Ad Show Optimization

Goal

Build a decision system that chooses **when, where, and how often** to show in-game ads to maximize **net business value** while controlling **UX harm** (churn, session shortening, complaints).

Near-term philosophy: no RL, no complex sequential architectures. Start with **causal uplift modeling from existing logs** + conservative deployment.

Key data reality: we do **not** have A/B test data with randomized ad exposure. We work from **observational logs** only, which means propensity-based debiasing (IPW) is essential from day one.

Definitions

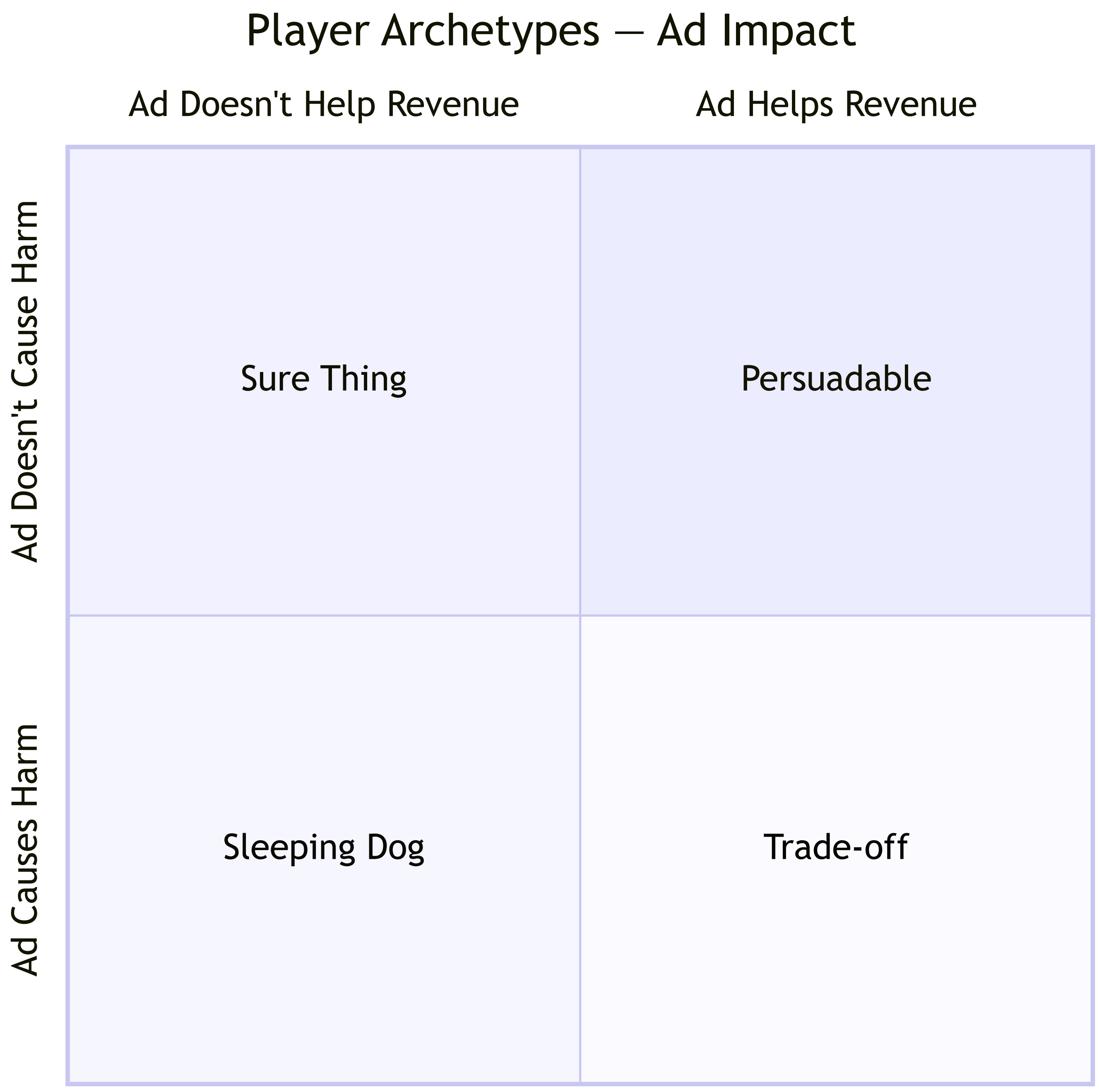
Term	Meaning
Ad opportunity	A moment where an ad *could* be shown (valid placement)
A	Treatment indicator: A=1 (ad shown), A=0 (ad not shown)
H	Harm horizon: time window for measuring churn (e.g., 7d, 30d)
Harm	Binary churn label: no session within H days after the impression
IPW	Inverse Propensity Weighting — reweighting to correct for non-random ad exposure

Non-goals (first iteration)

- No end-to-end RL / MDP planning.
- No Transformers / LSTMs until we see signal from simpler models.
- No survival / hazard modeling (optional upgrade later).
- No new data collection.

Player archetypes

Every player falls into one of four quadrants w.r.t. ad exposure:



Quadrant	Revenue impact	Harm impact	Action
Persuadable	Ad helps	No harm	**Show ads**
Sure Thing	No revenue gain	No harm	Ad is neutral — low priority
Trade-off	Ad helps	Causes harm	Careful cost/benefit via ΔEV
Sleeping Dog	No revenue gain	Causes harm	**Never show ads**

The model's primary job: **find Persuadables** and **protect Sleeping Dogs**.

The core intuition

At a single ad opportunity, estimate the **incremental expected value** of showing an ad now:

$$\Delta EV(x) = \underbrace{\Delta p_{\text{click}}(x) \cdot V_{\text{click}}(x)}_{\text{revenue gain}} - \underbrace{\Delta p_{\text{harm}}(x) \cdot V_{\text{harm}}(x)}_{\text{UX cost}}$$

Component	What it is	Source
$\Delta p_{\text{click}}(x)$	Incremental click probability	**Model estimates** (uplift)
$V_{\text{click}}(x)$	Value per click (varies by ad type, geo, user)	**External input** from ad-serving / finance
$\Delta p_{\text{harm}}(x)$	Incremental churn probability within horizon H	**Model estimates** (uplift)
$V_{\text{harm}}(x)$	Cost of losing this player (per-player, not constant)	**External input**: predicted LTV

Decision rule: show ad if $\Delta EV(x) > 0$ (or $> \theta$ threshold).

Toy numeric example

	Value
Δp_{click}	0.07
V_{click}	$20
Δp_{harm}	0.02
V_{harm}	$30

$$\Delta EV = 0.07 \times 20 - 0.02 \times 30 = 1.40 - 0.60 = +\$0.80 \quad \Rightarrow \text{show}$$

The model's job is to make Δp_{click} and Δp_{harm} **contextual** — functions of user, gameplay state, ad candidate, and history. The V terms come from external systems.

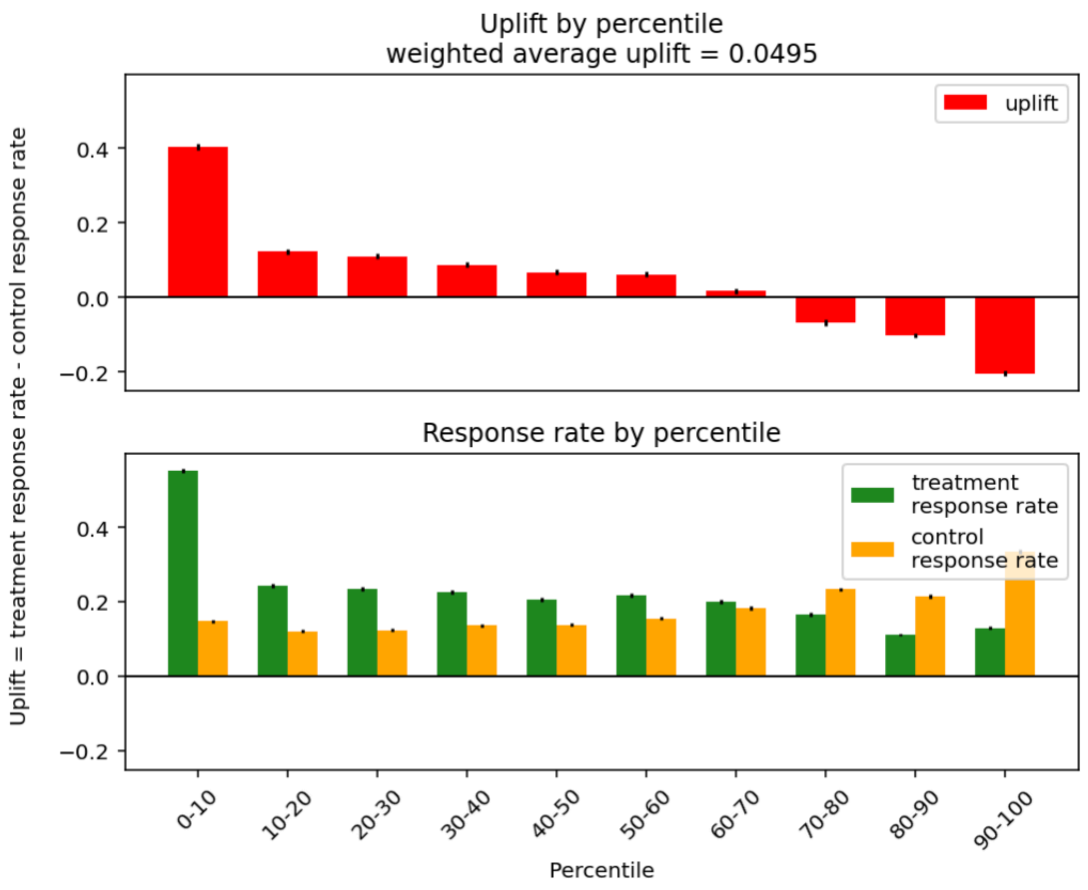
Why uplift modeling

We care about the **counterfactual difference**: what happens if we show an ad *now* vs. if we don't, in the same context.

That's a **treatment effect**:

$$\Delta p_{\text{harm}}(x) = P(\text{harm} \mid A{=}1, x) - P(\text{harm} \mid A{=}0, x)$$

Same structure for click. This is the standard **uplift modeling** framing — we estimate the *incremental* effect of treatment, not the absolute outcome.



(from https://www.uplift-modeling.com/en/latest/quick_start.html)

The counterfactual problem (why naive models are biased)

Ads are not shown randomly — the existing system targets specific moments and users.

Since we have no A/B data, the treated (A=1) and control (A=0) groups are not comparable. Without correction, any uplift estimate is biased. **IPW is our primary debiasing tool** (see appendix).

The dataset: defining ad opportunities

What is an "ad opportunity"

Create rows **only** for moments that are **eligible** for an ad:

- Valid placement exists
- Not in cutscene / critical gameplay
- Satisfies hard business/tech constraints

Why strict eligibility: including impossible moments creates $P(A{=}1|x) \approx 0$, causing IPW weights to explode.

Required columns per opportunity row

Column	Description
x_t	Features: user profile + gameplay context + ad candidate + session history + etc.
$A_t \in 0,1$	Ad shown (1) vs. not shown (0)
Y_t^{click}	Click within short attribution window (binary)

Y_t^{harm}	Churn within horizon H (binary)

Targets: start simple, upgrade later

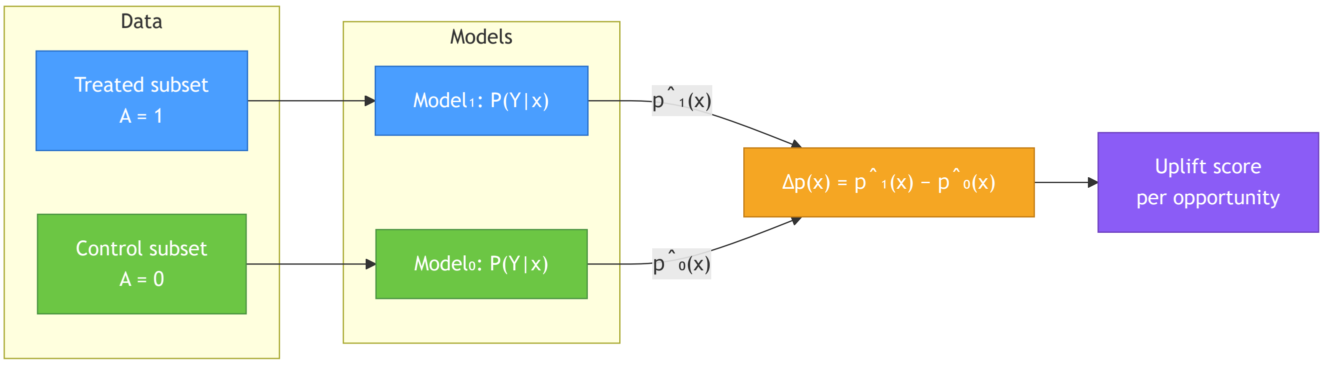
Click target

$Y_t^{\text{click}} = 1$ if click within short attribution window; else 0. Standard binary classification.

Harm target (v1: binary churn)

- Choose horizon H (e.g., 7d / 30d) — **business decision**.
- $Y_t^{\text{churn}} = 1$ if no session in next H days; else 0.
- Known weakness: user returning on day H+1 counts as churn.
- **Future upgrade:** survival/hazard model with censoring.

How to estimate uplift: T-learner



For each outcome (click, harm), train **two separate models** on treated and control subsets with IPW weights:

- **Treated model:** $\hat{p}_1(x) = \hat{P}(Y{=}1 \mid x)$ trained on A=1 rows (weighted by $1/\hat{e}(x)$)
- **Control model:** $\hat{p}_0(x) = \hat{P}(Y{=}1 \mid x)$ trained on A=0 rows (weighted by $1/(1{-}\hat{e}(x))$)
- **Uplift:** $\Delta p(x) = \hat{p}_1(x) - \hat{p}_0(x)$

Why T-learner over S-learner: S-learners (single model with A as a feature) can underestimate treatment effects — the treatment indicator gets regularized away, producing near-zero uplift. T-learners avoid this by construction.

Trade-off: T-learners don't share information between groups. With small treated samples, the treated model may be noisy. If so, X-learner is an upgrade path.

Calibration

Because we **subtract** probabilities to get uplift, calibration is critical:

- Apply (e.g., isotonic regression) on validation set
- Calibrate each model (treated, control) **before** computing uplift

Model choice: gradient boosting (XGBoost / CatBoost)

Model architecture is usually a smaller lever than:

1. Correct opportunity definition
2. Correct targets and horizons
3. Debiasing / weighting
4. Leakage-free validation

GBMs are fast to iterate, handle nonlinearity well, and minimize feature engineering. Sequential models are a later upgrade if signal warrants it.

Validation: time-based splits

Use **chronological** splits to avoid leakage:

- **Train:** older period
- **Validation:** middle period
- **Test:** most recent period

Why it matters:

- Hyperparameter selection favors features that generalize over time.
- Validation → test gap is an indicator of drift.

Offline evaluation

Standard predictive metrics

- AUC, logloss, calibration on held-out data
- Off-policy estimates using IPS / DR to estimate expected uplift under the proposed policy

Uplift-specific metrics

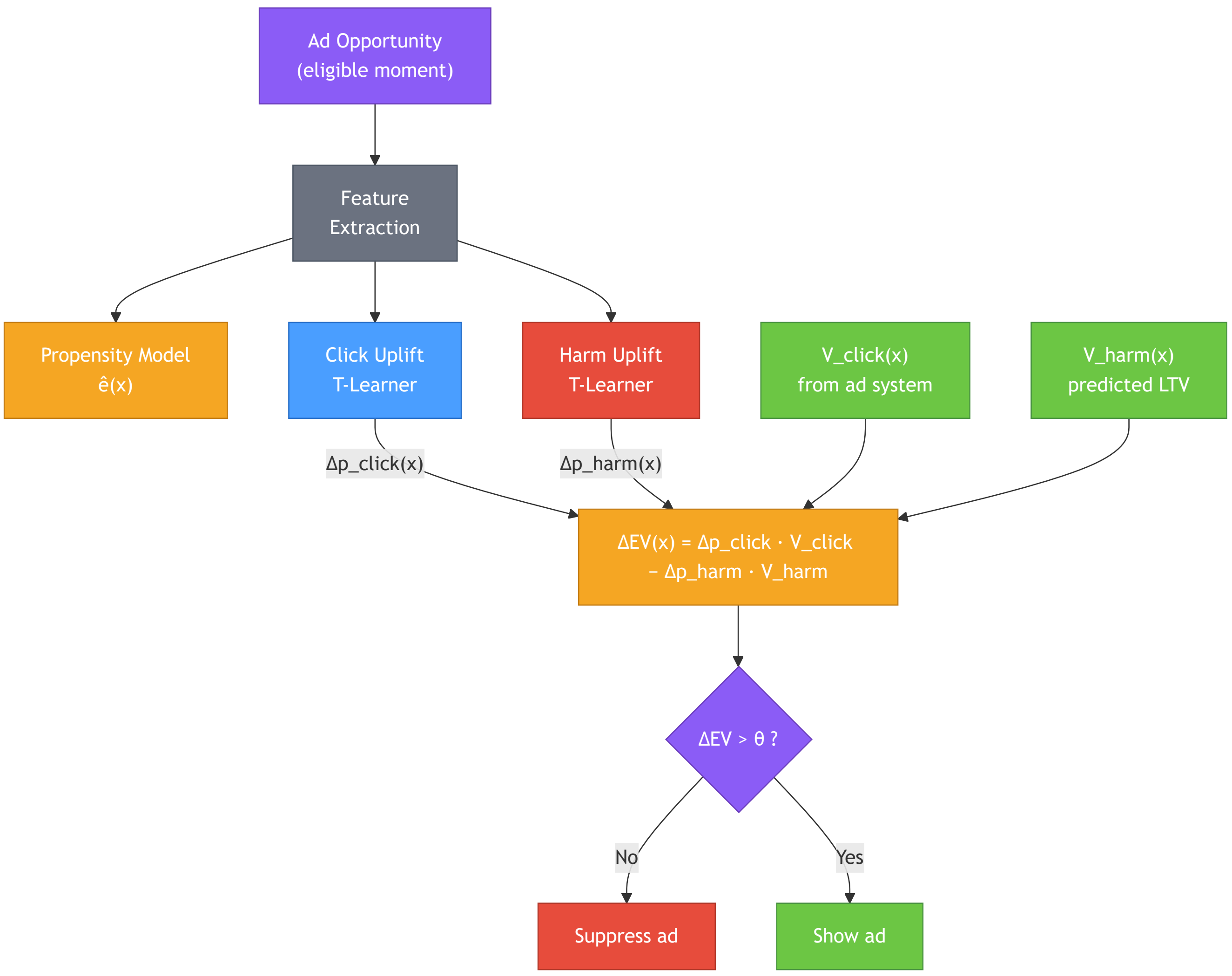
Standard AUC tells us if the outcome model is good — not whether the **uplift ranking** is correct. We need:

- **Qini curve / AUUC**: primary metric — cumulative uplift vs. fraction of population treated.
- **Uplift by decile**: actual treatment effect per bin — the business-facing proof.
- **Sleeping Dogs detection rate**: % of bottom-decile players correctly identified as harmed. The safety metric.
- **Uplift calibration**: predicted vs. actual uplift per bin.

What we cannot prove offline

Business impact with the same confidence as a randomized experiment. **Plan:** use offline eval to choose safe candidates, then run small guarded experiments.

Decision pipeline (end-to-end)



Rollout plan

Phase 0 — Data foundation

- Confirm "ad opportunity" definition is implementable from logs
- Ensure overlap: both A=0 and A=1 exist within eligible opportunities
- Choose horizon H for harm label
- Exteact target features (click, harm)
- Extract input features

Phase 1 — Baseline models

- Train propensity model $\hat{e}(x)$
  - Diagnose propensity score distribution — check for extreme $\hat{e}(x)$ regions
- Train T-learners (two models per outcome) for click and harm with IPW
- Calibrate both treated and control models
- Build $\Delta EV(x)$ scorer with external $V_{\text{click}}(x)$ and $V_{\text{harm}}(x)$
- Evaluate with uplift metrics (Qini, uplift by decile, Sleeping Dogs rate, etc.)

Phase 2 — Shadow mode

- Deploy scorer in production — **no decisions, logging only**
- Log what the model *would* have done alongside actual production decisions
- Compare recommendations vs. actual outcomes over

Phase 3 — Conservative guarded deployment

- Add hard caps (frequency limit, per-session max)
- Deploy to small holdout (e.g., 5%)
- Show ads if $\Delta EV > \theta$
- Monitor online metrics: retention, session length, complaints

Phase 4 — Improve fidelity

- Upgrade harm model to survival/hazard if needed
- Upgrade to X-learner or uplift trees if T-learner is noisy
- Consider direct revenue uplift $\tau_R(x)$ instead of $\Delta p_{\text{click}} \cdot V_{\text{click}}$
- Consider sequential encoders (e.g., Transformer) if event-level signal justifies complexity

Open questions

1. What **harm proxy** is acceptable for v1? (7d vs 30d churn, session length drop, etc.)
2. What **value inputs** are available? $V_{\text{click}}(x)$ from ad system, $V_{\text{harm}}(x)$ per player?
3. Do we have enough **overlap** (contexts where ads are sometimes shown and sometimes not)?
4. What **hard constraints** must always hold? (frequency caps, placement rules, safety floors)

Known limitations

- Uplift is a stepping stone, not the end state. Major ad-tech companies (TikTok, Unity) use RL for session-level ad timing. Our uplift approach is validated for observational-data-first starts, but high-frequency game ads will likely need policy optimization to capture session dynamics fully.
- Session-level state is implicit, not structural. FoRI features (ads-in-last-Xh, time-since-last-ad) partially capture session dynamics, but the T-learner still treats each opportunity as independent. If fatigue effects are strongly sequential, this is a thing to upgrade.
- `V_harm` is a weak input. Per-player predicted LTV (especially whale detection) is a hard modeling problem on its own and likely a big lever for improving decision quality.
- Propensity assumes one behavior policy. If the production ad-show logic changed over time (rule updates, A/B tests), the propensity model must be scoped to a stable policy window or account for multiple loggers.

Appendix - Propensity model + IPW

Propensity model

Train a model to predict the production system's ad-show behavior:

$$\hat{e}(x) = P(A{=}1 \mid x)$$

Why reweighting is needed

If the current system shows ads mostly in "safe" contexts, then naive comparison makes ads look safer than they are. IPW upweights rare decisions (e.g., showing an ad where the system usually doesn't) to approximate randomized exposure.

IPW weights

Condition	Weight
A = 1 (ad shown)	$w = 1/\hat{e}(x)$
A = 0 (ad not shown)	$w = 1/(1 - \hat{e}(x))$

> **Assumption — Overlap (positivity):** for similar contexts x, we need both A=1 and A=0 observations. Regions where $\hat{e}(x) \approx 0$ or ≈ 1 have unidentifiable treatment effects.

> **Risk — Variance:** extreme propensity scores cause weights to blow up, making estimates noisy. This is a fundamental cost of observational data vs. A/B.
>
> **Mitigations:**
>
> - Clip $\hat{e}(x)$ to $[\epsilon, 1{-}\epsilon]$ (e.g., $\epsilon = 0.01$)
> - Cap maximum weight w
> - Use self-normalized weights (SNIPS) for reporting
> - Run overlap diagnostics before training uplift models

Doubly robust estimator (upgrade path)

DR combines an outcome model with propensity weighting — consistent if **either** model is correct:

$$\hat{Y}^{DR}(x_i, W{=}t) = \hat{Y}(x_i, W{=}t) + \frac{Y_i - \hat{Y}(x_i, W{=}t)}{\hat{e}_t(x_i)} \cdot \mathbb{1}\{W_i = t\}$$

Not required for early phases, but natural next steps.
