A. NOTE ON REPRODUCIBILITY

For the research presented in this paper, We implemented two versions of our algorithm, including a version tailored for a large-scale distributed analytics platform for the real-world experiments, and another version suitable for use in a single-machine python environment. We enclose the code for the single-machine implementation of our algorithm along with the submission.

For our reported synthetic data experiments, we provide a detailed description of the data generating process and the implementation of all the methods in the experiments. For the real-world experiments on a large-scale distributed analytics platform, we describe the details of the experiments to enhance the understanding of the role of our proposed method.

The environment we use is:

- **Python 2.8.1**
- **scikit-learn=0.22.1**
- **numpy=1.18.1**
- **pandas=1.0.0**

Codebase to reproduce the experimental results:

- **run_simulation.py**:Main script for the simulation in Section 4.1
- **run_auction.py**: Main script for the simulated auction in Section 4.2.
- **ctrf**: Module contains the CTRF implementations.

We also provide two example bash scripts **simulated_data_varying.sh** for running the simulated experiments in Section 4.1 and **simulated_auction_reverse_varing,sh.** for simulated auction in Section 4.2

The implementation details for each algorithm:

- LR: The **LogisticRegression** class in **sklearn.linear_model**, with L_2 penalty.
- GBDT: **GradientBoostingClassifier** class in **sklearn.ensemble**.
- CNT-RF/RND-RF/Combine-RF: **RandomForestClassifier** class in **sklearn.ensemble**, trained on corresponding data.
- CTRF: Python module provided in codebase.

For methods adjusted with sampling weights (LR-IPW,GBDT-IPW), we first estimate the ratio of density with a logistic regression to classify training data and testing data [2]. We calculate the IPW weights $w_i = p_i^{\text{test}}/(1 - p_i^{\text{test}})$, where p_i^{test} is the predicted probability for unit i belongs to testing data. We feed the IPW weights when fitting model with, **model.fit(X,y,sample_weight=weights)**, where **weights** is the IPW weights.

For the hyperparameters in the random forest model, we set the number of trees to be 50, 0.3 feature subsampling rate and the max number of nodes to be 100 (same for GBDT model).

B. DETAILS ON EXPERIMENTS

B.1 Details on Synthetic Auction

We enumerate the steps for generating the synthetic auction data.

- Step 1: We utilize the **scikit-learn.make_classification** function to generate synthetic relevance data (x, y). Each data point corresponds to one ad to be shown.
- Step 2: We fit a random forest model to the data to calculate a relevance score/probability of being click p for each ad.
- Step 3: We run a simulated auction based on the relevance score with some additional noise p'. Each auction determines the ad's layout on one page. In each auction, 20 ads are being considered to compete for the position in the layout with at most 5 slots. Notice that the relevance reserve serves as a filter to determine whether the ad can join the auction.
- Step 4: we assign position based on the p' in the auction with high relevance assigned to the top position.
- Step 5: We generate click based on true relevance score p with Bernoulli trials and randomly pick up one ad to click if the user would click multiple ads on the same page.

For randomized data, we skip the auction stage and simply randomly pick some ads to show on the page. We run 10000 auctions for the randomized data and 25000 auctions on the log data. The final sample size ratio between randomized and log data is approximately 1:5.

B.2 Details on End-to-End Optimization Task

We also provide a detailed description on how we calculate the degree of feature shifts in real-world task and how we pick up the optimization tasks.

In order to compute distribution shift between two different environments, we use discrete bins to represent each feature as a multinomial distribution similar to approach described in [1]. After that, we applied Jensen-Shannon (JS) divergence metric to compute the similarity of two multinomial distribution for the same feature in counterfactual vs factual environment. The JS Divergence of two probability distribution P and Q are given below:

$$JS(P||Q) = \frac{1}{2}D_{KL}(P||M) + \frac{1}{2}D_{KL}(Q||M), M = \frac{1}{2}(P+Q) \tag{1}$$

JS Divergence is the symmetric version of Kullback–Leibler divergence which can be computed as below for a given multinomial distribution with k different bins.

$$D_{KL}(P||Q) = \sum_{i}^{k}(P(i)\ ln(\frac{P(i)}{Q(i)})) \tag{2}$$

Once the JS divergence of each feature is computed based on counterfactual (P) vs factual (P^*) feature distributions, the final distribution shift score (DS) over N features is computed as root mean square of all JS divergence values across all features as follows:

$$DS = \sqrt{\frac{1}{N}\sum_{i}^{N}\left[JS(P_i||P_i^*)\right]^2} \tag{3}$$

The distribution shift (DS) scores for selected real use cases are given below in Table 1. We calculate the feature shifts of 10 candidate task in total and compare the two tasks in the paper with other eight tasks. Based on the JS-divergence metrics, the two tasks we demonstrate in the paper serve as good examples for covariates shifts and drastic change in the mechanism.

Table 1: Comparison for distribution shifts

DS (Text Ads Case)	10^{-2}
DS (Shopping Ads Case)	$5x10^{-3}$
Average DS (Others)	$45x10^{-5}$
STD of DS (Others)	$35x10^{-5}$

C. PROOF FOR THEOREMS

First, we give explicit description for the theorems in Section 3.3 which are from previous literature. The first theorem in [4] establishes the relationship between conditional invariant property and robust prediction

Theorem .1 (Adversarial robustness). *Suppose we have training data from various sources.* $\{(x_i^k, z_i^k, y_i^k)\} \sim \mathcal{P}^k, k = 1, 2, \cdots, K$ *and wish to make prediction on targeting* $\{(x_i^{K+1}, z_i^{K+1}, y_i^{K+1})\} \sim \mathcal{P}^{K+1}$. *Assume there exists a unique subset of features* S^* *such that:* $y_i^k|S_i^{*k} \overset{d}{=} y_i^{k'}|S_i^{*k'}, k \neq k' \in \{1, 2, \cdots K+1\}$ *(conditional invariant property). Then:*

$$E_{\mathcal{P}^{1,\cdots,K}}(y_i|S_i^*) = \text{argmin}_f \text{sup}_{(x_i,z_i,y_i)\sim\mathcal{P}} E||f(x_i, z_i), y_i||^2, \tag{4}$$

where $\mathcal{P}$ *is the family of distributions of* (x_i, z_i, y_i) *satisfying the invariant property.* $\mathcal{P}^{1,\cdots,K}$ *is the distribution pooling* $\mathcal{P}^1, \mathcal{P}^2 \cdots, \mathcal{P}^k$ *together.*

The second theorem from [3, 4] states relationship between conditional invariant property and causality.

Theorem .2 (Relationship to causality). *If we further assume* (x_i, z_i, y_i) *can be expressed with a direct acyclic graph (DAG) or structural equation model (SEM). Namely, let* $c_i = (x_i, z_i)$, $c_i^j = h_j(c_i^{\boldsymbol{PA}_j}, e_i^j)$, $y_i = h_y(c_i^{\boldsymbol{PA}_Y}, e_i)$. *Then we have* $S_i^* = c_i^{\boldsymbol{PA}_Y}$, *where* $c^{\boldsymbol{PA}_j}$ *denotes the parents for* c_j, $c^{\boldsymbol{PA}_Y}$ *denotes the parents for* y, e_i^j, e_i *are the noises,* $h_j(\cdot,\cdot)$ *and* $h_y(\cdot,\cdot)$ *are deterministic functions.*

Now we prove the theorem in the main text to validate the use of **R-data**, *Proof:* Assuming certain regularity conditions, such as the integrals are well-defined, suppose the model trained can converge to the conditional mean,

$$E(y_i|x_i, z_i) \quad \rightarrow_p \quad \int_{\mathcal{Y}} yp(y|x, z)dy = \int_{\mathcal{Y}} y\frac{p(y, x, z)}{p(x, z)}dy \tag{5}$$

Furthermore, under randomization conditions, we have,

$$\int_{\mathcal{Y}} y \frac{p(y,x,z)}{p(x,z)} dy = \int_{\mathcal{Y}} y \frac{p(y,x,z)}{p(x_i^1)p(x_i^2 \cdots p(x_i^p)p(z_i^1) \cdots p(z_i^{p'}))} dy \quad (6)$$

$$= \int_{\mathcal{Y}} y \frac{p(y|c_i^{PA_Y})p(x_i^1)p(x_i^2) \cdots p(x_i^p)p(z_i^1) \cdots p(z_i^{p'})}{p(x_i^1)p(x_i^2) \cdots p(x_i^p)p(z_i^1) \cdots p(z_i^{p'})} dy \quad (7)$$

$$= \int_{\mathcal{Y}} y \frac{p(y(\text{do}(c_i^{PA^Y}))p(x_i^1)p(x_i^2) \cdots p(x_i^p)p(z_i^1) \cdots p(z_i^{p'})}{p(x_i^1)p(x_i^2) \cdots p(x_i^p)p(z_i^1) \cdots p(z_i^{p'})} dy \quad (8)$$

$$= E(y_i|c_i^{PA_Y}) = E(y_i|S_i^*) \quad (9)$$

REFERENCES

[1] Murat Ali Bayir, Mingsen Xu, Yaojia Zhu, and Yifan Shi. 2019. Genie: An Open Box Counterfactual Policy Estimator for Optimizing Sponsored Search Marketplace. In *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining*. ACM, 465–473.

[2] Steffen Bickel, Michael Brückner, and Tobias Scheffer. 2009. Discriminative learning under covariate shift. *Journal of Machine Learning Research* 10, Sep (2009), 2137–2155.

[3] Jonas Peters, Peter Bühlmann, and Nicolai Meinshausen. 2016. Causal inference by using invariant prediction: identification and confidence intervals. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 78, 5 (2016), 947–1012.

[4] Mateo Rojas-Carulla, Bernhard Schölkopf, Richard Turner, and Jonas Peters. 2018. Invariant models for causal transfer learning. *The Journal of Machine Learning Research* 19, 1 (2018), 1309–1342.