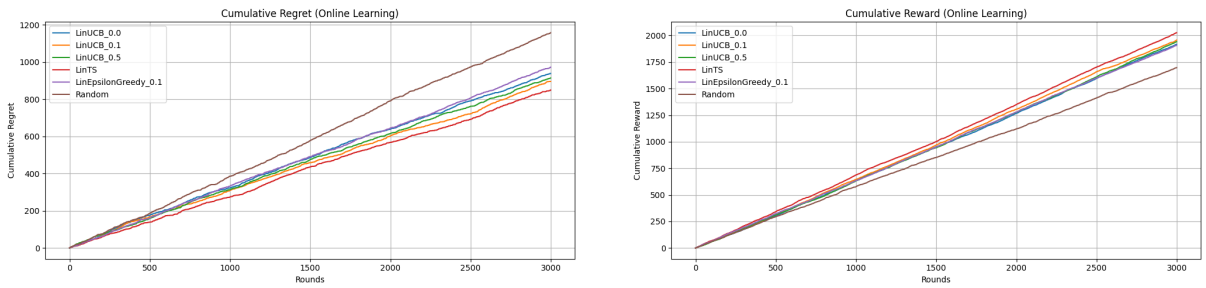
Table 1 shows the results for the offline evaluation methods on the synthetic data. We observe that the replay method and doubly robust results are practically the same, and both underestimate the ground truth average reward. So in this case, with the limited environment, we still cannot use these offline evaluation methods consistently to achieve near ground truth results, but this could be due to the synthetic data we created.

Table 1: Offline Policy Value Estimates vs. Ground Truth - Average reward per round.

Policy	**Ground Truth**	**Replay Method (RM)**	**Doubly Robust (DR)**
LinUCB ($\alpha = 0.1$)	0.681	0.611	0.608
LinUCB ($\alpha = 0.5$)	0.666	0.602	0.607
LinUCB ($\alpha = 0.0$)	0.660	0.614	0.614
LinTS ($\alpha = 1.0$)	0.656	0.572	0.579
LinEpsilonGreedy ($\varepsilon = 0.1$)	0.652	0.609	0.605

Online Learning Simulation

Figure 11 shows the cumulative regret and cumulative reward obtained when running the online simulation over 3,000 rounds. We can observe that the random policy is beat by all of our bandit algorithms, and the winning policy in this case seems to be Thompson Sampling. However, due to the limited rounds, we wanted to check if this holds up even if run on different seeds, hence we ran the simulation 30 times using different seeds, shown in figure 12. The result that we see is again that each policy outperforms the random policy, however, the difference between learning policies is very marginal. It would be an interesting extension to explore how the algorithms perform if we allow them to run for 30,000 rounds,however, due to computational limitations, we only explored this up to 3,000.



(a) Cumulative Regret - 3,000 rounds

(b) Cumulative Reward - 3,000 rounds

Figure 11: Performance comparison for at total of 3,000 rounds

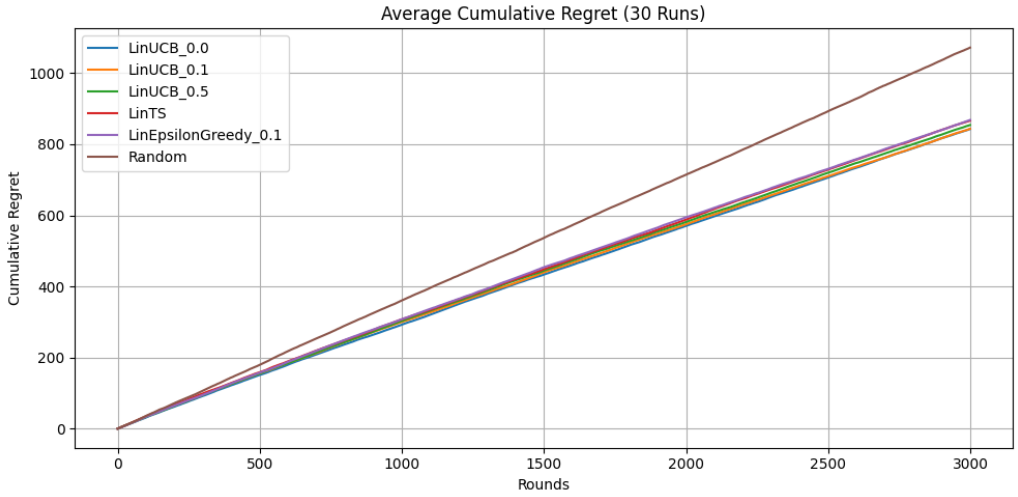


Figure 12: Average Cumualtive Regret over 30 independent runs.

5. True Offline Policy Evaluation

In this section, we evaluate fixed policies using logged bandit feedback alone, without any online interaction or policy retraining. This is a core challenge in contextual bandits—estimating the expected performance of a policy solely based on historical data.

We apply the earlier discussed three standard Off-Policy Evaluation estimators: IPW, DM, and DR. These estimators are applied to multiple policies, including random, Thompson Sampling, and LinUCB variants. Crucially, the evaluation is carried out on both the reduced dataset with only the top-3 most frequent arms, and on the full dataset containing all 80 arms (both with reduced context dimension to 12). This enables us to analyze how the amount of action space overlap influences estimator reliability and confidence intervals.

To ensure fairness and consistency, we use the same dataset from the previous sections, now preprocessed to include only the relevant arms and context features. Propensity scores are corrected accordingly to reflect true logging probabilities—for the reduced dataset with 3 arms, these are set to $1/3$ uniformly.

Results on Reduced Dataset (Top-3 Arms)

Figure 13 shows the estimated policy values across different estimators (IPW, DM, DR) along with their corresponding 95% confidence intervals. We observe that all estimators return similar values for each policy, and importantly, all confidence intervals overlap substantially. This indicates that under the reduced dataset, we cannot claim any statistically significant performance difference between policies.

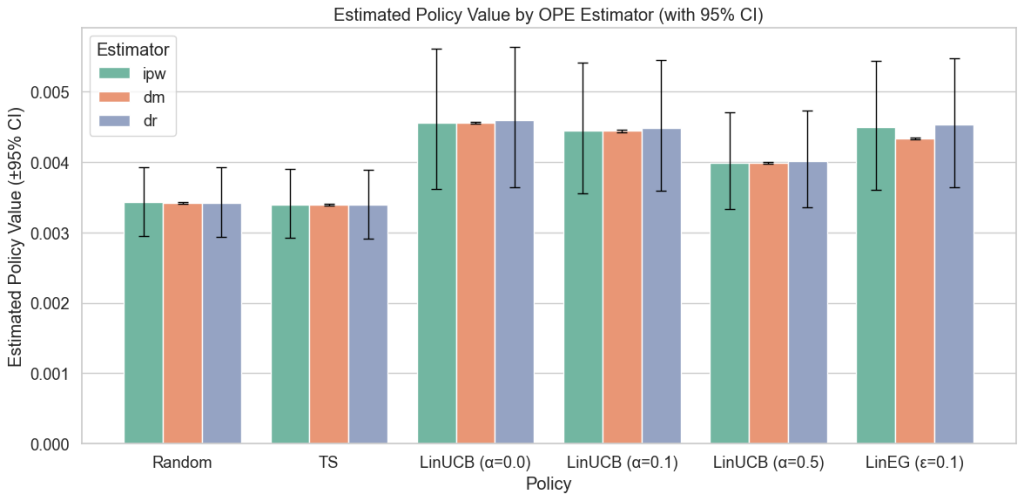


Figure 13: **Estimated Policy Value using OPE estimators (Reduced Dataset).** All confidence intervals overlap, suggesting no statistically significant difference in policy performance under this regime.

Results on Full Dataset (80 Arms)

Figure 14 displays the same evaluation procedure but now on the full dataset with 80 arms and true logged propensity scores. The outcome is surprisingly consistent with the reduced case: policy value estimates remain similar and again show large overlapping confidence intervals. The primary distinction lies in the noticeably tighter intervals, which is likely a result of the increased volume of logged data per action. However, the general conclusion remains unchangeddisc - riminating among the policies based on these offline estimates remains difficult.

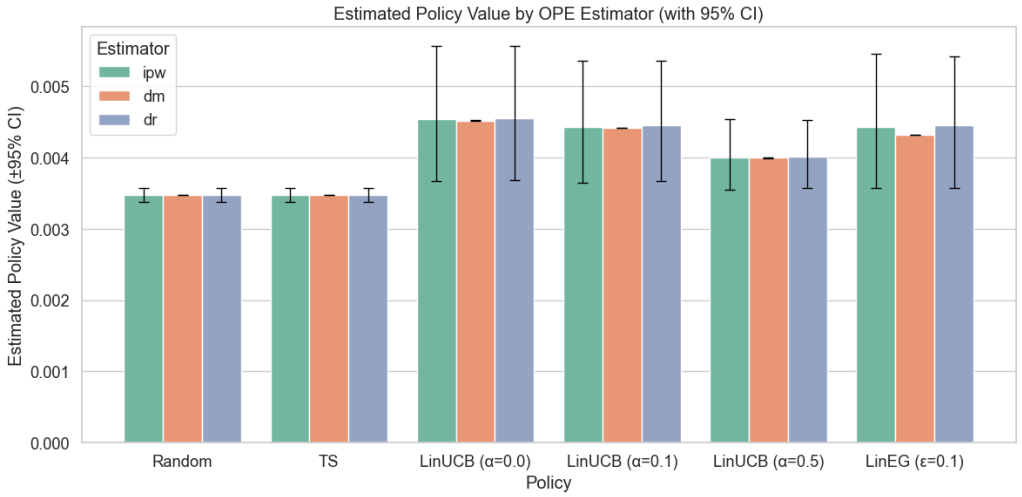


Figure 14: **Estimated Policy Value using OPE estimators (Full Dataset).** Wider coverage of arms improves confidence precision, but overlapping intervals still prevent confident ranking of policies.

Discussion and Limitations

Despite extensive effort to ensure correct propensity scores, consistent estimator implementations, and robust reward modeling, the results of our true offline evaluation remain inconclusive. While the DR estimator in particular shows lower variance, the confidence intervals for all estimators still overlap, even across policies with seemingly different learning behaviors. This highlights the limitations of OPE in practice - especially in settings with limited or sparse logged data and weak signal strength.

We believe further refinement is needed to improve these evaluations, potentially by incorporating additional reward signal modeling, better policy diversity, or leveraging more advanced estimators. Nonetheless, our findings are valuable in that they illustrate the fragility of OPE conclusions under real-world data limitations.

6. Conclusion

Throughout this project, we explored the practical application of contextual bandit algorithms to real-world logged data using the Open Bandit Pipeline. Our aim was to evaluate the performance of classic algorithms such as LinUCB and Thompson Sampling under realistic constraints - including sparse binary rewards, large action spaces, and limited policy overlap.

By implementing both online-style learning via logged replay and formal offline policy evaluation using OPE estimators (IPW, DM, DR), we gained a deeper appreciation of the challenges in deploying contextual bandits in practice. We experimented with selective updating, reward modeling, and extensive evaluation protocols. Along the way, we encountered computational bottlenecks, estimator variance issues, and surprising results - particularly the difficulty of distinguishing policy performance when confidence intervals overlap substantially.

Importantly, we also recognize that not all strategies we tried (e.g., selective updating or naive replay) reflect best practices in the field. Yet, experimenting with these approaches allowed us to better understand the conceptual and methodological foundations of contextual bandits and OPE. The hands-on implementation gave us practical insight into when and why certain estimators or algorithmic choices may fail, even if they appear theoretically sound.

Overall, this project has been an invaluable learning experience. It deepened our technical understanding and sparked new questions about how to design more robust learning systems under real-world constraints.

References

Dudík, Miroslav; Langford, John & Li, Lihong (2011), "Doubly Robust Policy Evaluation and Learning". In: *Proceedings of the 28th International Conference on Machine Learning (ICML)*, pp. 1097–1104.

Saito, Yusuke; Maeda, Shinichi; Shiroiwa, Takuya; Noda, Kota; Yuta, Saito, et al. (2020), "Open Bandit Dataset and Pipeline: Towards Realistic and Reproducible Off-Policy Evaluation". In: *NeurIPS 2020 Offline Reinforcement Learning Workshop*. URL: https://arxiv.org/abs/2008.07146.

Su, Yi; Dimakopoulou, Maria; Krishnamurthy, Akshay & Dudík, Miroslav (2020), "Doubly Robust Off-Policy Evaluation with Shrinkage". In: *Proceedings of the 37th International Conference on Machine Learning (PMLR)*. Vol. 119, pp. 9167–9176.

Wang, Yu-Xiang; Agarwal, Alekh & Dudík, Miroslav (2017), "Optimal and Adaptive Off-policy Evaluation in Contextual Bandits". In: *Proceedings of the 34th International Conference on Machine Learning*. Vol. 70. PMLR. PMLR, pp. 3584–3593.