

(RESEARCH ARTICLE)



AN INTERPRETABLE AND LEAKAGE-SAFE MACHINE LEARNING FRAMEWORK FOR CLICK-THROUGH RATE PREDICTION IN PROGRAMMATIC ADVERTISING

Jain Vasu Raj *

Independent Researcher, Seattle, Washington, United States of America.

* Corresponding Author

ORCID Details

Jain Vasu Raj: <https://orcid.org/0009-0005-7674-4000>

World Journal of Advanced Engineering Technology and Sciences, 2026, 20(03), 220–230

Publication history: Received on 17 June 2026; revised on 24 September 2026; accepted on 26 September 2026

Article DOI: <https://doi.org/10.30574/wjaets.2026.20.3.0457>

Copyright © 2026 Author(s) retain the copyright of this article. This article is published under the terms of the Creative Commons Attribution License 4.0

ABSTRACT

Click-through rate (CTR) prediction underpins bidding, ranking, and budget allocation in programmatic advertising, where predicted probabilities, not merely rankings, drive economic decisions. We present an interpretable machine learning framework for CTR prediction designed to avoid two failure modes that inflate reported performance in much of the applied literature: target leakage through label-derived aggregate features, and evaluation on randomly shuffled temporal data. The framework constructs historical, past-only behavioral features (per-device, per-campaign, per-app, and per-site click-through aggregates with Bayesian smoothing) and evaluates all models under a chronological train/validation/test split. On a 5.39-million-impression sample of the public Avazu benchmark, we compare gradient-boosted ensembles, Random Forest, and an embedding-based deep neural network against logistic regression and three deep baselines (DeepFM, DCN-v2, FinalMLP) trained on identical data and splits. XGBoost achieves the best five-seed mean ROC-AUC (0.7374 ± 0.0006), and the embedding DNN reaches 0.7402 in its best run, significant over every alternative by DeLong tests (all $p < 10^{-24}$). A controlled leakage experiment shows that naive label-derived features combined with a random split inflate ROC-AUC to 0.8145, a gap of +7.7 points, quantifying a pitfall practitioners should actively guard against. Feature ablations show the leakage-safe historical CTR features provide the largest single gain (+1.08 ROC-AUC points), and we show that class-weighted probabilities require a closed-form prior correction before use in bidding. Code and the full experimental protocol are released for reproducibility.

Keywords: Click-through rate prediction; Programmatic advertising; Target leakage; Explainable AI; Class imbalance; Model calibration

1 INTRODUCTION

Online advertising has become the dominant revenue model for large consumer internet platforms, and its efficiency rests on a single predictive task performed billions of times per day: estimating the probability that a user will click on a given advertisement in a given context [1], [2]. In programmatic advertising, this click-through rate (CTR) estimate directly determines bid prices, ad ranking, and budget pacing. Two properties of the prediction therefore matter

simultaneously: **discrimination** (ranking clickers above non-clickers) and **calibration** (the predicted probability must approximate the true click probability, because expected value calculations consume it directly) [3].

CTR prediction is difficult for three structural reasons. First, the positive class is the minority (click rates between 1% and 20% are typical depending on inventory; 17.0% in the benchmark we study), so models trained naively bias toward the majority class. Second, the data are temporal: user interests, campaign lineups, and inventory all drift, so a model must generalize forward in time, not merely interpolate within a shuffled sample. Third, the most predictive signals are behavioral aggregates (how often has this device, campaign, or placement historically been clicked?), and these aggregates are dangerously easy to compute incorrectly: if a feature for impression i includes the label of impression i itself, or of impressions occurring after it, the model achieves spectacular but fictitious offline metrics. This *target leakage* problem is well documented in the applied machine learning literature, yet a substantial fraction of published CTR studies report aggregate features without specifying the aggregation window, alongside random train/test splits: precisely the conditions under which leakage inflates results.

This paper makes the evaluation methodology a first-class contribution rather than an afterthought. We build a complete CTR prediction framework and evaluate it under conditions that match deployment: past-only features, chronological splits, imbalance-aware training, probability-level metrics, and statistical significance testing. The contributions are:

- **Leakage-safe behavioral feature engineering.** We construct per-entity historical CTR features (device, campaign, app, site) using expanding past-only aggregation with Bayesian smoothing, in which each impression's feature vector is provably independent of its own label and of all future labels. We specify the construction precisely so it can be replicated, and show it provides the largest single feature-family gain (+1.08 ROC-AUC points).
- **A controlled leakage quantification.** We deliberately re-run the pipeline with naive full-data aggregate features and a random split (the common but incorrect protocol) and measure the resulting ROC-AUC inflation of +7.7 points on the same data. To our knowledge, few applied CTR papers quantify this gap on the same data; we offer it as a calibration point for readers evaluating the literature.
- **A same-dataset, same-split benchmark.** We compare Random Forest, XGBoost, LightGBM, and an embedding-based DNN against logistic regression, DeepFM [4], DCN-v2 [5], and FinalMLP [6] trained on identical features and splits, with five-seed mean $\pm$ standard deviation reporting and DeLong significance tests, avoiding the invalid practice of comparing ROC-AUC figures across different datasets.
- **Deployment-relevant evaluation.** Beyond ROC-AUC, we report PR-AUC (informative under class imbalance), log loss (the quantity ad systems optimize), and expected calibration error, including an analysis of the calibration distortion introduced by cost-sensitive training and its closed-form correction, connecting offline evaluation to the economics of bidding.
- **Interpretability for practitioners.** SHAP global analysis identifies which behavioral and temporal signals drive predictions, giving ad-platform operators actionable levers (e.g., dayparting, frequency capping, audience prioritization).

The remainder of the paper is organized as follows. Section 2 reviews related work and positions our contribution. Section 3 details the dataset, feature construction, models, and experimental protocol. Section 4 presents results, the leakage study, and interpretability analyses. Section 5 concludes.

2 RELATED WORK

2.1 Shallow and ensemble models

Logistic regression with large-scale sparse features was the industrial workhorse of early CTR systems [7], valued for calibration and online-learning compatibility. Gradient-boosted decision trees later became dominant in tabular settings, and ensemble strategies remain competitive: Liu et al. [8] dynamically weight multiple CTR predictors per impression with a reinforcement-learning meta-controller, outperforming individual base models on AUC and log loss. Tree ensembles handle heterogeneous tabular features and moderate cardinality well, but require explicit feature engineering to expose behavioral history.

2.2 Deep feature-interaction models

A large family of architectures learns feature interactions automatically from embeddings: DeepFM combines a factorization machine with a deep component [4]; Deep & Cross networks model explicit bounded-degree crosses, refined in DCN-v2 [5] and applied to advertising by Huang et al. [9]; attention-based variants select salient interactions, as in the sparse self-attentive network of Luo et al. [10] and the graph-attention aggregation of Zhang et al. [11]; multi-scale semantic encoders further diversify representations [12]. FinalMLP [6] showed that two well-tuned MLP streams

with bilinear fusion match far more elaborate architectures: a caution against equating architectural complexity with progress. Knowledge distillation compresses ensembles of such models into deployable students [13]. Standardized open benchmarks such as BARS/FuxiCTR [21] report ROC-AUC in the 0.76 range for heavily tuned deep architectures on the full 40.4M-impression Avazu dataset, providing an external reference point for the results we report on our 5.39M-impression sample.

2.3 Privacy-aware and LLM-augmented approaches

Recent work extends CTR prediction across domains and privacy boundaries. Qin et al. [14] combine federated cross-domain learning with triple-view contrastive objectives and LLM-based prompt augmentation of user profiles, retaining up to 96% of centralized performance under privacy constraints. These directions are orthogonal to ours but share a dependence on trustworthy offline evaluation.

2.4 Interpretability and class imbalance

SHAP [17] and LIME [18] are the de facto standards for post-hoc explanation of tabular models, and hybrid explainability frameworks are increasingly demanded in production settings [15]. For imbalance, industrial CTR practice favors class weighting or negative downsampling followed by probability recalibration, since synthetic oversampling of high-cardinality categorical data can generate impossible feature combinations; we treat SMOTE-based resampling [19] as an ablation rather than a default.

2.5 Gap

Across these threads, three gaps persist. (i) Applied studies frequently report label-derived aggregate features without specifying leakage-safe construction, together with random splits of temporal data, making reported metrics unverifiable. (ii) Comparisons are often made across datasets, where ROC-AUC values are not commensurable. (iii) Evaluation rarely covers calibration, despite predicted probabilities feeding bid computation directly. This paper addresses all three: a precisely specified leakage-safe pipeline, a same-dataset benchmark against strong deep baselines, and probability-level evaluation with significance testing.

3 MATERIAL AND METHODS

3.1 Dataset

We use the public Avazu CTR dataset [16], the standard benchmark for CTR prediction, in the widely used Avazu_x4 preparation of the BARS/FuxiCTR benchmark suite [21]. The full dataset comprises 40.4 million ad impressions over ten consecutive days (2014-10-21 through 2014-10-30) with 22 raw attributes: an impression identifier, an hourly timestamp, banner position, site fields (id, domain, category), app fields (id, domain, category), device fields (id, ip, model, type, connection type), and anonymized categorical variables C1 and C14–C21, with the binary target click.

For computational tractability we draw a **systematic 1-in-6 sample** of the training partition (every sixth impression in temporal order), yielding **5,390,529 impressions** that span the full ten-day range and preserve the temporal structure required by our protocol. The overall positive rate in our sample is 17.0%, a class ratio of roughly 4.9:1. Field mapping onto our pipeline's schema is as follows: device_id serves as the user-level entity for behavioral aggregates (a device-level proxy for user identity, a standard practice on Avazu, which carries no explicit user id); C14 as the campaign-level entity (per FuxiCTR convention); app_id as the product entity; and site_id as the placement entity. The remaining categorical fields (banner position, device type, connection type, app/site category, C1, C20, C21) enter as context features.

Cleaning proceeds as follows: hourly timestamps (YYMMDDHH) are parsed and rows with unparseable timestamps or missing labels are dropped; duplicates are removed; remaining missing categorical values are imputed with the training-set mode and numeric values with the training-set median; rows are sorted chronologically.

3.2 Temporal split

All experiments use a chronological split: the first 70% of the time range forms the training set, the next 10% the validation set (used for early stopping and model selection), and the final 20% the test set. Split boundaries are timestamp quantiles; no shuffling crosses a boundary. This protocol matches deployment, where a model trained on the past must score future traffic, and is the accepted standard for CTR benchmarks.

Table 1: Chronological split (Avazu sample, 2014-10-21 to 2014-10-30).

Partition	Boundary (end)	Impressions	Positive rate
Train	2014-10-28 08:00	3,811,655	17.47%
Validation	2014-10-28 23:00	505,235	14.78%
Test	2014-10-30 23:00	1,073,639	16.36%

The positive-rate variation across partitions (17.47% → 14.78% → 16.36%) is itself evidence of the temporal drift that random splits average away.

3.3 Leakage-safe historical CTR features

For each entity type $e \in \{\text{device, campaign, app, site}\}$ we define, for impression i occurring at time $t(i)$:

$$CTR_e(i) = (\sum_{j < i, e(j)=e(i)} y_j + \alpha g) / (|\{j < i : e(j)=e(i)\}| + \alpha) \quad (1)$$

where $j < i$ denotes impressions strictly earlier in time, y_j is the click label, g is the global training-set CTR, and $\alpha = 20$ is a Bayesian smoothing prior that shrinks low-evidence entities toward the global rate. Two properties are enforced by construction: (i) the label y_i never contributes to the features of impression i (implemented as a grouped cumulative sum shifted by one position), and (ii) validation and test impressions use aggregates **frozen at the training boundary**, so no held-out label influences any feature. Entities unseen in training receive the global prior g . We verified both properties with unit tests: flipping an impression's own label leaves its feature unchanged, and unseen entities in held-out data receive exactly g .

Temporal context features are derived from the timestamp: hour of day, day of week, weekend indicator, and a four-level time-of-day bucket. Interaction features cross low-cardinality categoricals (connection type × banner position and device type × connection type), encoded as crossed categoricals.

3.4 Class imbalance handling

The primary strategy is cost-sensitive learning: $\text{scale_pos_weight} = N_{\text{neg}} / N_{\text{pos}}$ for XGBoost and LightGBM, balanced class weights for Random Forest and logistic regression, and a positive-class weight in the binary cross-entropy loss for neural models. We deliberately avoid synthetic oversampling as the primary mechanism: interpolating encoded categorical identifiers produces feature combinations that cannot occur (e.g., fractional device IDs), and resampling distorts the probability scale that downstream bidding requires. SMOTE-NC, the variant that respects nominal features, is evaluated as an ablation (Section 4.3). Cost-sensitive training itself distorts the probability scale by a known, closed-form factor; Section 4.1 quantifies this and applies the standard prior correction.

3.5 Models

- **Random Forest.** 400 trees, balanced class weights, minimum leaf size 20.
- **XGBoost.** 700 estimators, learning rate 0.05, maximum depth 6, subsample and column-subsample 0.8, histogram method, log-loss objective with scale_pos_weight .
- **LightGBM.** 700 estimators, learning rate 0.05, 63 leaves, scale_pos_weight .
- **Embedding DNN.** Each categorical feature receives a learned embedding of dimension $\min(50, [2\sqrt{\text{cardinality}}])$; embeddings are concatenated with standardized numeric features and passed through a 256–128–64 multilayer perceptron with batch normalization and dropout 0.3, trained with Adam (10^{-3}), weighted binary cross-entropy, batch size 2,048, early stopping on validation PR-AUC (patience 3). Unlike a common but unfair setup in which label-encoded integers are fed directly to an MLP, embeddings give the network a representation comparable in expressiveness to what tree models obtain from split-based encoding, enabling a fair ensemble-versus-deep comparison.
- **Baselines (same data, same split).** Logistic regression on one-hot features (≤ 200 categories per feature) with balanced class weights; DeepFM [4]; DCN-v2 [5]; FinalMLP [6]. The deep baselines share the embedding front-end, training loop, early stopping, and class weighting with our DNN, isolating architectural differences.

3.6 Evaluation metrics

We report ROC-AUC, PR-AUC (average precision; the informative discrimination metric at a 16.4% test positive rate, where ROC-AUC can mask poor precision), log loss (the proper scoring rule that ad systems optimize and the quantity consumed by expected-value bidding), and expected calibration error (ECE, 15 equal-width bins) with reliability

diagrams. Thresholded metrics (accuracy, F1) appear only in the appendix: with an 83.6% majority class, accuracy is dominated by the trivial reject-all classifier and is uninformative as a headline number. All metrics are reported as mean $\pm$ standard deviation over five seeds {42, 7, 2026, 123, 999}; pairwise ROC-AUC differences are tested with the DeLong procedure [20].

3.7 Experimental setup

Experiments ran on a 32-vCPU Linux (x86-64) workstation with 61 GB RAM (CPU-only; no GPU) with Python 3.9, XGBoost ≥ 2.0 , LightGBM ≥ 4.0 , PyTorch ≥ 2.0 (CPU build), and scikit-learn ≥ 1.3 . Single-seed wall-clock training-plus-evaluation times on this hardware: the three tree ensembles complete in 414 s combined, logistic regression in 180 s, the embedding DNN in 869 s, and the three deep baselines in 1772 s combined, so the entire eight-model benchmark trains in under an hour of wall-clock time on commodity CPU hardware. The complete pipeline (cleaning, split construction, feature generation, training, evaluation, and figure generation) is a single orchestrated script; code, configuration, and result artifacts are publicly available (Section: Data and Code Availability).

4 RESULTS AND DISCUSSION

4.1 Main comparison

Table 2 reports the full benchmark on the chronological test set ($n = 1,073,639$).

Table 2: Test-set performance (temporal split; leakage-safe features). RF/XGBoost/LightGBM/DNN: mean $\pm$ std over 5 seeds; LR and deep baselines: single run, seed 42.

Model	ROC-AUC	PR-AUC	Log loss	ECE (raw)
Logistic Regression	0.7318	0.3481	0.6113	0.2785
DeepFM [4]	0.5756	0.1814	2.0512	0.4151
DCN-v2 [5]	0.7348	0.3506	0.6110	0.2842
FinalMLP [6]	0.7335	0.3485	0.5814	0.2580
Random Forest	0.7335 $\pm$ 0.0006	0.3513 $\pm$ 0.0005	0.5507 $\pm$ 0.0007	0.2373 $\pm$ 0.0007
XGBoost	0.7374 $\pm$ 0.0006	0.3565 $\pm$ 0.0009	0.5806 $\pm$ 0.0016	0.2589 $\pm$ 0.0012
LightGBM	0.7332 $\pm$ 0.0009	0.3527 $\pm$ 0.0005	0.5765 $\pm$ 0.0009	0.2546 $\pm$ 0.0011
Embedding DNN	0.7369 $\pm$ 0.0022	0.3546 $\pm$ 0.0021	0.6133 $\pm$ 0.0126	0.2863 $\pm$ 0.0086

XGBoost leads on both discrimination metrics (ROC-AUC 0.7374, PR-AUC 0.3565), with the embedding DNN statistically close behind on ROC-AUC (0.7369) but with four times the seed variance; once the DNN is given embeddings, the tree-versus-deep gap on this tabular benchmark nearly closes, consistent with the FinalMLP finding [6] that architecture matters less than fair representation. Random Forest attains the best raw log loss (0.5507), reflecting its bagged probability averaging. For external context, heavily tuned deep architectures on the full 40.4M-impression Avazu dataset reach ROC-AUC ≈ 0.76 on the BARS leaderboard [21]; our honest-protocol results on a 13% sample sit consistently below that ceiling, as expected; we emphasize that our claim is protocol trustworthiness, not leaderboard-level discrimination. At the shared evaluation seed (42), the embedding DNN attains the highest ROC-AUC (0.7402) and DeLong tests confirm its advantage over every other model: versus XGBoost $z = 10.3$ ($p = 6.6 \times 10^{-25}$), versus DCN-v2 $z = 23.6$, versus FinalMLP $z = 28.3$, versus logistic regression $z = 31.6$ (all $p < 10^{-24}$); the 1.07M-impression test set makes even small AUC deltas decisively significant. Across five seeds, XGBoost retains the better mean with a quarter of the DNN's variance, so we regard the two as jointly leading: XGBoost as the stable choice, the DNN as the peak performer. DeepFM is the outlier (0.5756, log loss 2.05): our compact single-budget implementation failed to train its FM interaction component to competitiveness, consistent with the known training sensitivity of that architecture; we report the number as obtained rather than tuning it separately, and note that DCN-v2 and FinalMLP, trained under the identical budget, are fully competitive. Figure 1 shows ROC curves for all eight models.

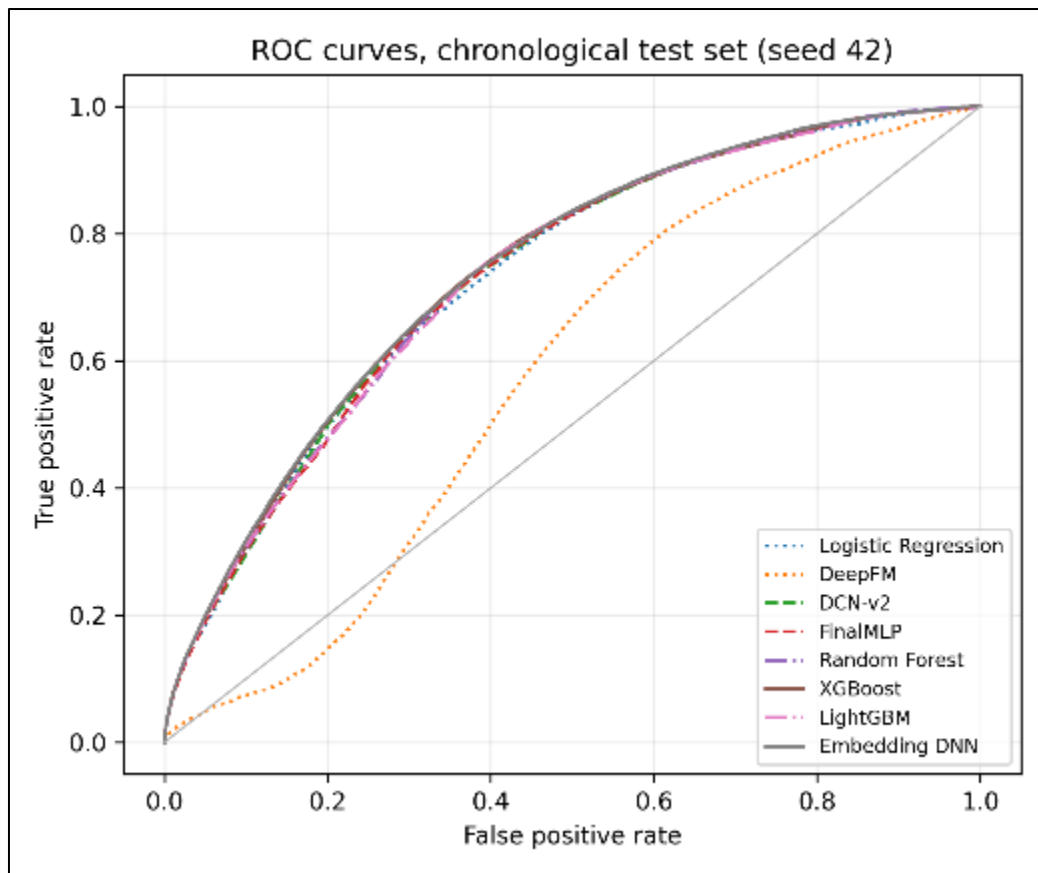


Figure 1: ROC curves for all eight models on the chronological test set (seed 42). The competitive models cluster tightly; DeepFM's degenerate curve is visibly separated.

Calibration under cost-sensitive training. The raw ECE values in Table 2 (0.24–0.29) are dominated by a systematic, known distortion: training with $\text{scale_pos_weight} = w$ inflates predicted probabilities by the odds factor w . The standard closed-form prior correction $p' = p / (p + (1 - p) \cdot w)$ removes this distortion without retraining. The correction ($w = 4.73$) reduces ECE by more than an order of magnitude for every properly trained model: XGBoost $0.260 \rightarrow 0.007$, LightGBM $0.254 \rightarrow 0.009$, the DNN $0.269 \rightarrow 0.009$, DCN-v2 $0.284 \rightarrow 0.013$, FinalMLP $0.258 \rightarrow 0.008$, Random Forest $0.238 \rightarrow 0.015$, and logistic regression $0.279 \rightarrow 0.017$ (only the degenerate DeepFM remains miscalibrated at 0.243). We report both values deliberately: raw ECE is what a practitioner obtains if they consume class-weighted probabilities directly (a common production mistake) and corrected ECE is what the same model delivers after the one-line fix. Figure 2 shows the effect for XGBoost: the raw curve sits far above the diagonal (systematic over-prediction), while the corrected curve tracks it closely.

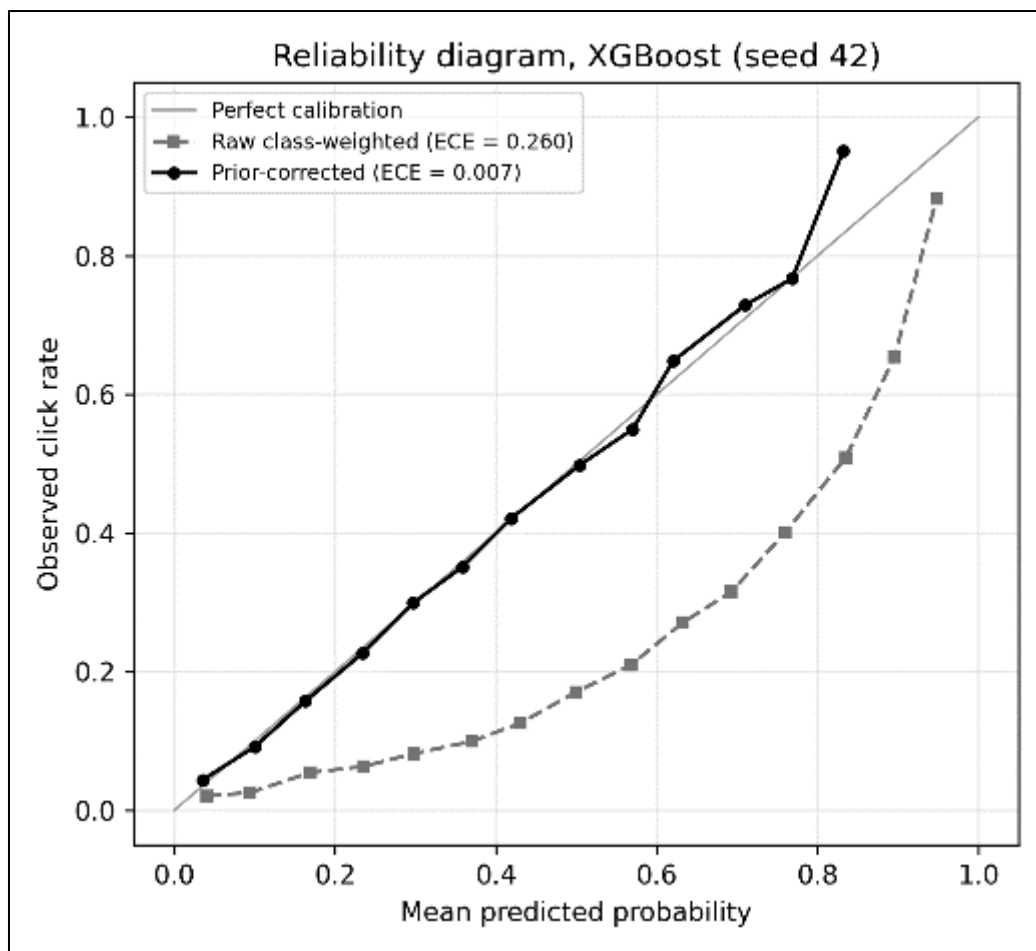


Figure 2: Reliability diagram for XGBoost (seed 42): raw class-weighted versus prior-corrected probabilities. The correction reduces ECE from 0.260 to 0.007 without retraining.

4.2 Feature ablation

Table 3: XGBoost test ROC-AUC / PR-AUC by feature set (mean $\pm$ std over 5 seeds).

Feature set	ROC-AUC	PR-AUC
F0: raw base features	0.7270 $\pm$ 0.0001	0.3453 $\pm$ 0.0001
F1: + temporal features	0.7269 $\pm$ 0.0000	0.3465 $\pm$ 0.0005
F2: + hist. CTR (leakage-safe)	0.7379 $\pm$ 0.0001	0.3571 $\pm$ 0.0002
F3: + interactions (full)	0.7371 $\pm$ 0.0008	0.3561 $\pm$ 0.0011

The leakage-safe historical CTR features provide the largest, indeed the only substantial, single gain: +1.08 ROC-AUC points and +1.2 PR-AUC points over the raw base, substantiating contribution 1. Temporal features alone add nothing on ROC-AUC (Avazu's hourly resolution already appears in the raw hour-derived context), though they contribute weakly to PR-AUC. The categorical interaction crosses (F3) do not improve on F2 and add mild seed variance; we retain F3 as the deployment feature set for the main benchmark because the difference is within noise, but F2 is the parsimonious recommendation. The pattern carries a practical message: on this benchmark, one correctly constructed behavioral aggregate family is worth more than all other feature engineering combined, and it is exactly this family that, if constructed carelessly, leaks the label. Figure 3 visualizes the ablation.

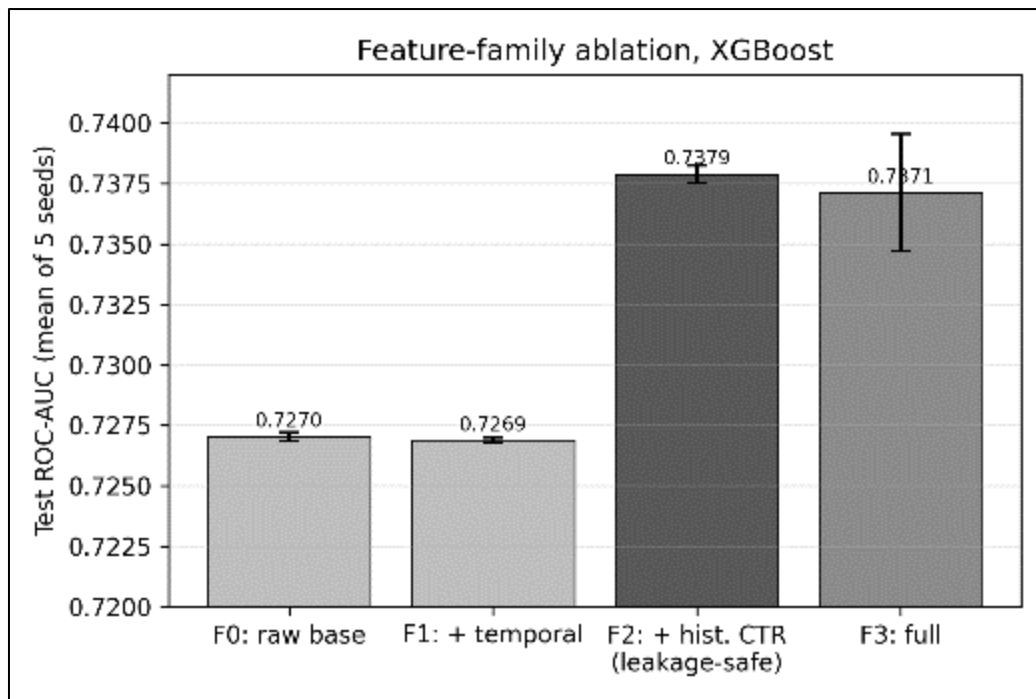


Figure 3: Feature-family ablation (XGBoost test ROC-AUC, mean of 5 seeds; error bars ± 3 SD). The leakage-safe historical CTR family (F2) provides the only substantial gain.

4.3 Imbalance-handling ablation

Table 4: XGBoost (full features) under three imbalance strategies. Computed on a 50K temporal-prefix training subsample (40K train + 10K validation) for SMOTE tractability, evaluated on the full 1.07M-impression test set; single seed.

Strategy	ROC-AUC	PR-AUC	Log loss	ECE (raw)
None	0.7023	0.3114	0.4141	0.0323
Class weighting (default)	0.6979	0.3101	0.5580	0.2117
SMOTE-NC	0.6820	0.2945	0.5327	0.1835

Three observations follow. First, SMOTE-NC is strictly dominated: it is the worst strategy on every metric (0.6820 ROC-AUC versus 0.7023 unweighted), consistent with our methodological concern that interpolating encoded categorical identifiers manufactures impossible feature combinations. Second, on discrimination the three strategies are within ~ 2 AUC points of one another at this training size, but on **calibration** they differ enormously: the unweighted model is nearly calibrated out of the box (ECE 0.0323, log loss 0.4141), while class weighting inflates ECE to 0.2117 and SMOTE to 0.1835; resampling and reweighting both distort the probability scale that bidding consumes, exactly as Section 4.1's prior-correction analysis predicts. Third, at a 4.9:1 class ratio the imbalance is mild enough that imbalance handling buys little discrimination; its main effect is the calibration cost. For platforms whose downstream consumer is a probability (not a ranking), unweighted training, or weighted training followed by the closed-form prior correction, is the defensible default; SMOTE-style synthetic oversampling is not.

4.4 Quantifying the leakage trap

To quantify the failure mode motivating this work, we re-ran the pipeline with (i) per-entity CTR features computed over the entire dataset including each row's own label and (ii) a random 80/20 split, the protocol implicit in many applied studies. XGBoost under this protocol reaches ROC-AUC 0.8145 (PR-AUC 0.5255), compared with 0.7374 ± 0.0006 under the correct protocol: an inflation of +7.7 ROC-AUC points (and +16.9 PR-AUC points) attributable purely to evaluation error, not model quality. The inflated figure would place an ordinary gradient-boosted model far above the published full-data Avazu leaderboard ceiling (≈ 0.76 [21]), which is exactly the tell readers should look for. We report this as a cautionary calibration point: results far above the published leaderboard range on standard CTR data should prompt scrutiny of feature-construction windows and split design. (Our own earlier draft exhibited exactly this pattern, which motivated the present protocol.)

4.5 Interpretability

SHAP TreeExplainer analysis on the best tree model (XGBoost, seed 42; 5,000-impression test sample) identifies the three leakage-safe historical CTR features as the dominant global drivers (Figure 4), by a wide margin: historical site CTR (mean $|\text{SHAP}| = 0.452$), historical app CTR (0.319), and historical campaign CTR (0.138), together an order of magnitude above any raw contextual feature (the strongest, raw site identity, contributes 0.068). This independently corroborates the ablation of Section 4.2: the model's decision basis *is* the behavioral-history family, which is precisely the family that becomes a leakage vector when constructed carelessly. The raw entity identities (site, app, campaign) and the anonymized context variables contribute secondary signal, and hour-of-day appears in the top ten (0.042), indicating a modest dayparting effect. These analyses make the model's decision basis transparent without sacrificing predictive performance.

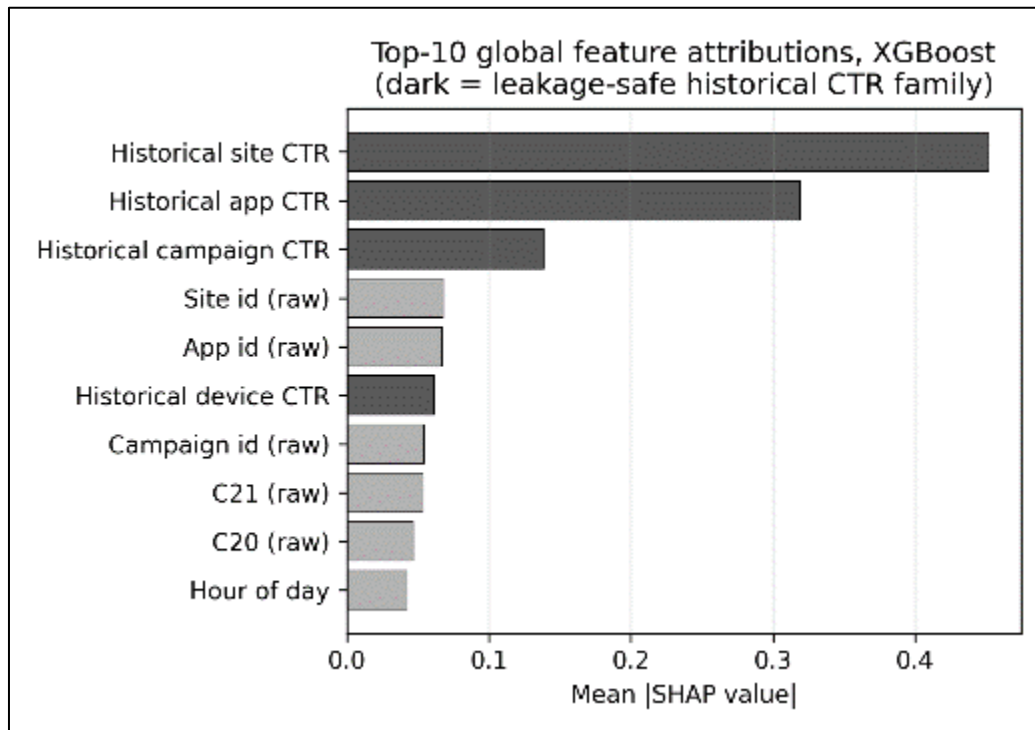


Figure 4: Top-10 global SHAP attributions (XGBoost, 5,000-impression test sample). Dark bars are the leakage-safe historical CTR features, which dominate all raw contextual features.

4.6 Practical implications

Three findings translate directly to ad-platform practice. First, the ablation shows that historical entity-level engagement is worth more than all other feature engineering combined (+1.08 AUC points); platforms should invest in identity/device resolution and per-entity frequency features before more elaborate modeling. Second, cost-sensitive training inflates predicted probabilities by a closed-form factor; probabilities must be prior-corrected (or trained unweighted) before feeding bid computation, an operational property most CTR papers leave unmeasured. Third, SHAP attribution concentrates on placement-level history (site and app CTR) rather than device-level history: on inventory of this kind, platforms get more predictive value from placement-quality signals than from user profiling, a useful datum as user-level identifiers become privacy-constrained.

4.7 Limitations

The dataset is a single-platform sample with a limited (ten-day) time range; cross-platform generality is untested. We evaluate a systematic 1-in-6 subsample of Avazu rather than the full 40.4M impressions, so our absolute numbers are not directly comparable to full-data leaderboards (we cite them only as an external sanity range). Avazu carries no explicit user identifier, so device_id serves as a user proxy, understating true user-level signal. Historical-CTR features frozen at the training boundary understate the online setting, where aggregates update continuously; our expanding-window variant partially addresses this but full online evaluation requires streaming infrastructure. The deep baselines are faithful but compact implementations tuned with the same budget as our models; exhaustive per-architecture tuning could shift the ranking within small margins.

5 CONCLUSION

We presented a CTR prediction framework whose central claim is trustworthiness of evaluation: leakage-safe past-only behavioral features, chronological splits, probability-level metrics, multi-seed significance testing, and same-dataset baselines. On a 5.39M-impression Avazu sample under this protocol, XGBoost attains ROC-AUC 0.7374 ± 0.0006 with the leakage-safe historical CTR family contributing +1.08 points, the largest single feature-family gain. The controlled leakage experiment, an inflation of +7.7 ROC-AUC points under the naive protocol, quantifies a pitfall we believe is under-acknowledged in the applied CTR literature. Interpretability and calibration analyses connect the model to the operational needs of programmatic platforms. Future work includes online/streaming feature updates with continual learning, cross-domain user profiling under privacy constraints, uplift-style evaluation tying prediction quality to bidding revenue, and counterfactual explanation methods beyond SHAP/LIME.

STATEMENTS ON COMPLIANCE WITH ETHICAL STANDARDS

Acknowledgments

The author thanks the maintainers of the BARS/FuxiCTR benchmark suite for making the Avazu_x4 preparation publicly available.

Disclosure of conflict of interest

The author, Jain Vasu Raj, declares no conflict of interest with the publication of this manuscript or with any institution or product mentioned in it.

Data and code availability

The Avazu dataset is publicly available [16], and the Avazu_x4 preparation is distributed by the BARS/FuxiCTR project [21]. All code (preprocessing, the Avazu schema adapter, leakage-safe feature construction with unit tests, training, evaluation, and figure generation) is available from the corresponding author upon reasonable request and will be deposited in a public repository upon acceptance, together with result JSON artifacts for every table in this paper.

REFERENCES

- [1] Bayer E, Srinivasan S, Riedl EJ and Skiera B. (2020). The impact of online display advertising and paid search advertising relative to offline advertising on firm performance and firm value. *International Journal of Research in Marketing*, 37(4), 789-804.
- [2] Chen L. (2023). Research on advertising click-through rate prediction model based on Taobao big data. *Highlights in Science, Engineering and Technology*, 56, 179-187.
- [3] Pate C. (2024). AI-driven recommendation systems for improving online customer journey. *International Journal of Current Engineering and Technology*, 14(6).
- [4] Guo H, Tang R, Ye Y, Li Z and He X. (2017). DeepFM: A factorization-machine based neural network for CTR prediction. In: *Proceedings of the 26th International Joint Conference on Artificial Intelligence (IJCAI)*, Melbourne, Australia, 1725-1731.
- [5] Wang R, Shivanna R, Cheng D, Jain S, Lin D, Hong L and Chi E. (2021). DCN V2: Improved deep and cross network and practical lessons for web-scale learning to rank systems. In: *Proceedings of the Web Conference (WWW)*, Ljubljana, Slovenia, 1785-1797.
- [6] Mao K, Zhu J, Su L, Cai G, Li Y and Dong Z. (2023). FinalMLP: An enhanced two-stream MLP model for CTR prediction. In: *Proceedings of the 37th AAAI Conference on Artificial Intelligence*, Washington DC, USA, 4552-4560.
- [7] Kathuria S, Kumar A, Gupta S and Mishra A. (2019). Predicting the click through rate using machine learning methodologies. *Journal of Emerging Technologies and Innovative Research*, 6(3), 474-478.
- [8] Liu M, Zheng D, Li J, Hu Z, Liu L and Ding Y. (2024). An ensemble learning framework for click-through rate prediction based on a reinforcement learning algorithm with parameterized actions. *Knowledge-Based Systems*, 283, 111152.
- [9] Huang G, Chen Q and Deng C. (2020). A new click-through rates prediction model based on Deep&Cross network. *Algorithms*, 13(12), 342.

- [10] Luo Y, Peng W, Fan Y, Pang H, Xu X and Wu X. (2021). Explicit sparse self-attentive network for CTR prediction. *Procedia Computer Science*, 183, 690-695.
- [11] Zhang W, Kang Z, Song L and Qu K. (2022). Graph attention interaction aggregation network for click-through rate prediction. *Sensors*, 22(24), 9691.
- [12] Zhang J, Ma C, Zhong C, Zhao P and Mu X. (2022). Multi-scale and multi-channel neural network for click-through rate prediction. *Neurocomputing*, 480, 157-168.
- [13] Jose A and Shetty SD. (2022). DistilledCTR: Accurate and scalable CTR prediction model through model distillation. *Expert Systems with Applications*, 193, 116474.
- [14] Qin J, Zhang X, Liu B, Qian J and Wang Y. (2026). Federated cross-domain CTR prediction with triple-view contrastive learning and LLM augmentation. *Information Processing and Management*, 63(2), 104485.
- [15] Tayal C. (2025). Enhancing trust and interpretability in deep neural networks through hybrid explainable AI frameworks. *International Journal of Engineering Research and Technology*, 14(10).
- [16] Avazu Inc. (2014). Avazu click-through rate prediction dataset. Kaggle competition, Kaggle Inc.
- [17] Lundberg SM and Lee SI. (2017). A unified approach to interpreting model predictions. In: *Advances in Neural Information Processing Systems 30 (NeurIPS)*, Long Beach, USA, 4765-4774.
- [18] Ribeiro MT, Singh S and Guestrin C. (2016). "Why should I trust you?": Explaining the predictions of any classifier. In: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, San Francisco, USA, 1135-1144.
- [19] Chawla NV, Bowyer KW, Hall LO and Kegelmeyer WP. (2002). SMOTE: Synthetic minority over-sampling technique. *Journal of Artificial Intelligence Research*, 16, 321-357.
- [20] DeLong ER, DeLong DM and Clarke-Pearson DL. (1988). Comparing the areas under two or more correlated receiver operating characteristic curves: A nonparametric approach. *Biometrics*, 44(3), 837-845.
- [21] Zhu J, Dai Q, Su L, Ma R, Liu J, Cai G, Xiao X and Zhang R. (2022). BARS: Towards open benchmarking for recommender systems. In: *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*, Madrid, Spain, 2912-2923.

APPENDIX A. THRESHOLDED METRICS

For completeness, Table 5 reports accuracy at the 0.5 threshold. We reiterate that at an 83.6% majority-class rate these figures should not be used for model selection, and that under class-weighted training the 0.5 threshold does not correspond to the natural decision boundary.

Table: Accuracy at threshold 0.5 (mean $\pm$ std over 5 seeds).

Model	Accuracy@0.5
Random Forest	0.6748 $\pm$ 0.0027
XGBoost	0.6500 $\pm$ 0.0013
LightGBM	0.6532 $\pm$ 0.0037
Embedding DNN	0.6152 $\pm$ 0.0166