MARS-Gym: Offline Reinforcement Learning for Recommender Systems in Marketplaces

Marlesson R. O. Santana[*]
Deep Learning Brazil
Federal University of Goiás
Goiânia, Brazil
marlessonsa@gmail.com

Luckeciano C. Melo[*]
Deep Learning Brazil
Goiânia, Brazil
luckeciano@gmail.com

Fernando H. F. Camargo[*][†]
Deep Learning Brazil
Federal University of Goiás
Goiânia, Brazil

Bruno Brandão
Deep Learning Brazil
Federal University of Goiás
Goiânia, Brazil
brunobrandao1523@gmail.com

Anderson Soares
Deep Learning Brazil
Federal University of Goiás
Goiânia, Brazil
anderson@inf.ufg.br

Renan M. Oliveira
iFood Research
Campinas, Brazil
renan.oliveira@ifood.com.br

Sandor Caetano
iFood Research
Campinas, Brazil
sandor.caetano@ifood.com.br

Abstract

Recommender Systems are especially challenging for marketplaces since they must maximize user satisfaction while maintaining the healthiness and fairness of such ecosystems. In this context, we observed a lack of resources to design, train, and evaluate agents that learn from interaction logs of these production systems, especially in the offline setting. For this matter, we propose the **MA**rketplace **R**ecommender **S**ystems Gym (*MARS-Gym*), an open-source framework to empower researchers and engineers to quickly build and evaluate Reinforcement Learning agents for recommendations in marketplaces. *MARS-Gym* addresses the whole development pipeline: data processing, model design and offline training, and multi-sided evaluation. We also provide the implementation of a diverse set of baseline agents, with a metrics-driven analysis of them in the Trivago marketplace dataset, to illustrate how to conduct a holistic assessment using the available metrics of recommendation, off-policy estimation, and fairness. With *MARS-Gym*, we expect to bridge the gap between academic research and production systems, as well as to facilitate the design of new algorithms and applications.

1 Introduction

Recommender Systems (RS) are essential tools to offering a high-quality experience to users on online platforms. These systems become even more critical for marketplaces. In this specific scenario, recommenders should also consider all partners, rather than assume a user-centric perspective. This requirement is critical to the healthiness and fairness of such ecosystems. Despite the recent progress in building successful recommendation models to real-world applications [5, 29], and, in special,

[*] denotes equal contribution.

[†] Contact: fernando.camargo.ai@gmail.com.

Offline Reinforcement Learning Workshop at Neural Information Processing Systems, 2020.

for marketplaces [41, 30], we observed a lack of resources to experiment models that learn from interaction logs of these ecosystems effectively. To the best of our knowledge, there are no frameworks that help researchers and engineers to quickly design, train, and evaluate Reinforcement Learning (RL) agents to optimize the dynamics of marketplaces, with special focus on real-world applications.

In this work, we present *MARS-Gym* (**MA**rketplace **R**ecommender **S**ystems Gym), a open source framework for modeling, training, and evaluating RL-based recommender systems for marketplaces. Three main components compose the framework. The first one is a highly customizable module where the consumer can ingest and process a massive amount of interaction logs for learning using spark jobs. The second component was designed for training purposes, with counterfactual losses to support offline RL. It holds an extensible module built on top of PyTorch [34] to design learning architectures. It also possesses an OpenAI's Gym [9] environment that ingests the processed dataset to run a multi-agent system that simulates the targeted marketplace.

Finally, the last component is an evaluation module that provides a set of distinct perspectives on the agent's performance. It presents not only traditional recommendation metrics but also off-policy evaluation metrics, to account for the bias induced from the historical data representation of marketplace dynamics. It also provides fairness indicators to analyze the long-term impact of such recommenders in the ecosystem concerning sensitive attributes. This component is powered by a user-friendly interface to facilitate the analysis and comparison between agents.

This paper is organized as follows. In Section 2, we describe related work. Section 3 presents the theoretical background. Section 4 details the framework usage, its internal components, and the evaluation protocol. Section 5 presents the experimental results and analysis for the baselines agents. Finally, Section 6 concludes and shares our thoughts for future work.

2 Related Work

We identified some works which focused on marketplace recommendations. Wang *et al.* [41] proposed a model based on multi-armed bandits and UCB (Upper Confidence Bound) exploration. It was trained via multi-objective optimization to address exploration and diversity of recommendations in a food delivery marketplace. McInerney *et al.* [30], on the other side, formulated the problem as a contextual bandit in an off-policy training scenario to recommend not only items but also an associated explanation in an online music streaming service. For comparison, our work also addresses marketplace recommendation with sequential decision-making formulation; however, we propose a framework and benchmark environment to improve research and development in this subject, rather than new models and applications.

In terms of environments for RS evaluation, we identified works that focused on open-sourcing datasets [6, 21, 19], elaborating competitions [25, 44, 2], and releasing environments or platforms to engage researchers. In the last line of contribution, which is closely related to our work, we highlight *RecoGym* [36] and *PyRecGym* [38] platforms. Like our work, they also created OpenAI's Gym environments for sequential interaction, which enables the development of RL agents in RS. However, those platforms assumed a user-centric perspective where the episodes are users' sessions, and the objective is to maximize their satisfaction. On the other side, our work assumes the perspective of the whole marketplace, simulating several users acting in each episode, aiming to maximize the satisfaction of all sides involved.

In the field of offline RL, we noticed several works in theory and algorithm development [27, 4, 26, 32], benchmarks and datasets [20, 17, 16], and even applications in robotics [11] and advertising engines [8]. In this work, our objective is to provide such techniques to improve modeling and deployment of agents into productions systems in the context of Recommender Systems, addressing some of the main challenges for real-world RL [14]. We implement off-policy metrics and losses based on counterfactual estimation to compare against traditional recommendation metrics and diagnose any source of bias induced by the historical dataset.

Finally, in terms of fairness, there is a large body of recent work that formulates this concept for marketplaces [10, 1] and use it as evaluation protocol or even as optimization objective [31, 35, 41]. Our work presents fairness metrics either to ensure that models account for the satisfaction and visibility for all involved sides and also to diagnose any source of bias in sensitive attributes.

3 Preliminaries

3.1 Reinforcement Learning and Recommender Systems

We formalize a Recommender System by the main aspects of its composition. We call a policy π, the primary means of making decisions. Thus, in the case where a dataset $\mathcal{D}$ is available, the policy used to acquire it is called a collection policy π_c. In order to make recommendations, the policy takes into account $x \in \mathcal{X} : \mathbb{U} \times \mathcal{C}$ composed of the current user $u \in \mathbb{U}$ and contextual information $c \in \mathcal{C}$. Finally, the policy selects an action $a \in \mathcal{A}$ – according to $\pi(a \mid x)$ – as a recommendation to the user.

The RL problem formulation is formalized as a Markov Decision Process (MDP) [39]. In an MDP, an *agent* is whatever form that makes decisions; at a time step t, it receives from the environment *state* $s_t \in \mathcal{S}$ and chooses from a finite set of *actions* $a_t \in \mathcal{A}$. The environment changes, following a dynamics $p : \mathcal{S} \times \mathcal{S} \times \mathcal{A} \to [0, \infty)$, which represents the probability density of the next state. Finally, the environment sends a feedback *reward* $r : \mathcal{S} \times \mathcal{A} \to [0, 1]$. The ultimate objective of an RL agent is to maximize the cumulative reward, i.e., $\max \mathbb{E}[\sum_{t=0}^{T} \gamma^t r(s_t, a_t)]$, where γ is a discount factor to account for delayed rewards.

3.2 Off-Policy Evaluation and Learning

MARS-Gym addresses the process of training a Recommender System as a off-policy, batch learning from partially observed feedback in a historical dataset. This consideration is advantageous because we do not require any additional interaction with production systems, which is often very costly. This advantage is valid for learning and evaluation, which makes iteration cycles faster and helps to detect poor policies before any deployment.

Nevertheless, this is acknowledged as a hard problem, because such data is biased towards the collection policy (predictions favored by the historical algorithm will be over-represented) and is incomplete (feedback for other predictions will not be available) [40]. To address this problem, we need counterfactual estimators [7], in order to estimate how other systems would have performed if they had been recommending items in the place of the collection policy. Several works applied these estimators for both off-policy evaluation [28, 13] and learning [30, 40].

For evaluation, *MARS-Gym* implements three main estimators. The first one is called the Direct Method (DM), which forms an estimate $\hat{\varrho}(x, a)$ of the expected reward conditioned on the context and action. It is simple and has low variance, but it depends on the collection policy data, which is often biased. The second estimator, called Inverse Propensity Score (IPS), forms an estimate of the collection policy π_c and uses Importance Sampling to re-weight rewards generated by it such that they are unbiased estimates of the evaluated policy π_e. Although less prone to induce bias, it has a much larger variance, especially when our estimation of the collection policy is small.

The third estimator is the Doubly Robust (DR) Estimator in (1), which combines both reward and collection policy estimates, using the former as a baseline and the latter for correction. If at least one of them is accurate, then DR is also accurate. We will guide our analysis mainly using this metric.

$$\hat{v}^{\pi}_{DR} = \frac{1}{|\mathcal{D}|} \sum_{(x,a,r) \in \mathcal{D}} \left[\frac{\pi_e(a \mid x)}{\hat{\pi_c}(a \mid x)} (r - \hat{\varrho}(x, a)) + \sum_{a \in \mathcal{A}} \pi_e(a \mid x) \hat{\varrho}(x, a) \right], \tag{1}$$

where $\varrho(x, a) = \mathbb{E}_{(x,a,r) \sim \pi_c}[r | x, a]$.

Finally, for learning, we apply the Counterfactual Risk Minimization (CRM) [40]. This learning principle uses one of the counterfactual estimators to modify the log-likelihood. Instead of evaluation policy, we consider an uniform distribution over actions $\mathcal{U}(a)$, to account for the fact that the data come from the production policy π_c and not from an uniform random experiment [30]. Concretely, the current version of *MARS-Gym* implements CRM via IPS, as shown in (2), where $p_{\boldsymbol{\theta}}$ is a distribution, parameterized by $\boldsymbol{\theta}$, from which we derive the agent policy π.

$$\mathcal{L}(\boldsymbol{\theta}) = -\frac{1}{|\mathcal{D}|} \sum_{(x,a,r) \in \mathcal{D}} \frac{\mathcal{U}(a)}{\hat{\pi_c}(a \mid x)} \log p_{\boldsymbol{\theta}}(r \mid x, a). \tag{2}$$

3.3 Fairness Evaluation

Recommender Systems are very prone to reproduce dataset biases as well as to disregard societal and economic impacts when they solely maximize user satisfaction. They are also capable of *changing* the data distribution, i.e., the way people interact within the production system. Therefore, we need to account the long-term impact of such systems in production by diagnosing sources of unfairness that could impact the entities in the marketplace [15].

We focus on two main sources. First, when the user has sensitive attributes that could potentially cause any form of disadvantage or poorer experience for them in the platform. Second, when partners do not have fair opportunity to be presented to users: partner diversity and visibility are intrinsically related to the healthiness of marketplaces, especially because they are acknowledged to suffer from “superstar economics" [37].

In *MARS-Gym*, we consider the notion of fairness in three perspectives [43]: *disparate treatment*, *impact*, and *mistreatment*. In this paper, we present an analysis focused in the third perspective, for the sake of conciseness. *Disparate mistreatment* arises when the misclassification rates differ for groups of users having different values for a given sensitive attribute. To compute these rates, we need ground-truth actions for each interaction. In the current version of the *MARS-Gym*, we measure it by using true positive rates. Therefore, we need to satisfy (3), where a^* is the ground-truth action:

$$\pi(a = a^* \mid z_i, a^* = a_k) = \pi(a = a^* \mid z_j, a^* = a_k), \forall a_k \in \mathcal{A}, \forall z_i, z_j \in \mathcal{Z}. \tag{3}$$

4 *MARS-Gym* Architecture

In this section, we present how MARS-Gym simulates the marketplaces, as well as the design of internal modules used to model, train, and evaluate RL agents for Recommender Systems. The source code of *MARS-Gym* is available in `https://github.com/deeplearningbrasil/mars-gym`.

4.1 MDP Counterfactual Simulation

MARS-Gym models the marketplace as an MDP. The framework ingests a ground-truth dataset sampled from a collection policy in the real marketplace to perform a counterfactual simulation, i.e., reason about the following question: *"how the policy in evaluation would perform if it were in the place of the collection policy, with the same perspective of the dynamics and reward distribution?"*. *MARS-Gym* generates an OpenAI's Gym environment where the data drives the internal transitions resulted from the interaction within the recommender. The consumer of *MARS-Gym* must provide the ground-truth dataset.

Fig. 1 presents how *MARS-Gym* simulates the dynamics of the marketplace. First of all, the framework filters only successful interactions (e.g., buys or clicks), once they are the available source of the true reward distribution. Then, the gym environment wraps the resultant data. We compose each environment step with one interaction. The provided observation contains the associated user and its metadata, as well as the contextual information from the log. As selected action, the agent should return the recommendation of one partner. The environment also provides additional informative data via a dictionary (for example, a pre-selected list of potential recommendations, in order to narrow the action space).

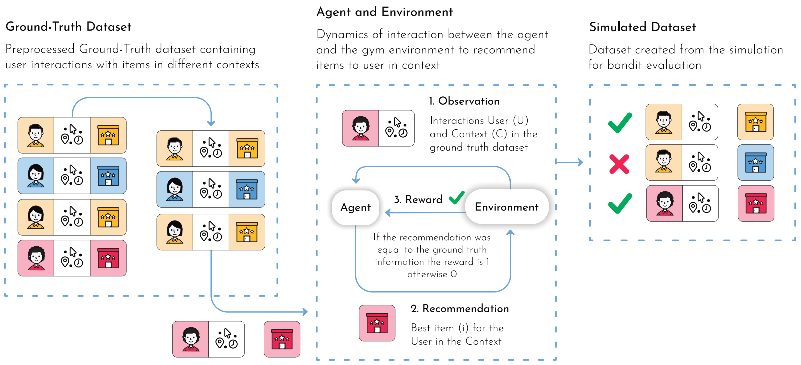

Ground-Truth Dataset
Preprocessed Ground-Truth dataset containing user interactions with items in different contexts
Agent and Environment
Dynamics of interaction between the agent and the gym environment to recommend items to user in context
Simulated Dataset
Dataset created from the simulation for bandit evaluation
1. Observation
Interactions User (U) and Context (C) in the ground truth dataset
Agent
3. Reward
Environment
If the recommendation was equal to the ground truth information the reward is 1 otherwise 0
2. Recommendation
Best item (i) for the User in the Context

Figure 1: Diagram flow of *MARS-Gym*, from dataset ingestion for environment generation to the MDP simulation.

We compute the reward by comparing the agent's selected action and the partner provided by the log. The agent receives a positive reward if they match. Therefore, the agent only discovers the targeted partner in the scenario of a successful recommendation. Otherwise, it should explore the actions to build its knowledge.

The sequence of steps follows the sequence of interactions in the filtered ground-truth dataset. Hence, we maintain the same temporal dynamic. We define an episode as one iteration trough all logs, rather than the user session. This behavior intends to approximate the multi-agent scenario and keep the perspective on the marketplace, not solely in the user. Finally, the interactions between the proposed agent and the environment generate new interaction logs. This simulated data trains the agent and also provides the cumulative reward curve as the first source of evaluation. In the next subsection, we describe the design of *MARS-Gym* to accomplish this simulation.

4.2 System Design

We compose *MARS-Gym* with three main internal components: The Data Engineering Module, the Simulation Module, and the Evaluation Module. Fig. 2 shows a visual representation of our implementation. In the remainder of this subsection, we describe each module separately.

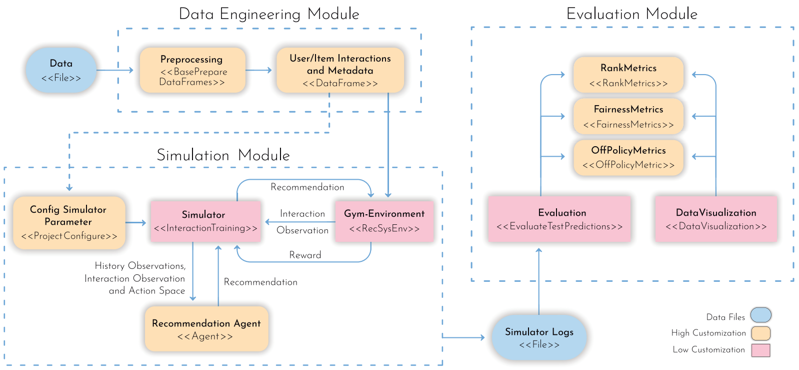


Figure 2: *MARS-Gym* Architecture and its three internal modules.

4.2.1 Data Engineering Module

This module processes the incoming data to satisfy all requirements to create the OpenAI's Gym environment. It cleans the dataset and applies all transformations to provide the list of interactions and metadata associated with the user or partners as output.

We ingest the dataset where each row should have the user identifier, the fields containing contextual information, the action (partner) presented to the user, and the success flag (if an interaction is successful in the marketplace scenario).

Naturally, distinct marketplaces collect data in many different ways. Therefore, it is impossible to provide a single data processing task that handles all scenarios. For this reason, this module is highly customizable, and we provide abstractions and tools to help in creating data processing tasks.

4.2.2 Simulation Module

This module is the core of *MARS-Gym* and implements the MDP counterfactual simulation described in Subsection 4.1. Additionally, it is also the module to design and train learning agents. We compose *MARS-Gym* with three main components: the Recommendation Agent, the Gym Environment, and the Simulator.

As previously described, the Gym Environment implements the interface of OpenAI's Gym. It ingests the processed dataset to create a representation of marketplace dynamics by using the same successful interactions and computing the rewards using their corresponding actions as ground truth. On the other side, the Recommendation Agent module implements the agent's interface, which exposes the *act* and the *fit* methods. Specifically, *MARS-Gym* requires the *act* method to return not only the action but also the respective probabilities of each available action. Therefore, any decision-making system that satisfies these requirements can run a simulation.

For the *fit* method, *MARS-Gym* expects to train the agent using previous experience. We already provide high-level PyTorch modules with parameterized architectures (including logistic regression, factorization machine, and neural networks), state-of-the-art gradient-based optimization algorithms, and regularization methods. As we also implement the whole PyTorch's optimization procedure (with GPU support), the only work we left for the user is to combine these building blocks and select the appropriated hyperparameters.

The third component is the Simulator, which plays the central role for the module. It manages both agent and environment to conduct the simulation across episodes, accumulating the experience data for training and the rewards for evaluation. The proposed architecture assumes low customization in this module, as well as for the Gym Environment. In other words, we do not expect any modification in these modules in order to simulate new agents or marketplaces.

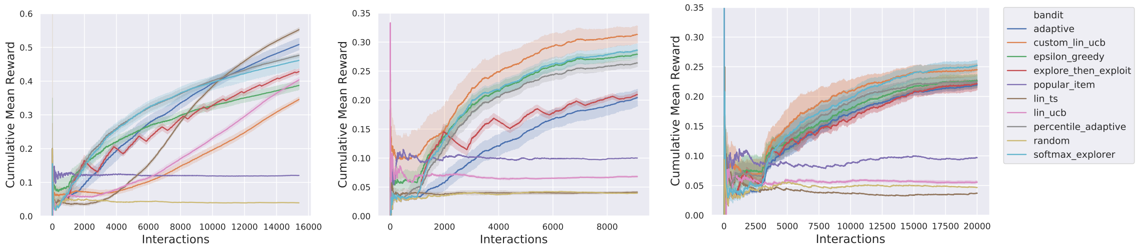
4.2.3 Evaluation Module

This final module inputs the logs generated from MDP counterfactual simulation to perform a multi-sided evaluation. The evaluation task computes metrics of recommendation, off-policy evaluation, and fairness. Then, the data visualization component provides a user-friendly interface to present and compare these metrics among different agents (see Appendix D for some samples).

The Simulation Module generates two different types of logs to be ingested by the Evaluation Module. During the simulation, it splits the processed dataset into train and test subsets. Naturally, it uses the training subset to conduct the training process, and the Evaluation Module uses the resultant logs to compute and plot the cumulative mean reward. On the other hand, the test subset provides a new perspective of the same MDP, but the agent interacts with it only *after* training, in order to measure generalization. With this test subset, we compute traditional metrics for Recommender Systems, such as Precision, MaP [42], nDCG [22], Coverage [18] and Personalization.

In terms of off-policy evaluation, we implement the policy value estimators listed in Subsection 3.2: Direct Method, Inverse Propensity Score (and its variations CIPS and SNIPS), and Doubly-Robust Estimator. For these metrics, we compute the expected reward estimator, $\hat{\varrho}(x, a)$, and the collection policy estimator, $\hat{\pi}_c$, as described in Subsection 3.2. They are automatically trained using the whole processed dataset (i.e., the output of the Data Engineering Module) and the same base model of the agent. Then, we use the ground-truth actions and the list of probabilities to compute the metrics.

Finally, for fairness evaluation, we propose implementations for the three perspectives listed in section 3.3. Specifically, for disparate mistreatment (which will be used in this paper) we consider the ground-truth actions in interactions logs to compute classification metrics (such as accuracy, true positive rate, and others). They are grouped by the values of the sensitive attribute, so that we can evaluate the contition in equation 3.



(a) Como, Italy

(b) Rio de Janeiro, Brazil

(c) RecSys Cities

Figure 3: Bandit Simulation Results.

5 Experimental Results and Analysis

In this section, we present our baseline agents, evaluation results and interpretations. We hope to illustrate the usage of the framework and to facilitate the work of new users. For evaluation, we used the Trivago markeplace dataset [33], which we describe in Appendix A. We implemented a diverse set of Contextual Bandits as baselines for easy extension and comparison with other agents. They apply a variety of explorations strategies, most of them inspired by Cortes [12]. We also proposed CustomLinUCB, which extends LinUCB by implementing a non-linear reward model.

An agent consists of an oracle that predicts scores for each possible interaction between the user and the available actions in the specific context. Therefore, the exploration strategies use these scores to compute probabilities and choose an action. We trained the agents via off-policy batch learning, applying the CRM loss function. The training procedure happened in epochs of a fixed number of interactions, using the whole experience buffer acquired throughout the simulation. Finally, we also performed hyperparameter tuning in all agents and presented the best set found for each of them.

5.1 Bandit Simulation Results

We performed the simulation of each proposed task until the convergence of most methods to observe cumulative mean reward throughout the simulation. For statistical significance, we represent each curve by the mean and confidence interval across five executions of the same experiment. Fig. 3 presents the results of the simulations for few different tasks and methods. The complete list of simulation results is in Appendix B.

We observed that he performance of these methods dramatically varies across tasks, which reinforces that there is no better exploration strategy for all scenarios. Nevertheless, the experiments also show some patterns among the tasks. Linear methods like LinTS and LinUCB obtain better results when the search space is smaller but does not scale well as the dimensionality of the MDP increases. On the other hand, the proposed CustomLinUCB presents better results in these scenarios but has worse sample efficiency in the low data regime. We hypothesize that the introduction of non-linearities in the oracle increases the capacity of the representation of contextual information in the policy, at the cost of a harder optimization problem.

The curve of the Explore-Exploit agent, which locally interleaves peaks and valleys as the method switches from full exploitation to full exploration are important to highlight. It suggests a sensibility of the switch hyperparameter, which is undesirable for production environments. Ultimately, we also point out that simple methods (in an implementation perspective) such as ϵ-greedy and softmax explorer present satisfactory results, suggesting to be ideal for composing more complex agents and algorithms as initial setup.

5.2 Recommendation Metrics and Off-Policy Evaluation

We evaluated the bandits according to traditional recommendation metrics and off-policy metrics in the test subset of the “Chicago, USA" task. Table 1 presents the average results over five executions for some agents and metrics in the framework. For the complete list, we refer to Appendix C.

Table 1: Recommendation Metrics for "Chicago, USA" task.

		Rank Metrics		Off-Policy	
	Precision	NDCG@5	Cove@5	SNIPS	DR
CustomLinUCB	**0.338**	0.443	0.371	0.319	**0.299**
AdaptiveExplorer	0.331	0.415	0.379	**0.331**	0.291
SoftmaxExplorer	0.316	**0.446**	0.328	0.302	0.281
ϵ-Greedy	0.311	0.441	0.343	0.322	0.278
LinUCB	0.065	0.183	0.297	0.049	0.043
MostPopularItem	0.074	0.199	0.153	0.052	0.061
Without CRM	**0.354**	**0.457**	**0.303**	0.346	0.312
With CRM	0.344	0.434	0.300	**0.352**	**0.320**

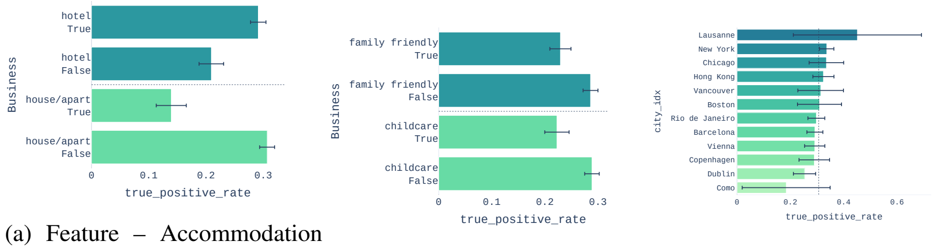
The precision metric highly correlates to the simulations results in Fig. 5b (see Appendix B). CustomLinUCB presented the best results in both the simulated cumulative mean reward and doubly robust value. For other bandits, the performance varies from counterfactual simulation to rank metrics and off-policy evaluation. We hypothesize that these changes are related to the sample efficiency of each exploration strategy, as well as the generalization properties from the learned policies. Furthermore, we observe that the SoftmaxExplorer outperforms all other methods in the nDCG metric, which suggests good performance in scenarios of slate recommendation.

We observe a consistent drop from the precision to the off-policy metrics, suggesting that all methods naturally exploit the sampling bias. Comparatively, we see a high correlation between those metrics. However, the ranking of bandits is not the same, which suggests that some bandits suffer more from sampling bias. This evidence and analysis are important for model deployment, once that off-policy metrics often correlates better with online experiments [24, 23].

Ultimately, we conducted an ablation study regarding CRM loss. For this purpose, we trained two policies using the training data in the full offline setting (i.e., directly using the interaction logs for optimization, with no MDP simulation), varying the loss function. From Table 1, we observe that the policy with CRM trades off accuracy in recommendation metrics to improve the unbiased, off-policy metrics. This result validates that this technique is essential to reduce the effect of sampling bias during training, and, therefore, improve off-policy evaluation.

5.3 Fairness Evaluation

We evaluated the SoftmaxExplorer bandit on the “RecSys Cities" task, in the perspective of disparate mistreatment (Fig. 4). We selected a few sensitive attributes across partners in the marketplace.



(a) Feature – Accommodation Type

(b) Feature – Business Affinity

(c) Feature – City

Figure 4: Fairness analysis for SoftmaxExplorer.

In Fig. 4a, we present features related to the type of accomodation. We observe that hotels receive a much better recommendation from the agent than the other types, such as hostels or apartments, suggesting disparate mistreatment, which is undesirable for the health of the marketplace. Fig. 4b also presents disparate mistreatment for accommodations that are family-friendly or provides childcare. These are not only business features from partners but also user requirements.

From a different perspective, Fig. 4c shows the true positive rates for each city analyzed. In general, we see similar results for all of them. The metrics for some cities present higher confidence intervals, which is directly related to having fewer data. Nevertheless, we diagnose differences among some cities (e.g., comparing New York and Dublin), which we could explore to understand the source of unfairness and improve the recommender system.

Ultimately, all charts in Fig. 4 diagnose different scenarios in which the recommender system does not satisfy the current notions of fairness. They are the reproduction of many biases in the dataset, and the machine learning pipeline and platform must address it. Therefore, these metrics provide insights into where the system needs to improve to maintain the marketplace fairness and healthiness.

6 Conclusions and Future Work

In this work, we proposed *MARS-Gym*, an open-source framework to model, train, and evaluate RL-based recommender systems for marketplaces. We presented in detail its internal components and provide baseline implementation and analysis to serve as an introduction for the users. We also point out that the presented results are an extra contribution of our work on the task of benchmarking contextual bandits, complementing the results of previous works.

As future releases of this framework, we plan to implement extensions for multi-objective and constrained optimization to extend the support for marketplaces with multiples stakeholders during learning. It addresses, for example, the recommendation in marketplaces with a delivery system, where logistic restrictions arise. We also plan to extend the framework to address the hierarchical setting, where the agent manages multiples recommendation models rather than solely actions. This problem is valuable for current production systems, where we often have a diverse set of models running concurrently and need to evaluate their adaptability and degradability over time.

Acknowledgments and Disclosure of Funding

We would like to thank iFood Research for financial support and helpful reviews.

References

[1] Himan Abdollahpouri, Robin Burke, and Bamshad Mobasher. Recommender systems as multistakeholder environments. In *Proceedings of the 25th Conference on User Modeling, Adaptation and Personalization*, UMAP '17, page 347–348, New York, NY, USA, 2017. Association for Computing Machinery. ISBN 9781450346351. doi: 10.1145/3079628.3079657. URL `https://doi.org/10.1145/3079628.3079657`.

[2] Fabian Abel, Yashar Deldjoo, Mehdi Elahi, and Daniel Kohlsdorf. Recsys challenge 2017: Offline and online evaluation. In *Proceedings of the Eleventh ACM Conference on Recommender Systems*, pages 372–373, New York, NY, USA, 2017. Association for Computing Machinery. ISBN 9781450346528. doi: 10.1145/3109859.3109954. URL `https://doi.org/10.1145/3109859.3109954`.

[3] Jens Adamczak. Recsys challenge 2019 · trivago tech blog, March 2019. URL `https://tech.trivago.com/2019/03/11/recsys-challenge-2019`.

[4] Rishabh Agarwal, Dale Schuurmans, and Mohammad Norouzi. An optimistic perspective on offline reinforcement learning, 2020.

[5] Xavier Amatriain and Justin Basilico. Recommender systems in industry: A netflix case study. In *Recommender systems handbook*, pages 385–419. Springer, 2015.

[6] James Bennett, Stan Lanning, et al. The netflix prize. In *Proceedings of KDD cup and workshop*, volume 2007, page 35, San Jose, California, USA, 2007. Citeseer, ACM SIGKDD Explorations Newsletter.

[7] Léon Bottou, Jonas Peters, Joaquin Quiñonero-Candela, Denis X. Charles, D. Max Chickering, Elon Portugaly, Dipankar Ray, Patrice Simard, and Ed Snelson. Counterfactual reasoning and learning systems: The example of computational advertising. *Journal of Machine Learning Research*, 14(65):3207–3260, 2013. URL `http://jmlr.org/papers/v14/bottou13a.html`.

[8] Léon Bottou, Jonas Peters, Joaquin Quiñonero-Candela, Denis X. Charles, D. Max Chickering, Elon Portugaly, Dipankar Ray, Patrice Simard, and Ed Snelson. Counterfactual reasoning and learning systems, 2013.

[9] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym. *CoRR*, abs/1606.01540, 2016. URL `http://arxiv.org/abs/1606.01540`.

[10] Robin Burke. Multisided fairness for recommendation. *CoRR*, abs/1707.00093, 2017. URL `http://arxiv.org/abs/1707.00093`.

[11] Serkan Cabi, Sergio Gómez Colmenarejo, Alexander Novikov, Ksenia Konyushkova, Scott Reed, Rae Jeong, Konrad Zolna, Yusuf Aytar, David Budden, Mel Vecerik, Oleg Sushkov, David Barker, Jonathan Scholz, Misha Denil, Nando de Freitas, and Ziyu Wang. Scaling data-driven robotics with reward sketching and batch reinforcement learning, 2020.

[12] David Cortes. Adapting multi-armed bandits policies to contextual bandits scenarios. *CoRR*, abs/1811.04383, 2018. URL `http://arxiv.org/abs/1811.04383`.

[13] Miroslav Dudík, John Langford, and Lihong Li. Doubly robust policy evaluation and learning. In *Proceedings of the 28th International Conference on International Conference on Machine Learning*, ICML'11, page 1097–1104, Madison, WI, USA, 2011. Omnipress. ISBN 9781450306195.

[14] Gabriel Dulac-Arnold, Daniel Mankowitz, and Todd Hester. Challenges of real-world reinforcement learning, 2019.

[15] Alexander D'Amour, Hansa Srinivasan, James Atwood, Pallavi Baljekar, D. Sculley, and Yoni Halpern. Fairness is not static: Deeper understanding of long term fairness via simulation studies. In *Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency*, FAT* '20, page 525–534, New York, NY, USA, 2020. Association for Computing Machinery. ISBN 9781450369367. doi: 10.1145/3351095.3372878. URL `https://doi.org/10.1145/3351095.3372878`.

[16] Justin Fu, Aviral Kumar, Ofir Nachum, George Tucker, and Sergey Levine. D4rl: Datasets for deep data-driven reinforcement learning, 2020.

[17] Scott Fujimoto, Edoardo Conti, Mohammad Ghavamzadeh, and Joelle Pineau. Benchmarking batch deep reinforcement learning algorithms, 2019.

[18] Mouzhi Ge, Carla Delgado-Battenfeld, and Dietmar Jannach. Beyond accuracy: Evaluating recommender systems by coverage and serendipity. In *Proceedings of the Fourth ACM Conference on Recommender Systems*, RecSys '10, page 257–260, New York, NY, USA, 2010. Association for Computing Machinery. ISBN 9781605589060. doi: 10.1145/1864708.1864761. URL `https://doi.org/10.1145/1864708.1864761`.

[19] X. Geng, H. Zhang, J. Bian, and T. Chua. Learning image and user features for recommendation in social networks. In *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 4274–4282. IEEE, 2015.

[20] Caglar Gulcehre, Ziyu Wang, Alexander Novikov, Tom Le Paine, Sergio Gomez Colmenarejo, Konrad Zolna, Rishabh Agarwal, Josh Merel, Daniel Mankowitz, Cosmin Paduraru, Gabriel Dulac-Arnold, Jerry Li, Mohammad Norouzi, Matt Hoffman, Ofir Nachum, George Tucker, Nicolas Heess, and Nando de Freitas. Rl unplugged: Benchmarks for offline reinforcement learning, 2020.

[21] F. Maxwell Harper and Joseph A. Konstan. The movielens datasets: History and context. *ACM Trans. Interact. Intell. Syst.*, 5(4), December 2015. ISSN 2160-6455. doi: 10.1145/2827872. URL `https://doi.org/10.1145/2827872`.

[22] Kalervo Järvelin and Jaana Kekäläinen. Ir evaluation methods for retrieving highly relevant documents. In *Proceedings of the 23rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '00, page 41–48, New York, NY, USA, 2000. Association for Computing Machinery. ISBN 1581132263. doi: 10.1145/345508.345545. URL `https://doi.org/10.1145/345508.345545`.

[23] Olivier Jeunen, Dmytro Mykhaylov, David Rohde, Flavian Vasile, Alexandre Gilotte, and Martin Bompaire. Learning from bandit feedback: An overview of the state-of-the-art, 2019.

[24] Olivier Jeunen, David Rohde, and Flavian Vasile. On the value of bandit feedback for offline recommender system evaluation, 2019.

[25] Peter Knees, Yashar Deldjoo, Farshad Bakhshandegan Moghaddam, Jens Adamczak, Gerard-Paul Leyson, and Philipp Monreal. Recsys challenge 2019: Session-based hotel recommendations. In *Proceedings of the Thirteenth ACM Conference on Recommender Systems*, RecSys '19, New York, NY, USA, 2019. ACM. ISBN 978-1-4503-6243-6/19/09. doi: 10.1145/3298689.3346974. URL `https://doi.org/10.1145/3298689.3346974`.

[26] Aviral Kumar, Justin Fu, George Tucker, and Sergey Levine. Stabilizing off-policy q-learning via bootstrapping error reduction, 2019.

[27] Sergey Levine, Aviral Kumar, George Tucker, and Justin Fu. Offline reinforcement learning: Tutorial, review, and perspectives on open problems, 2020.

[28] Lihong Li, Shunbao Chen, Jim Kleban, and Ankur Gupta. Counterfactual estimation and optimization of click metrics in search engines: A case study. In *Proceedings of the 24th International Conference on World Wide Web*, WWW '15 Companion, page 929–934, New York, NY, USA, 2015. Association for Computing Machinery. ISBN 9781450334730. doi: 10.1145/2740908.2742562. URL `https://doi.org/10.1145/2740908.2742562`.

[29] R Logesh and V Subramaniyaswamy. Exploring hybrid recommender systems for personalized travel applications. In *Cognitive informatics and soft computing*, pages 535–544. Springer, 2019.

[30] James McInerney, Benjamin Lacker, Samantha Hansen, Karl Higley, Hugues Bouchard, Alois Gruson, and Rishabh Mehrotra. Explore, exploit, and explain: Personalizing explainable recommendations with bandits. In *Proceedings of the 12th ACM Conference on Recommender*

Systems, RecSys '18, page 31–39, New York, NY, USA, 2018. Association for Computing Machinery. ISBN 9781450359016. doi: 10.1145/3240323.3240354. URL https://doi.org/10.1145/3240323.3240354.

[31] Rishabh Mehrotra, James McInerney, Hugues Bouchard, Mounia Lalmas, and Fernando Diaz. Towards a fair marketplace: Counterfactual evaluation of the trade-off between relevance, fairness & satisfaction in recommendation systems. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, CIKM '18, page 2243–2251, New York, NY, USA, 2018. Association for Computing Machinery. ISBN 9781450360142. doi: 10.1145/3269206.3272027. URL https://doi.org/10.1145/3269206.3272027.

[32] Ashvin Nair, Murtaza Dalal, Abhishek Gupta, and Sergey Levine. Accelerating online reinforcement learning with offline datasets, 2020.

[33] The ACM Conference Series on Recommender Systems. Acm recsys challenge 2019 | dataset, 2019. URL https://recsys.trivago.cloud/challenge/dataset.

[34] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019. URL http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf.

[35] Gourab K Patro, Abhijnan Chakraborty, Niloy Ganguly, and Krishna P. Gummadi. Incremental fairness in two-sided market platforms: On smoothly updating recommendations, 2019.

[36] David Rohde, Stephen Bonner, Travis Dunlop, Flavian Vasile, and Alexandros Karatzoglou. Recogym: A reinforcement learning environment for the problem of product recommendation in online advertising, 2018.

[37] Sherwin Rosen. The economics of superstars. *American Economic Review*, 71(5):845–58, 1981. URL https://EconPapers.repec.org/RePEc:aea:aecrev:v:71:y:1981:i:5:p:845-58.

[38] Bichen Shi, Makbule Gulcin Ozsoy, Neil Hurley, Barry Smyth, Elias Z. Tragos, James Geraci, and Aonghus Lawlor. Pyrecgym: A reinforcement learning gym for recommender systems. In *Proceedings of the 13th ACM Conference on Recommender Systems*, RecSys '19, page 491–495, New York, NY, USA, 2019. Association for Computing Machinery. ISBN 9781450362436. doi: 10.1145/3298689.3346981. URL https://doi.org/10.1145/3298689.3346981.

[39] R.S. Sutton and A.G. Barto. *Reinforcement Learning: An Introduction*. Adaptive Computation and Machine Learning series. MIT Press, 2nd edition, 2018. ISBN 9780262039246. URL https://books.google.com.br/books?id=sWV0DwAAQBAJ.

[40] Adith Swaminathan and Thorsten Joachims. Counterfactual risk minimization: Learning from logged bandit feedback. In *Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37*, ICML'15, page 814–823. JMLR.org, 2015.

[41] Yuyan Wang, Yuanchi Ning, Isaac Liu, and Xian Xing Zhang. Food discovery with uber eats: Recommending for the marketplace, 2018. URL https://eng.uber.com/uber-eats-recommending-marketplace/.

[42] Yisong Yue, Thomas Finley, Filip Radlinski, and Thorsten Joachims. A support vector method for optimizing average precision. In *Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '07, page 271–278, New York, NY, USA, 2007. Association for Computing Machinery. ISBN 9781595935977. doi: 10.1145/1277741.1277790. URL https://doi.org/10.1145/1277741.1277790.

[43] Muhammad Bilal Zafar, Isabel Valera, Manuel Gomez Rodriguez, and Krishna P. Gummadi. Fairness beyond disparate treatment & disparate impact: Learning classification without disparate mistreatment. In *Proceedings of the 26th International Conference on World Wide Web*, WWW '17, page 1171–1180, Republic and Canton of Geneva, CHE, 2017. International World Wide Web Conferences Steering Committee. ISBN 9781450349130. doi: 10.1145/3038912.3052660. URL `https://doi.org/10.1145/3038912.3052660`.

[44] Hamed Zamani, Markus Schedl, Paul Lamere, and Ching-Wei Chen. An analysis of approaches taken in the acm recsys challenge 2018 for automatic music playlist continuation. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 10(5):1–21, 2019.

A Benchmark Description

In this section, we describe the Trivago marketplace dataset [33], which we ingested into *MARS-Gym* for experiments.

A.1 Trivago Marketplace

Trivago is a global hotel search platform located in more than 190 countries and provides access to more than two million hotels for travelers. It centralizes hotel offers on a single platform. Hence, the platform needs to give good recommendations to users and to ensure that affiliated hotels are treated fairly by the platform in different scenarios. Furthermore, user preferences change over time and depend on the trip purpose, as well as accommodation prices, type, and availability. Finally, there is an interest of all partners (traveler, advertising booking site, and Trivago) to suggest good recommendations in different aspects [3].

A.2 Dataset Description and Benchmark Tasks

Trivago organized the ACM RecSys Challenge in 2019 [33]. For this competition, it provided a dataset that consists of session logs with 910k samples in approximately 35k different cities. Each session contains a sequence of interactions between a user and the platform. They can represent different actions, such as rating, get item metadata (info, image, and deals), sort list, search for a destination or point of interest. In addition to the user session information, the dataset also provides different item metadata that characterize the hotels. Table 2 shows the general statistics of the Trivago challenge dataset.

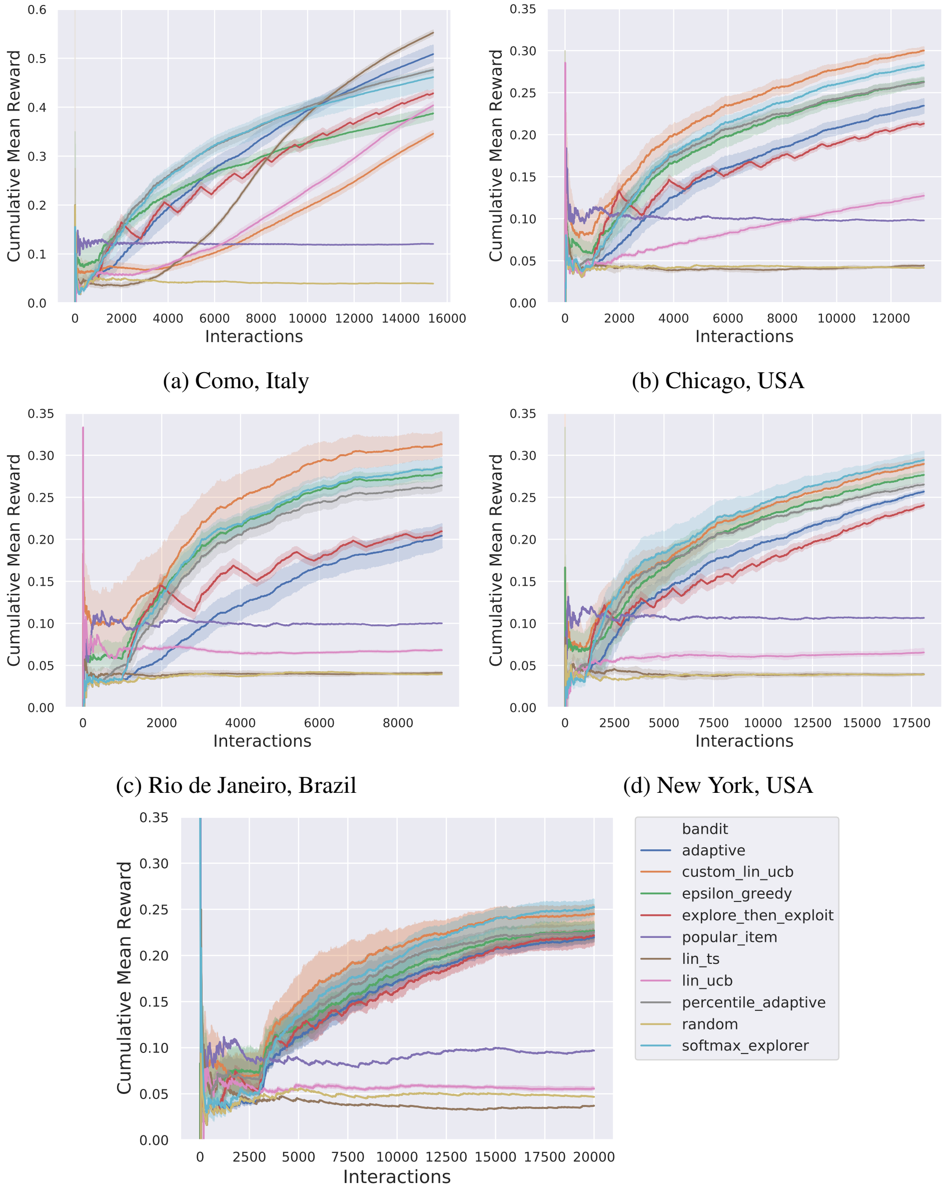
Table 2: Trivago marketplace – dataset statistics and benchmark tasks

Task	Interactions	clickouts	user	session	item
RecSys Cities	761,702	62,168	31,075	36,846	12,002
New York, USA	223,320	18,160	9,158	10,869	1,961
Rio de Janeiro, Brazil	161,973	9,122	4,429	5,563	2,080
Chicago, USA	22,939	1,890	1,155	1,347	662
Como, Italy	1,718	155	112	122	328

We organized the dataset in five tasks, which are primarily identified by the city of interest. In Table 2, we present the statistics for each proposed task. The task "RecSys Cities" contains a set of 12 cities presented in the dataset. We chose cities based on several factors: the **episode length**, to represent how long is the sequence of actions that the agent handles; the **state space size**, represented by the number of unique sessions and users, which exposes the variability of contextual information; the **action space size** (i.e., the number of available partners for recommendation), which indicates how complex is the exploration problem; and the **number of clickouts**, which gives an idea of how many successful interactions the city has to be used as ground-truth actions during the simulation. For further information, we refer to the dataset page [33] and our source code.

B Bandit Simulation Results

Fig. 5 shows the cumulative mean reward throughout the simulation all bandits provided in *MARS-Gym*, for all benchmark tasks described in Appendix A.



(a) Como, Italy

(b) Chicago, USA

(c) Rio de Janeiro, Brazil

(d) New York, USA

(e) RecSys Cities

Figure 5: Bandit Simulation Results.

C Recommendation Metrics and Off-Policy Evaluation

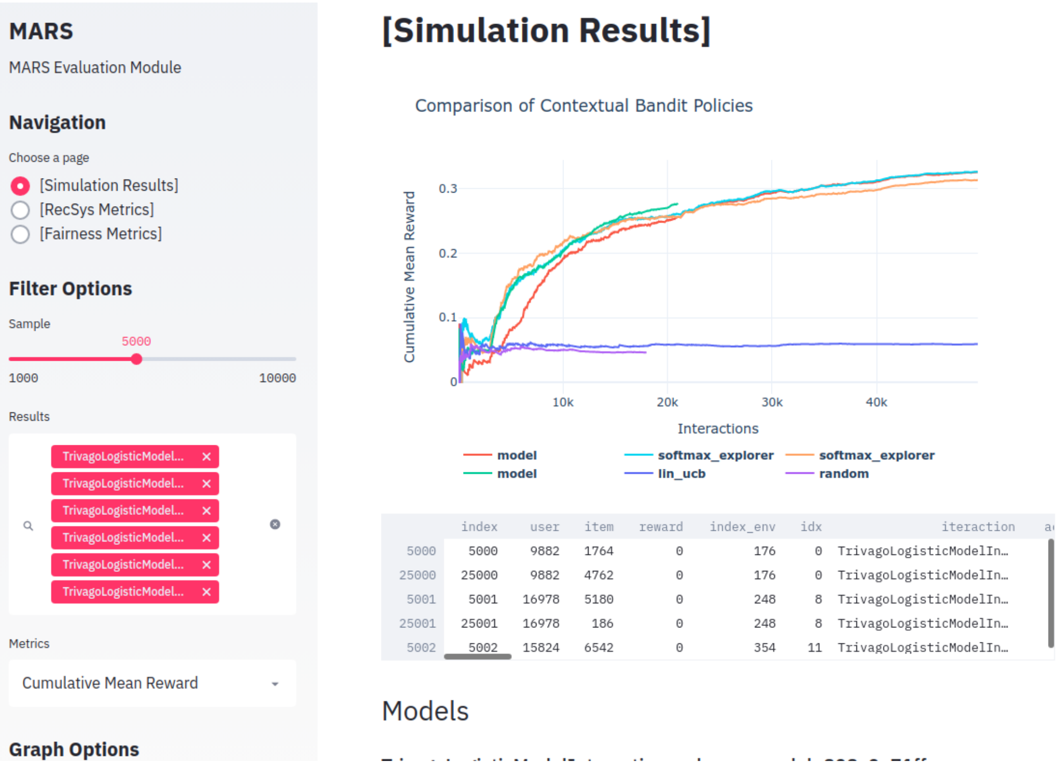
Table 3 shows the recommendation and off-policy metrics for all contextual bandits provided by *MARS-Gym* as baselines, in the context of task "Chicago, USA".

Table 3: Recommendation Metrics for "Chicago, USA" task.

	Classic Recommendation Metrics				Off-Policy Metrics			
	Precision	NDCG@5	Cove@5	Per@5	IPS	SNIPS	DE	DR
CustomLinUCB	**0.338**	0.443	0.371	0.724	**0.324**	0.319	0.180	**0.299**
AdaptiveGreedy	0.331	0.415	0.379	0.770	0.314	**0.331**	**0.181**	0.291
PercentileAdaptiveGreedy	0.319	0.426	0.364	0.750	0.306	0.307	0.171	0.279
SoftmaxExplorer	0.316	**0.446**	0.328	0.727	0.308	0.302	0.171	0.281
ExploreThenExploit	0.315	0.423	0.313	0.737	0.308	0.316	0.167	0.280
ϵ-Greedy	0.311	0.441	0.343	0.737	0.305	0.322	0.165	0.278
LinUCB	0.065	0.183	0.297	0.721	0.047	0.049	0.033	0.043
LinTS	0.029	0.112	**0.419**	**0.778**	0.031	0.033	0.018	0.031
MostPopularItem	0.074	0.199	0.153	0.592	0.063	0.052	0.046	0.061
Random	0.042	0.145	0.392	0.777	0.028	0.031	0.024	0.029
Policy without CRM	**0.354**	**0.457**	**0.303**	0.717	0.340	0.346	**0.199**	0.312
Policy with CRM	0.344	0.434	0.300	**0.730**	**0.350**	**0.352**	0.195	**0.320**

D Visualization Tool Samples

In this appendix, we present some screenshots samples from *MARS-Gym* visualization tool. It provides a user-friendly interface to facilitate feature analysis and the comparison among learning agents.



MARS

MARS Evaluation Module

Navigation

Choose a page

● [Simulation Results]
○ [RecSys Metrics]
○ [Fairness Metrics]

Filter Options

Sample: 5000

Results: TrivagoLogisticModel... ×, TrivagoLogisticModel... ×, TrivagoLogisticModel... ×, TrivagoLogisticModel... ×, TrivagoLogisticModel... ×, TrivagoLogisticModel... ×

Metrics: Cumulative Mean Reward

Graph Options

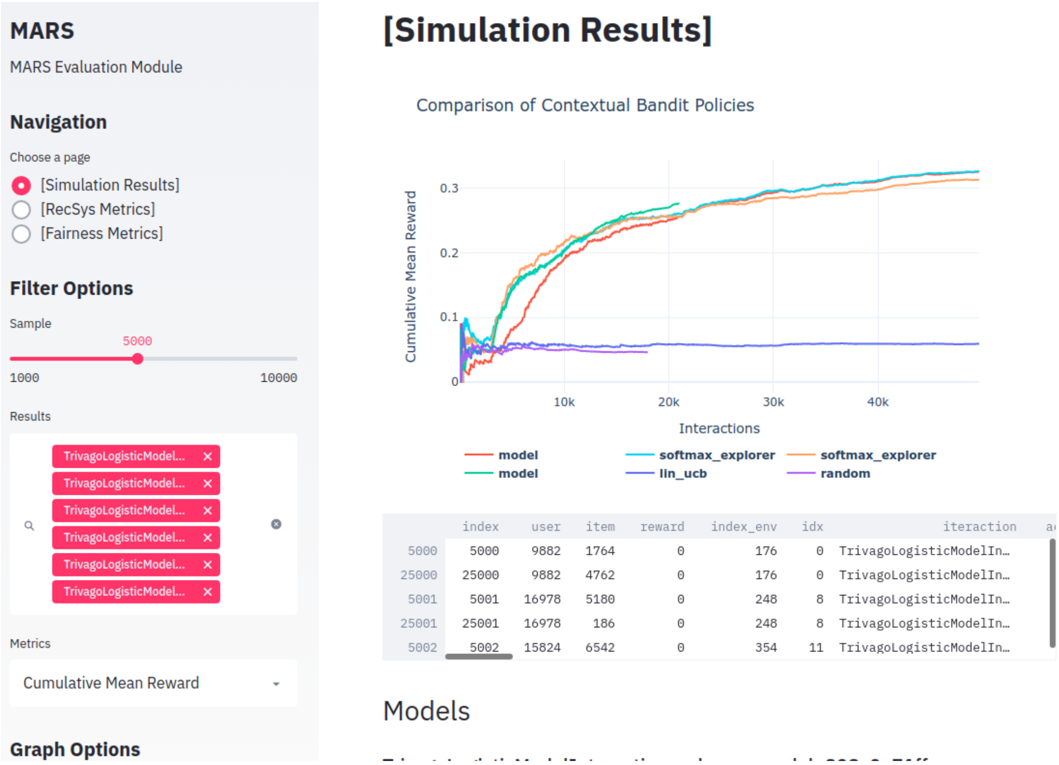
[Simulation Results]

Comparison of Contextual Bandit Policies

	index	user	item	reward	index_env	idx	iteraction
5000	5000	9882	1764	0	176	0	TrivagoLogisticModelIn…
25000	25000	9882	4762	0	176	0	TrivagoLogisticModelIn…
5001	5001	16978	5180	0	248	8	TrivagoLogisticModelIn…
25001	25001	16978	186	0	248	8	TrivagoLogisticModelIn…
5002	5002	15824	6542	0	354	11	TrivagoLogisticModelIn…

Models

Figure 6: Visualization – Simulation Results.



MARS

MARS Evaluation Module

Navigation

Choose a page

● [Simulation Results]
○ [RecSys Metrics]
○ [Fairness Metrics]

Filter Options

Sample: 5000 (1000 – 10000)

Results

TrivagoLogisticModel... ✕
TrivagoLogisticModel... ✕
TrivagoLogisticModel... ✕
TrivagoLogisticModel... ✕
TrivagoLogisticModel... ✕
TrivagoLogisticModel... ✕

Metrics: Cumulative Mean Reward

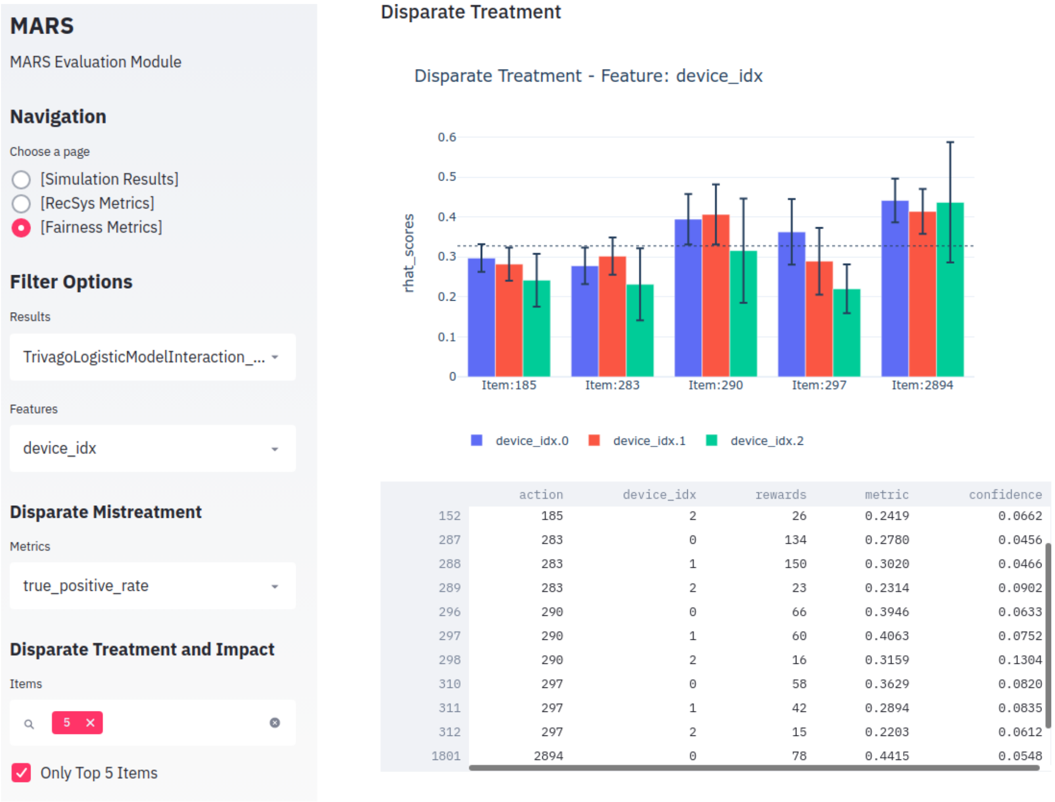
Graph Options

[Simulation Results]

	index	user	item	reward	index_env	idx	iteraction
5000	5000	9882	1764	0	176	0	TrivagoLogisticModelIn…
25000	25000	9882	4762	0	176	0	TrivagoLogisticModelIn…
5001	5001	16978	5180	0	248	8	TrivagoLogisticModelIn…
25001	25001	16978	186	0	248	8	TrivagoLogisticModelIn…
5002	5002	15824	6542	0	354	11	TrivagoLogisticModelIn…

Models

Figure 7: Visualization – Rank metrics and Off-policy evaluation.



MARS

MARS Evaluation Module

Navigation

Choose a page

○ [Simulation Results]
○ [RecSys Metrics]
● [Fairness Metrics]

Filter Options

Results: TrivagoLogisticModelInteraction_...

Features: device_idx

Disparate Mistreatment

Metrics: true_positive_rate

Disparate Treatment and Impact

Items: 5 ✕

☑ Only Top 5 Items

Disparate Treatment

	action	device_idx	rewards	metric	confidence
152	185	2	26	0.2419	0.0662
287	283	0	134	0.2780	0.0456
288	283	1	150	0.3020	0.0466
289	283	2	23	0.2314	0.0902
296	290	0	66	0.3946	0.0633
297	290	1	60	0.4063	0.0752
298	290	2	16	0.3159	0.1304
310	297	0	58	0.3629	0.0820
311	297	1	42	0.2894	0.0835
312	297	2	15	0.2203	0.0612
1801	2894	0	78	0.4415	0.0548

Figure 8: Visualization – Fairness Results.