Note for Task B - Ranking strategy

Objective

The goal of the ranking strategy is to optimize the ordering of offers within each campaign by predicting their expected performance and selecting a placement that maximizes:
- Publisher revenue per impression
- Click-through rate

The system accounts for position bias, leverages historical behavioral data and introduces a controlled degree of randomness through epsilon exploration to avoid overly rigid or suboptimal deterministic ranking. A Python project [Link] has been set up to suite these tasks.

Inputs

Signal	Description	Source
Campaign	Encoded campaign identifier	Metadata
Adunit	Encoded adunit identifier	Metadata
Number clicks	Log-transformed click volume	Click logs
Number impressions	Log-transformed impression volume	Impression logs
Number events	Log-transformed events volume	Event logs
Total publisher revenue	Log-transformed publisher revenue	Revenue aggregation
Total user revenue	Log-transformed user revenue	Revenue aggregation

All count and revenue features use log1p transformation to reduce skew, stabilize variance and improve model behavior on long-tailed distributions.

Window-Based Forecasting Strategy

User behavior and monetization performance change over time: the ranking system applies a multi-window forecasting approach rather than relying on a single historical snapshot. For each recency window (e.g. 0 to 7, 8 to 14, 15 to 30 days and an older bucket), a separate model is trained using the same input signals. This produces window-specific predictions for both CTR and eRPM, later combined using a recency-weighted aggregation schema.

Debiasing for Position Effects

Position bias is not treated as a direct feature: the strategy explicitly removes the impact of historical position before modeling performance.

This stage consists of:
1. Estimating empirical CTR and eRPM curves per position, aggregating impressions, clicks, and revenues for each position
2. Computing position-bias multipliers for each position (for both CTR and eRPM): $CTR_{bias_{pos}} = \frac{CTR_{mean_{pos}}}{CTR_{mean_{global}}}$
3. Mapping these multipliers to each row, in order to retrie the estimated "inflation factor" caused by visibility
4. Constructing debiased targets (for both CTR and eRPM) as: $CTR_{debiased_{pos}} = \frac{CTR_{pos}}{CTR_{bias_{pos}}}$

This ensures the model learns intrinsic adunit quality instead of merely reproducing historical exposure effects. Safeguards against low-volume positions and extreme ratios have been adopted.

Model description

Each model predicts a debiased version of the metric of interest:
- CTR model predicts debiased CTR
- eRPM model predicts debiased eRPM

```
GradientBoostingRegressor(
    n_estimators=400,
    learning_rate=0.03,
    max_depth=3,
    subsample=0.8,
    random_state=42
)
```

The models use the previously described Inputs as features to forecast the position-debiased CTR and eRPM values computed through the bias-removal stage. Both targets are modeled in log-space using log1p to stabilize variance; predictions are then transformed back using expm1 to ensure non-negativity and preserve proportional differences. This setup balances stability and flexibility, enabling the model to capture non-linear relationships in the data while reducing overfitting through subsampling and shallow trees.

Ranking rules

1. **CTR for each Campaign and Adunit** $CTR_{expected} = \sum_w w_{weight} * CTR_{pred,w}$
2. **eRPM for each Campaign and Adunit** $eRPM_{expected} = \sum_w w_{weight} * eRPM_{pred,w}$
3. **Score for ranking** $Score_i = CTR_{expected_i} * eRPC_{expected_i}$
4. **Ranking** sorting descending by $Score_i$ and reserving small percentage of traffic for **exploration** (ϵ-greedy sampling)

Guardrails / Constraints

Constraint	Description
Exploration rate	Reserve a small percentage of traffic for ϵ-greedy randomized sampling

In principle, the ranking strategy should also enforce offer diversity, since limiting repeated exposure to highly similar ads generally improves user experience and prevents homogenization of the auction outcomes, but no ad-unit taxonomy or content categorization is available in the provided dataset

Handling Ties / Exploration

- **Near-equals**: Random shuffle among offers within a threshold of score.

- **Exploration**: ϵ-greedy strategy, following an exponential decay schedule probability ϵ (e.g. from 0.1 to 0.05) for a higher initial exploration that decreases over time, to pick a random non-top offer.

Example of final ranking

Position	Camp id	Adunit	CTR	eRPM	CTR Exp	eRPM Exp	Score	Prop Pos	Pos Shift
12	14	A201	0.028	3.95	0.031	4.20	0.0130	1	+11
3	07	A144	0.044	2.95	0.046	3.10	0.0143	1	+2
28	14	A311	0.022	4.88	0.024	5.10	0.0122	2	+26
7	21	A077	0.036	3.42	0.038	3.60	0.0137	1	+6

After Deployment Metrics Evaluation

Metric	Purpose / Interpretation
Weighted eRPM delta	Primary KPI — measures overall revenue impact, weighting each offer by impressions.
Average eRPM delta	Ensures uplift is consistent across campaigns, not driven only by high-volume ones.
Weighted CTR delta	Confirms engagement quality while accounting for traffic distribution.
Average CTR delta	Broad health check to ensure no systemic CTR degradation across campaigns.
Campaign-level deltas	Allows identification of campaigns performances in terms of CTR and eRPM.

Metric	Purpose / Interpretation
Average position shift	Indicates how aggressive the re-ranking is; large shifts may require monitoring.

It's worth mentioning that even in cases where the observed average uplift is neutral or slightly negative, the approach may still be considered valuable due to the introduction of controlled exploration, which enables long-term discovery of higher-performing offers that the historical dataset could not reveal. It is also important to note that the dataset used for prediction is restricted and does not include offer-level categorization, recommendation signals or rich campaign descriptors; therefore, the model optimizes only on the available behavioral and revenue metrics. As a consequence, results close to the existing baseline (i.e. small deltas and modest position shifts) can still be interpreted positively, as they indicate stability under limited input information and provide a foundation for further improvements once additional features become available.