# Contextual Bandits in Payment Processing: Non-uniform Exploration and Supervised Learning

Akhila Vangara*
Adyen

Alex Egg*
Adyen

## Abstract

Uniform random exploration in contextual bandits supports off-policy learning via supervision but incurs high regret, making it impractical for many applications. Conversely, non-uniform exploration offers better immediate performance but lacks support for off-policy learning. Recent research suggests that regression oracles can bridge this gap by combining non-uniform exploration with supervised learning. In this paper, we analyze these approaches within a real-world industrial context at Adyen, a large global payments processor characterized by batch logged delayed feedback, short-term memory, and dynamic action spaces under the Empirical Risk Minimization (ERM) framework. Our analysis reveals that while regression oracles significantly improve performance, they introduce challenges due to rigid algorithmic assumptions. Specifically, we observe that as a policy improves, subsequent generations may perform worse due to shifts in the reward distribution and increased class imbalance in the training data. This effect arises when regression oracles influence probability estimates and the realizability of subsequent policy models, leading to fluctuations in performance across iterations. Our findings highlight the need for more adaptable algorithms that can leverage the benefits of regression oracles without introducing instability in policy performance over time.

**ACM Reference Format:**
Akhila Vangara and Alex Egg. 2025. Contextual Bandits in Payment Processing: Non-uniform Exploration and Supervised Learning. In *Companion Proceedings of KDD '25: Online and Adaptive Recommender Systems (OARS '25).* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

## 1 Introduction

Adyen is a global payments processor with a diverse product suite including tools to optimize transaction authorization rates. Various interventions, conditioned on context, can be applied at payment-time to boost transaction authorization rates, naturally framing this problem in a contextual bandit setting where interventions correspond to actions and authorization feedback from the bank (environment) serve as rewards. It is instructive to view this system in the framework of a recommender system.

In many industrial recommender system applications, learning from historical logs—often collected under biased feedback conditions—is crucial. This paper investigates how traditional supervised learning, via Empirical Risk Minimization (ERM), can be combined with decision-making in interactive settings as formalized by contextual bandits. Our aim is to leverage supervised signals from logged data to inform more effective exploration strategies.

Empirical Risk Minimization (ERM) underpins traditional supervised learning. In the ERM framework, a learning algorithm is given a dataset

$$\mathcal{D} = \{(x_i, y_i)\}_{i=1}^{n},$$

where each instance $x_i$ is paired with a fully observed ground-truth label $y_i$. The goal is to learn a hypothesis $h \in \mathcal{H}$ that minimizes the empirical risk,

$$\hat{R}(h) = \frac{1}{n} \sum_{i=1}^{n} \ell(y_i, h(x_i)),$$

with $\ell : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}_+$ quantifying the discrepancy between the predicted and true labels. This framework assumes that the training data is both complete and unbiased—a *full-information* setting. In contrast, many real-world systems rely on data collected through biased logging policies, which may cause ERM to inadvertently learn these biases instead of the true underlying relationships.

Contextual bandits introduce a setting where, at each round, an algorithm observes a context $x \in \mathcal{X}$ and must choose an action $a \in \mathcal{A}$ from a (finite or continuous) action set. Once an action is taken, the algorithm receives a reward $r \in \mathbb{R}$, but it observes feedback *only for the chosen action.* Formally, at each round $t$ the learning process is as follows:

(1) Receive a context $x_t \in \mathcal{X}$.
(2) Choose an action $a_t \in \mathcal{A}$ according to a policy $h : \mathcal{X} \to \mathcal{A}$.
(3) Observe the reward $r_t = r(x_t, a_t)$ associated with the chosen action.

The goal is to develop a policy that maximizes cumulative reward (or equivalently minimizes regret relative to the best policy). Two challenges arise from this formulation:

- **Exploration vs. Exploitation:** Since the algorithm sees only the reward of the chosen action, it must balance exploiting actions with high estimated rewards and exploring less-certain actions to acquire additional information.
- **Partial Feedback:** Unlike full-information settings, the learner receives only partial feedback, making standard ERM techniques directly inapplicable.

A common exploration strategy in this setting is the $\epsilon$-greedy policy. Under this approach, the learner selects the action with the highest estimated reward with probability $1 - \epsilon$, and chooses an action *uniformly at random* with probability $\epsilon$. Formally, for a given

*Both authors contributed equally to this research.


*OARS '25, August 3-7, 2025, Toronto, Canada*

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM
https://doi.org/10.1145/nnnnnnn.nnnnnnn

context $x$ and action set $\mathcal{A}$, the $\epsilon$-greedy policy $h_\epsilon$ is defined as

$$h_\epsilon(x) = \begin{cases} \arg\max_{a \in \mathcal{A}} R(x, a), & \text{with probability } 1 - \epsilon, \\ \text{a random action from } \mathcal{A}, & \text{with probability } \epsilon, \end{cases}$$

where $R(x, a)$ estimates the expected reward for taking action $a$ in context $x$.

The $\epsilon$-greedy exploration strategy has no specific modeling assumptions which means it can easily leverage existing ML infrastructure: a la regression models (e.g., gradient boosting or NNs). This makes it very popular in industry [2, 7, 11]), however, its linear regret due to the reliance on *uniform random exploration* is an expensive trade off.

Of course, contextual bandits literature has made progress against the regret issue for years and many solutions have been proposed however, they all come with limitations. For instance, early methods like Upper Confidence Bound (UCB/1) and Thompson Sampling enjoyed non-uniform exploration, but either lack adequate contextual integration or require heavy computational overhead and rigid modeling assumptions. LinUCB [6] improved upon this by incorporating linear models with analytical uncertainty bounds, though the linear model introduces context, its linearity limits its expressiveness (modeling assumptions). Subsequent approaches, such as NeuralUCB [10] and NNLinUCB [8], relax the linearity assumption but incur high computational costs or perform only "shadow" exploration on neural network features. In contrast, methods like EE-Net [9] employ multiple neural networks to decouple exploitation from exploration, achieving better performance in some settings but still have rigid modeling assumptions.

A recent line of work further advances exploration by introducing *regression oracles* [3, 4]. These methods transform supervised learning predictions into online policies, thereby harnessing the power of general-purpose supervised algorithms (e.g., neural networks or boosting) to address the exploration–exploitation tradeoff.

In this work, we build on these ideas by proposing a methodology that achieves non-uniform exploration in contextual bandits with standard supervised models.

***Our contributions are novel applications of regression oracles that enable***:

- Non-random uniform exploration without modeling assumptions
- Context between arms
- Logged bandit feedback

## 2 Methodology

Given the introduction to the setting and constraints at Adyen we hope to clearly define our problem space and layout our approach to a solution.

### 2.1 Problem Definition

The objective is to maximize cumulative reward over time by learning an optimal policy $\pi$ that maps contexts to actions.

| Algorithm | Contextual | Non-uniform | Assumptions* |
|---|---|---|---|
| UCB1 | × | ✓ | × |
| TS | × | ✓ | × |
| LinUCB | ✓ | ✓ | × |
| NNLinUCB | ✓ | ✓ | × |
| NeuralUCB | ✓ | ✓ | × |
| Epsilon-greedy | ✓ | × | ✓ |
| SquareCB | ✓ | ✓ | ✓ |

**Table 1: Comparison of contextual bandit algorithms across key dimensions. *Modeling Assumptions. One can observe the tradeoff between low cost exploration and model assumptions wrt context**

Currently, $\pi$ is $\epsilon$-greedy, using a boosting model during exploitation. However, uniform random exploration in $\epsilon$-greedy yields linear regret; reducing this regret remains an important open problem with substantial practical value.

Motivated from the introduction and the problem, we have 3 requirements:

- Usage of logged bandit feedback
- Contextual with no modeling assumptions
- Non-uniform random exploration

How can we perform non-uniform exploration while leveraging logged bandit feedback? Regression oracles provide a promising direction by connecting supervised learning techniques with contextual bandit algorithms.

### 2.2 Regression Oracles

Regression oracles are black box functions that make real-valued predictions for rewards $\hat{r}_t \in R$ based on context-action pairs $(x_t, a_t)$. After each prediction, the oracle receives the actual reward and updates its internal model $r_t$.

$$f(x, a) = \mathbb{E}[r \mid x, a],$$

the expected reward for action $a$ in context $x$. The regression oracle selects actions using the induced policy:

$$\pi_f(x) = \arg\max_{a \in \mathcal{A}} f(x, a),$$

where $f \in \mathcal{F}$ is the function learned by the regression oracle to approximate the true reward function.

*2.2.1 SquareCB.* Some regression oracles, like SquareCB [4] are able to explore *non-randomly* which is the key for industrial applications. At each step, it computes predictions for each action, selects the best, assigns probabilities inversely proportional to the gap from the best, *samples* an action, and updates the oracle. This routine is outlined in Algorithm 1, inspired by Krishnamurthy [5]. The key is the probability selection scheme from Abe & Long [1], is visualized in Figure 1. The key advantage is that it can leverage supervised learning models, such as boosting, for the oracle, which makes it practical for industrial applications.

### 2.3 Assumptions

The implementation of regression oracles in large-scale industrial settings, such as at Adyen, introduces unique considerations and

**Algorithm 1** SquareCB: A Regression Oracle-Based Non-Uniform Exploration Algorithm. The algorithm computes action probabilities based on the gap between the best predicted reward $\hat{r}_{t,b_t}$ and each alternative action $\hat{r}_{t,a}$, samples an action accordingly, and periodically retrains the offline regression oracle using a batch of logged data.

1: **Parameters:**
Learning rate $\gamma > 0$, exploration parameter $\mu > 0$.
Offline regression oracle *SqAlg* trained over a window of $L$ days.
2: **for** $t = 1, \ldots, T$ **do**
3: Receive context $x_t$.
4: For each action $a \in A$, compute

$$\hat{r}_{t,a} = \hat{r}_t(x_t, a).$$

5: Let $b_t = \text{argmax}_{a \in A} \hat{r}_{t,a}$.
6: For each $a \neq b_t$, define

$$p_{t,a} = \frac{1}{\mu + \gamma \left(\hat{r}_{t,b_t} - \hat{r}_{t,a}\right)},$$

and set

$$p_{t,b_t} = 1 - \sum_{a \neq b_t} p_{t,a}.$$

7: Sample $a_t \sim p_t$ and observe reward $r_t \sim R(x_t, a_t)$.
8: **end for**
9: Collect $T = L$ samples of $((x_t, a_t), r_t)$ and retrain *SqAlg*.

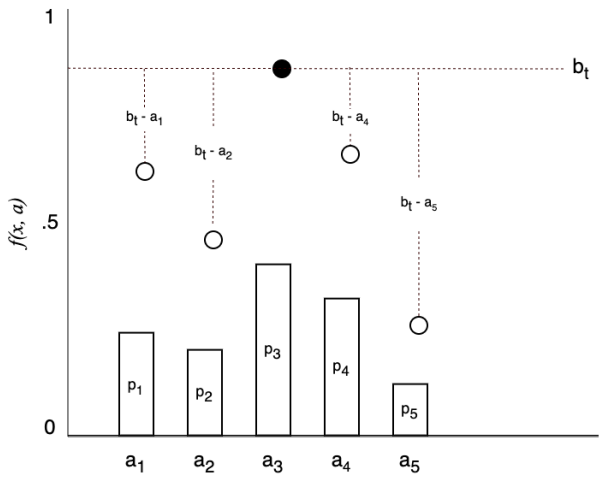


**Figure 1: SquareCB Regression Oracle. For $a \in \mathcal{A}$, let $r_a = r(x, a)$ (e.g., using XGBoost for prediction), then build an exploration probability distribution $p_t$ based on differences $b_t$.**

challenges. Below, we outline the key assumptions and how they adapt to our setting.

*2.3.1 Delayed Feedback (Offline Oracles).* A key assumption underlying regression oracles is the availability of *online oracles* that can update incrementally online. However, in many industrial environments—including Adyen—maintaining a system that supports online learning is impractical due to modeling or system constraints (eg updating a tree model incrementally [2]).

Instead, we employ a batch-offline system, where model updates occur in discrete batches rather than in real time. This approach strikes a necessary balance between performance and infrastructural efficiency. Nevertheless, it represents a departure from the theoretical underpinnings, as the model's parameters are updated in a delayed fashion rather than online. This could be viewed in paradigm of *delayed feedback* from reinforcement learning literature.

*2.3.2 Regression.* The SquareCB framework assumes that the functions in the value function class $F$ are optimized using square-loss regression – as implied by the name. However, given that our rewards are binary, we employ a binary cross-entropy loss function for our regression oracle. This adaptation aligns the loss function with the binary nature of our reward signals while preserving the oracle's ability to approximate expected rewards.

*2.3.3 Realizability.* Regression oracles operate under a strong realizability assumption, which requires the value function $r$ chosen by the oracle to closely approximate the true reward distribution. Specifically, for each trial (or in our case, transaction) $t$, there exists a function $r^* \in R$ such that:

$$r^*(x, a) = \mathbb{E}[l_t(a) \mid x_t = x],$$

where $r^*$ represents the true underlying reward function. The oracle's goal is to select a function $r$ that best approximates $r^*$. In practice, however, this assumption is often partially violated because the true reward distribution is complex and hidden. As such, even the best available value function $r$ may only approximate the true mean reward, introducing potential gaps between theoretical assumptions and practical implementations.

*2.3.4 Dynamic Action Space.* Traditional contextual bandit frameworks assume a static action space $\mathcal{A}$, where the set of available actions remains fixed. While this is true in our setting, we introduce a *dynamic action space*, where action space (eligible actions) can vary across payment contexts. This dynamic nature increases the complexity for both the *oracle* and the *policy*, as they must account for additional variability when learning and optimizing over the action space.

For each context $x_t$, the set of available actions $\mathcal{A}_t$ is determined dynamically using a two-step process:

(1) **Rule-based Filtering:** Remove actions that conflict with predefined constraints, such as:
- Merchant payment service provider contracts,
- Card network regulations (e.g., country restrictions imposed by Visa/MasterCard),
- Transaction amount thresholds.

(2) **Risk-based Pruning:** Exclude actions considered high-risk based on:
- Historical fraud rates for specific merchant, country, and currency combinations,
- Real-time monitoring of authorization rates.

This process typically produced $|\mathcal{A}_t| \in [2, 15]$ actions per transaction, with median 4 actions.

## 3 Experiments

In this section, we empirically evaluate our proposed regression oracle approach—implemented via SquareCB—in the context of payment processing at Adyen. We compare its performance against a standard $\epsilon$-greedy baseline under realistic industrial settings. Our experiments focus on key metrics such as overall performance, effective exploration, and the impact on subsequent model training.

### 3.1 Experimental Setup

Our evaluation framework consists of three main components:

(1) **Data Collection:** We utilize a logged bandit feedback dataset extracted from Adyen's production system. The dataset consists of 180 million samples collected over a 30-day period. Each sample comprises 56 contextual features (e.g., transaction amount, country, device type), an action label corresponding to the applied intervention, and a binary reward indicating whether the payment was authorized. Table 2 provides an overview of the dataset attributes.
(2) **Offline Oracle Training:** We train a regression oracle using the Empirical Risk Minimization (ERM) framework. In our implementation, we employ a boosting classifier with binary cross-entropy loss to approximate the expected reward for each context-action pair (last line of Algorithm 1.)
(3) **Policy Deployment and A/B Testing:** The trained oracle is integrated into the SquareCB policy to generate non-uniform exploration probabilities (see Algorithm 1). For comparison, we deploy a traditional $\epsilon$-greedy policy at varying exploration rates (1%, 4%, and 6%) as baselines. Both policies are A/B tested by routing 5% of the live traffic to each variant over a four-week period.

| Attribute | Description |
|---|---|
| Samples | 180M |
| Features | 56 |
| Target | Authorized (binary) |
| Imbalance | 90% positive |

**Table 2: Dataset Description for Offline Oracle Training**

### 3.2 Evaluation Metrics

We evaluate performance based on the following metrics:

- **Cumulative Reward/Uplift:** The primary metric is the cumulative number of authorized transactions. We report percentage uplifts relative to the $\epsilon$-greedy baseline.
- **Effective Exploration Rate:** Defined as the proportion of rounds in which a non-greedy (i.e., non-optimal) action is chosen. This metric helps us assess whether the non-uniform exploration of SquareCB improves the diversity of training data compared to uniform random exploration.
- **Action Diversity:** We quantify diversity using Lorenz curves and the traffic-weighted Gini coefficient, thereby measuring how evenly different interventions are selected across varying context groups.

The experiments involved A/B testing models with varying learning rates against $\epsilon$-greedy baselines. The baseline uniform random $\epsilon$-greedy policies were set at three levels of exploration: 1%, 4%, and 6%. Each variant received 5% of the total traffic to ensure a fair comparison. We conducted experiments over a four-week period.

Point estimates of success probabilities were calculated, along with 75% and 95% confidence intervals to measure the robustness of each policy.

| Experiment | Learning Rate | % Total Traffic | Duration |
|---|---|---|---|
| Oracle 1 | 1% | 5% | 4 weeks |
| Oracle 2 | 4% | 5% | 1 week |
| Oracle 3 | 6% | 5% | 1 week |
| $\epsilon$-greedy | N/A | 5% | 4 weeks |

**Table 3: A/B tests with different learning rates and traffic splits.**

As noted in the Introduction, we did not benchmark against UCB-style or Thompson-Sampling algorithms because they lack support for both contextual information and non-uniform exploration—two requirements central to our setting. NeuralUCB was likewise omitted due to its substantial computational overhead and additional neural-network modeling assumptions. A summary of these differences appears in Table 1.

Payments come into Adyen - if they pass various risk checks they will be sent to our optimization system where 1 or many interventions will be applied (based on the oracle) and sent to the environment for the reward.

## 4 Results

First we'll look at overall performance and then break it down by exploration and exploitation segments.

### 4.1 Overall Performance

Figure 2 shows that the best-performing SquareCB variant achieved a +0.1% uplift over the baseline $\epsilon$-greedy policy within the four-week test period. This improvement translates to an estimated 9 million incremental authorized transactions per year. The confidence intervals (95%) indicate that the improvement is statistically significant. The intuition behind the performance gains of regression oracles is attributed to the *reduced regret from non-uniform random exploration* – the premise of this whole line of research.

To confirm our intuition, we measured our effective/actual exploration rates for $\epsilon$-greedy and SquareCB policies respectively in Table 4 with the hypothesis that regression oracles would explore *less* but *more efficiently*. However, we observed surprising results: The 1% $\epsilon$-greedy policy was exploring only 0.7% of the time compared to 1.2% for even the most exploitative SquareCB 50K variant. These counterintuitive results are explored more in discussion section 4.3 .

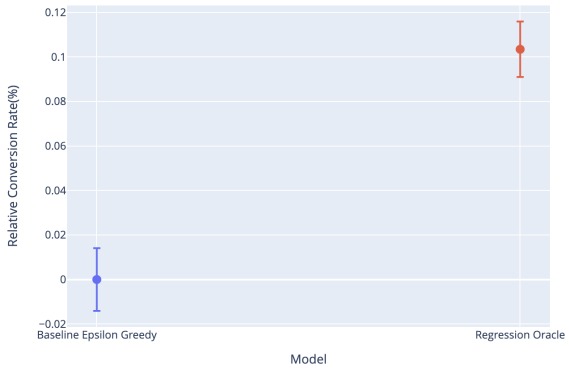


**Figure 2: 95% confidence intervals comparing the performance of SquareCB and $\epsilon$-greedy policies.**

Given these results, we wanted to compare $\epsilon$-greedy and SquareCB while controlling for exploration rates. So, in the spirit of fairness, Figure 3 illustrates the performance of SquareCB across different learning rates against the $\epsilon$-greedy baselines.


**Figure 3: Comparison of $\epsilon$-greedy and SquareCB variants across various exploration rates. Effective exploration rates are in table 4. e-greedy 6% is 3.45% and SquareCB 10k is 3.6% which are essentially the same rate of exploration and as you can see SquareCB variant outperforms. It should be highlighted that for equal exploration rates (3.5%) the regression oracle variant suffers much less regret.**

Interestingly enough, controlling for exploration rate, the SquareCB policy (square 10k) always outperforms the baseline (e-greedy 6%)

## 4.2 Exploration vs. Exploitation Trade-off

Given that we wish to see the benefits of non-random exploration policy, we compare performance on both parts of the traffic: exploration and exploitation. The hypothesis is that we should see marked uplift for the exploration traffic.

Focusing on the exploration traffic, when the non-optimal action was taken, the SquareCB policy shows a significant **+11%** improvement in performance, as it selects actions from a *non-uniform* random distribution via the SqAlg in Algorithm 1. Figure 4.

Now, looking just at the *exploitation* traffic, the results are counter-intuitive as we observed a slight decrease in exploitation performance, with a reduction of **0.33%** compared to the baseline policy (Figure 5).

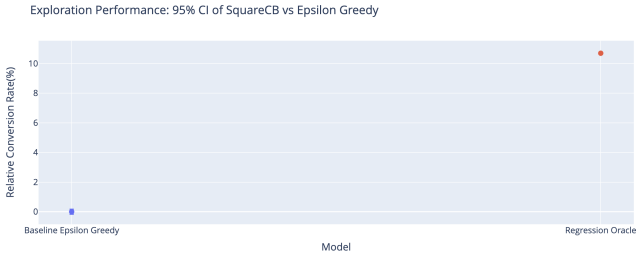


**Figure 4: Exploration performance (non-optimal action rounds) comparing SquareCB and $\epsilon$-greedy policies.**

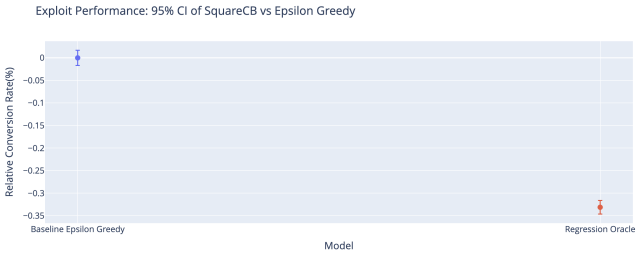


**Figure 5: Exploitation performance (optimal action rounds) for SquareCB and $\epsilon$-greedy policies.**

This trade-off is due to two main factors:

(1) **Partial Realizability**: The supervised classification algorithm imperfectly models the reward distribution, partially violating the realizability criteria. This highlights the importance of using high-quality classifiers, as a suboptimal classifier can increase regret, as noted in the original theoretical framework.
(2) **Action Probability Distribution**: Exploration and exploitation are no longer entirely random. Traffic where multiple actions have closely clustered probabilities tends to be explored more often, while traffic with a single clear "best" action (with a high probability gap between it and the others) is exploited. In our setup, this often corresponds to scenarios with larger action spaces ($|A| > 2$), where performance expectations are naturally lower due to increased variability.

Despite the slight loss in exploitation, the gains in exploration far outweigh this trade-off, *resulting in an overall improvement in policy performance* (Figure 2).

In addition to the strong performance we found some interesting insights as we analyzed the experimental results more closely.

## 4.3 Exploration Across Dynamic Action Spaces

In dynamic action spaces, adequate exploration across varying action space sizes is challenging and nuanced. $\epsilon$-greedy policies are invariant to action space and explore uniformly regardless of size which results in failing to address the data sparsity in larger spaces or *over-exploiting in small spaces*. In contrast, regression oracles like SquareCB adapts their exploration strategies to the action space, focusing more on larger action spaces where data scarcity is more pronounced. This behavior is visualized in Figure 6, which shows

how SquareCB allocates exploration more effectively than $\epsilon$-greedy across different action space sizes. Uniform random exploration in

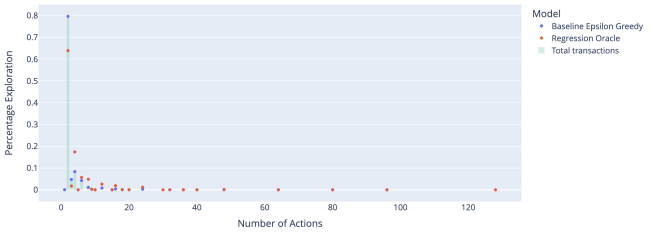


**Figure 6: Exploration rates across action space sizes for $\epsilon$-greedy and SquareCB policies. One can observe that for action space sizes greater than 3 the SquareCB policy starts to explore more frequently.**

the small-action-space regime is the cause of a phenomenon we're calling *effective exploration*. The *effective exploration* rate—defined as the percentage of instances where the action chosen was not the optimal action—was often lower than the nominal exploration rate ($\epsilon$) in $\epsilon$-greedy policies. This discrepancy arises because there is a $1/N_A$ chance of selecting the best action randomly, where $N_A$ is the size of the action space. For example, if $N_A = 2$ then effective exploration of a 1% $\epsilon$-greedy policy could be much lower than the expected 1% due to the high probability of selecting the optimal action during exploration.

| Variant | Effective Exploration % |
|---|---|
| SquareCB LR1k | 13.0% |
| SquareCB LR 4K | 6.5% |
| SquareCB LR 7K | 4.6% |
| SquareCB LR 10K | 3.6% |
| SquareCB LR 50K | 1.2% |
| $\epsilon$-Greedy (6%) | 3.41% |
| $\epsilon$-Greedy (4%) | 2.3% |
| $\epsilon$-Greedy (1%) | 0.7% |

**Table 4: Effective Exploration Rates for SquareCB and $\epsilon$-Greedy Policies. Counterintuitively, the $\epsilon$-greedy policies are exploring less than than expected due to the dynamics of the small action space regime.**

Given the dynamic action space in our setup, understanding effective exploration rate was crucial for assessing its contribution to training data diversity. Table 4 summarizes the effective exploration rates for SquareCB and $\epsilon$-greedy variants.

## 4.4 Action Diversity

As we have verified the expected exploration improvements promised by regression oracles, now we'd like to look at a common problem with polices in general: popularity bias, or framed in another perspective: action diversity.

Maintaining action diversity is critical for training robust next-generation policies. In our setup, dynamic action spaces inherently introduce biases, as certain actions are only available in specific contexts. This context-based action restriction creates an imbalance in the action distribution.

To evaluate action diversity, we measured Lorenz curves and Gini coefficients across various context groups. Figures 7 and 8 show improvements in action diversity for SquareCB compared to $\epsilon$-greedy policies. *SquareCB effectively diversified action selection in larger action spaces while maintaining performance in simpler contexts.*

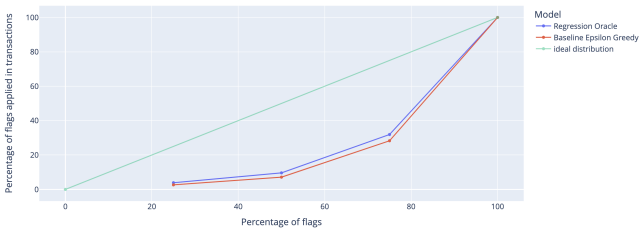


**Figure 7: Lorenz Curve for contexts with 4 possible actions. (The word "flags" in the chart can be interpreted as "actions".)**

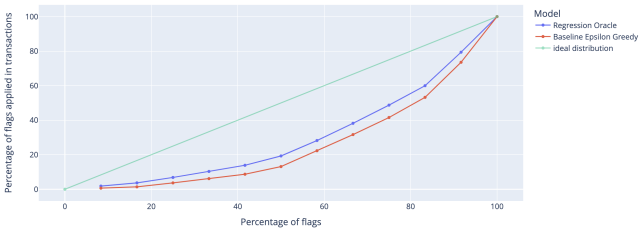


**Figure 8: Lorenz Curve for contexts with 12 possible actions. (The word "flags" in the chart can be interpreted as "actions".)**

Interestingly, in simpler contexts with only two possible actions, SquareCB and $\epsilon$-greedy policies showed nearly identical Lorenz curves (Figure 9). *This indicates that SquareCB allocates exploration where it is needed most, leaving low-dimensional action spaces largely unaffected.*

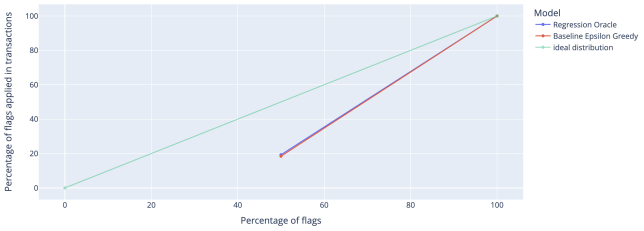


**Figure 9: Lorenz Curve for contexts with 2 possible actions. (The word "flags" in the chart can be interpreted as "actions".)**

Table 5 provides the traffic-weighted Gini coefficients for each policy, showing that *SquareCB reduced action inequality compared to the baseline $\epsilon$-greedy policy.*

These results indicate that SquareCB is better able to diversify action selection, especially in contexts with larger action sets. Nonetheless, the skewed exploration may lead to an imbalance in training data (fewer negative labels), which is discussed in Section 4.3.

| Model | Gini Coefficient |
|---|---|
| $\epsilon$-Greedy | 0.39742 |
| SquareCB | 0.39265 |

**Table 5: Traffic-weighted Gini coefficients for action diversity (lower values indicate better diversity).**

## 4.5 Class Imbalance

A key finding of our study is that while the enhanced exploration of SquareCB improves immediate policy performance, it also exacerbates class imbalance in the logged data used for training future models (oracles). For instance, bandit feedback from the baseline $\epsilon$-greedy policies exhibits a class imbalance of approximately 87.4% positive rewards. Under the SquareCB regime, this imbalance increases to 93.5%, which correlates with a **0.2%** performance regression (see Figure 10).

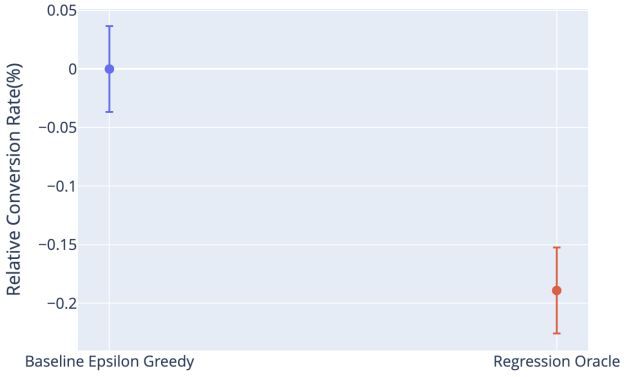


**Figure 10: Comparison of two $\epsilon$-greedy policies trained on data from uniform and non-uniform (SquareCB) exploration. The performance drop in the SquareCB-based training data is attributed to an increased imbalance between positive and negative labels.**

As SquareCB improves the success rate of exploration actions, the frequency of negative outcomes (i.e., failed actions) decreases. These negative labels are essential for supervised models to learn robust decision boundaries between good and poor actions. Their reduction biases the training process, ultimately degrading the generalization capability of subsequent models. In other words, as the proportion of positive rewards increases, the resulting imbalance in the training data inadvertently compromises the quality of future models. This feedback loop highlights a fundamental challenge when applying ERM to logged bandit feedback. As current policies improve, the quality of training data for subsequent models deteriorates, resulting in weaker supervised models that eventually harm future policies. *This paradox highlights a fundamental flaw in using ERM for bandit feedback.*

In addition, since the exploration is non-random, we can infer that the exploration data tend to be concentrated in the region of the higher action space size in this case of dynamic action spaces. Thus we lose the uniformity in data we have in epsilon greedy - of context and action combinations, which adds to the impact on future iterations of models trained on the logged feedback, explained further in the next section.

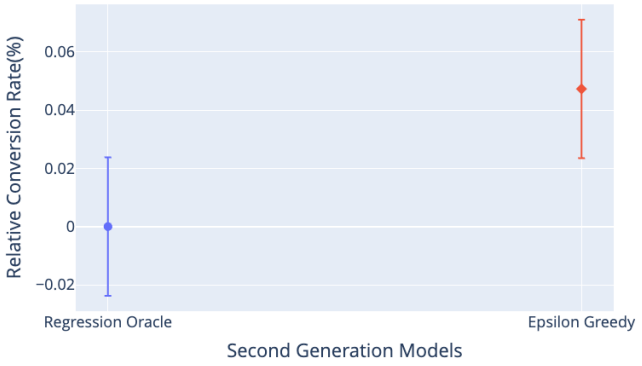


**Figure 11: Performance of two second generation models resulting from Epsilon Greedy policy and Regression Oracle. Despite Exploration sample being a significantly smaller part of the training data ($N_{exploitation} >> N_{exploration}$ ) and exploitation being the majority in the training data of each, the Regression Oracle's second generation regresses in performance compared to Epsilon Greedy.**

*4.5.1 Second Generation of Models.* As shown in Figure 11, the second generation of models is affected by the sampling skew introduced by the non-greedy selection rules of SquareCB. The *$\epsilon$-greedy* policy contributes a small share of uniformly sampled data, enriching the next-generation training set with diverse triplets (context, action, reward). While SquareCB increases action diversity by occasionally selecting the second- or third-best action, it still cannot match $\epsilon$-greedy's uniformly distributed (context, action) pairs and the resulting more balanced reward distribution that benefits subsequent training iterations.

# 5 Discussion

## 5.1 ERM for Bandit Feedback

In our setting at Adyen, ERM leverages logged bandit feedback $(x, a, r)$, where $r \in [0, 1]$, to learn the conditional distribution $P(r \mid x, a)$. The goal is to predict the expected reward $r$ for a given context-action pair $(x, a)$, enabling the policy to select actions that maximize expected rewards.

Empirical Risk Minimization (ERM) performs effectively when provided with ample and diverse training data. However, in the

context of logged bandit feedback, only a limited subset of context-action pairs is observed, as *counterfactual outcomes for unchosen actions remain unknown*. This scenario, commonly referred to as *partial feedback*, introduces significant imbalances in the training data. Specifically, the logged bandit data tends to disproportionately represent actions with higher predicted probabilities of success, while actions with lower probabilities are underexplored. This imbalance distorts the learned conditional distribution $P(r \mid x, a)$, as the dataset lacks sufficient negative labels (representing low-reward actions). Consequently, the model struggles to effectively distinguish between high- and low-reward actions, impairing its ability to generalize to unseen or underexplored actions. This challenge arises because ERM inherently assumes a *full-data setting*, an assumption that is fundamentally violated in the bandit feedback paradigm.

## 6 Future Work

**Counterfactual Risk Minimization (CRM)** provides an effective foundation for addressing the challenges of partial feedback in logged bandit data and represents the focus of our next line of research. By leveraging inverse propensity scoring (IPS) to reweight observed data, CRM addresses the inherent imbalances in logged feedback by assigning greater weight to underrepresented actions and mitigating the over-representation of high-reward actions, enabling more accurate reward estimation. Furthermore, CRM directly optimizes a counterfactual objective and employs variance reduction techniques, such as self-normalized estimators or clipping, to enhance stability and efficiency. Regularization methods further prevent overfitting to the reweighted data, ensuring better generalization to unseen or under-explored actions. By aligning the training objective with the counterfactual nature of logged bandit feedback, CRM offers a promising pathway to overcoming the limitations of ERM, paving the way for the development of robust, generalizable policies in settings with incomplete and biased feedback—an avenue we aim to explore in future work.

## 7 Conclusion

In this work, we addressed the challenge of leveraging logged bandit feedback to learn effective policies offline, a critical need in industry where untested policies cannot be fielded. While traditional contextual bandit methods rely on uniform random exploration or on-policy learning, we demonstrated the benefits of adopting regression oracles for non-uniform exploration, significantly reducing regret and improving performance.

## References


[1] Naoki Abe and Philip M. Long. Associative reinforcement learning using linear probabilistic concepts. In *Proceedings of the Sixteenth International Conference on Machine Learning*, pages 3–11. Morgan Kaufmann Publishers Inc., 1999.

[2] Alex Egg. Online learning for recommendations at grubhub. In *Proceedings of the 15th ACM Conference on Recommender Systems*, RecSys '21, page 569–571, New York, NY, USA, 2021. Association for Computing Machinery.

[3] Dylan Foster, Alekh Agarwal, Miroslav Dudik, Haipeng Luo, and Robert Schapire. Practical contextual bandits with regression oracles. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 1539–1548. PMLR, 10–15 Jul 2018.

[4] Dylan Foster and Alexander Rakhlin. Beyond UCB: Optimal and efficient contextual bandits with regression oracles. In Hal Daumé III and Aarti Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 3199–3210. PMLR, 13–18 Jul 2020.

[5] Akshay Krishnamurthy. Lecture 4: Contextual bandits, cs, umass amherst, February 2022.

[6] Lihong Li, Wei Chu, John Langford, and Robert E. Schapire. A contextual-bandit approach to personalized news article recommendation. In *Proceedings of the 19th International Conference on World Wide Web*, WWW '10, page 661–670, New York, NY, USA, 2010. Association for Computing Machinery.

[7] Rodel van Rooijen. Optimizing payment conversion rates with contextual multi-armed bandits, November 2020.

[8] Pan Xu, Zheng Wen, Handong Zhao, and Quanquan Gu. Neural contextual bandits with deep representation and shallow exploration. In *International Conference on Learning Representations*, 2022.

[9] Pan Xu, Zheng Wen, Handong Zhao, and Quanquan Gu. Neural contextual bandits with deep representation and shallow exploration. In *International Conference on Learning Representations*, 2022.

[10] Dongruo Zhou, Lihong Li, and Quanquan Gu. Neural contextual bandits with UCB-based exploration. In Hal Daumé III and Aarti Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 11492–11502. PMLR, 13–18 Jul 2020.

[11] Yunzhang Zhu, Gang Wang, Junli Yang, Dakan Wang, Jun Yan, and Zheng Chen. Revenue optimization with relevance constraint in sponsored search. In *KDD Workshop on Data Mining and Audience Intelligence for Advertising*, 2009.

# A Appendix

## A.1 Setting the Exploration Percentage

Upon tuning the learning rate and finding the initial selection of variants we wanted to experiment with, we also found the corresponding exploration percentages we would gain if we used these learning rates for the SquareCB policy.

| Learning Rate | Exploration Percentage |
|---|---|
| 10K | 3.7% |
| 7K | 4.6% |
| 4K | 6.5% |
| 1K | 13% |

**Table 6: Exploration Percentage Guaranteed by various learning rates for SquareCB**

## A.2 Learning Rate Tuning

The SquareCB policy, has a tunable parameter $\gamma$, the learning rate. Essentially this controls the amount of emphasis we wish to apply in the probability distribution calculation of squareCB:

$$p_t(a) = \frac{1}{A + \gamma(\hat{y}_{t,b_t} - \hat{y}_{t,a_t})} \tag{1}$$

Where $\hat{y}_{t,b_t}$ is the predicted probability for the best action and $\hat{y}_{t,a_t}$ is the predicted probability of every other action. Therefore, $\gamma$ controls the importance given to the distance of an action from the best action [4]. Therefore we tune the probability of selecting an action keeping in mind: the lower the learning rate, the closer the probability of selecting a certain action moves to uniform selection, the higher the learning rate, the more this probability depends on the distance from the predicted probability of success of the best action.

An initial selection of models for experimentation was done through tuning of the learning rate - by observing the distribution of the "predicted probability of success" of the action selected, by the supervised classification model with a range of learning rates. Given the large volume of transactions at Adyen, the goal was to ensure that the predicted probability of success distribution of the SquareCB Policy was as close to a 100% greedy policy. For reference, this is how the distribution of probability of success of selection action looked for a learning rate of 1000:

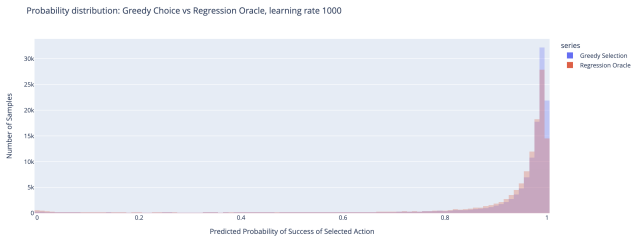


**Figure 12: Probability distribution of selection actions $a$ for a purely greedy/exploitation policy vs a square-cb policy.**

And this is how it looked for a learning rate of 10000:

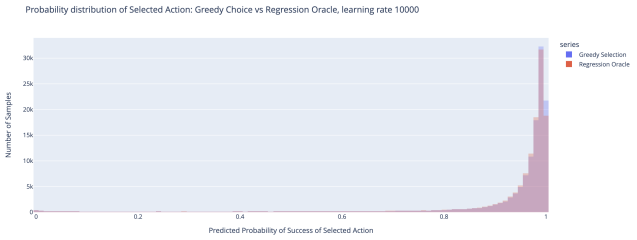


**Figure 13: Selected Action Probability Distribution with learning rate 10000**

The selection of learning rate was done based on this probability distribution, exploration percentage and performance in A/B testing over a period of experimentation.

## A.3 Hyperparams

*A.3.1 Model Configuration.* We implemented our regression oracle using XGBoost with the following key hyperparameters:

**Table 7: XGBoost Hyperparameters**

| Parameter | Value | Description |
|---|---|---|
| learning_rate | 0.1 | Step size shrinkage |
| max_depth | 10 | Maximum tree depth |
| subsample | 0.8 | Fraction of samples per tree |
| colsample_bytree | 0.8 | Fraction of features per tree |
| n_estimators | 1000 | Number of boosting rounds |
| objective | binary:logistic | Loss function |

Training required approximately **4 hours per model** on 32 CPU cores, with periodic retraining every 7 days to maintain freshness.

*A.3.2 Feature Engineering.* Our feature pipeline included:

- **Contextual features**: Transaction amount, currency, country, merchant category, device type
- **Temporal features**: Rolling 7-day authorization rates, time since last transaction
- **Embedding-based features**: Merchant representations learned via historical transaction patterns
- **Normalization**: Min-max scaling for monetary amounts
- **Imputation**: Median values for missing numeric features, special category for missing categoricals