Can XGBoost Learn Ratio and Product Features From Raw Inputs Under Distribution Shift?

Anonymous Author

Abstract

Many predictive tasks in tabular data use engineered features such as ratios or products. These features can encode invariances to rescaling and unit changes. This paper studies whether XGBoost learns these structures from raw positive inputs in a way that remains stable under targeted distribution shifts.

This paper builds a synthetic benchmark of binary classification tasks. Each task is defined by a small number of latent coordinates. The coordinates are log-ratios or log-products of subsets of raw features. A task then applies a simple motif to these coordinates. Motifs include monotone coordinates, contrasts, interactions, non-monotone transforms, and gating.

This paper trains XGBoost in a raw mode and two oracle modes. The raw mode uses only raw inputs. The oracle coordinate mode augments inputs with intermediate coordinates. This is an analogue of feature engineering. The oracle signal mode augments inputs with the final latent signal. Across experiments, the raw mode often fits training data but fails under targeted shifts that preserve the intended coordinate. The largest gaps occur on interaction motifs and on preserve shifts. These failures indicate shortcut learning in raw space rather than recovery of the intended invariant structure.

1 Introduction

XGBoost is widely used for tabular prediction. It can approximate many nonlinear functions. Approximation does not guarantee robust generalization. A model can fit training data using incidental cues. This matters under distribution shift.

Ratios and products are common engineered features. They often encode invariances. A ratio is invariant to multiplying numerator and denominator by the same constant. A product is invariant to inverse scaling that keeps the product fixed. A robust predictor should not degrade when raw magnitudes change but the true coordinate stays the same.

This paper asks a targeted question. Can XGBoost trained on raw features learn ratio or product structure in a way that remains stable under shifts that preserve that structure?

When to care. This issue matters when feature magnitudes can change without changing the intended meaning. This includes unit changes, inflation or denomination changes, sensor calibration drift, vendor and geography mix shifts, and pipeline normalization changes.

What to do in practice. If a ratio or product is known and cheap, engineer it. If the structure is unknown, use targeted perturbation checks that preserve candidate invariances. If performance is sensitive to such checks, treat the model as using magnitude shortcuts.

Related work. Tree ensembles are a standard baseline for tabular prediction. Random forests and gradient boosted decision trees are widely used due to strong accuracy and simple deployment [1, 2]. XGBoost is a common implementation and is often competitive on structured datasets [3]. These models partition the feature space with axis-aligned splits. They can represent interactions, ratios, and products, but the representation can be inefficient when the signal depends on coordinated behavior across many raw features. Practitioners often address this with feature engineering such as log transforms, ratios, and products.

Robustness under shift and invariant structure. A separate literature studies generalization under distribution shift. One line of work uses invariance across environments as a guiding principle [4, 5]. Another line emphasizes that strong training performance can coexist with unstable behavior due to spurious correlations and underspecification [6, 7]. These viewpoints motivate evaluations that apply targeted interventions or perturbations that preserve the intended semantics while changing nuisance variation. This paper follows that logic in a tabular setting. It uses preserve shifts that keep a latent coordinate fixed while changing raw magnitudes. It then measures whether models trained on raw inputs recover the invariant coordinate or rely on scale-dependent shortcuts.

2 Problem setup

Let $x \in \mathbb{R}^d_{>0}$ denote raw features. A task defines a latent signal $s = f(x)$. The label is a noisy function of s. This paper uses a logistic link for most tasks:

$$\mathbb{P}(y = 1 \mid x) = \sigma\big(\beta(s - t)\big), \qquad \sigma(z) = \frac{1}{1 + e^{-z}}, \tag{1}$$

where $\beta > 0$ and t controls prevalence.

This paper compares a raw feature model to two oracle variants. The raw mode uses only x. The oracle coordinate mode augments x with intermediate coordinates. The oracle signal mode augments x with the final signal s. The coordinate oracle is the closer analogue to feature engineering. The signal oracle is an upper bound.

This paper measures performance with test PRAUC. This paper also measures a paired gap:

$$\Delta\text{PRAUC} = \text{PRAUC}_{\text{oracle}} - \text{PRAUC}_{\text{raw}}. \tag{2}$$

3 Synthetic benchmark

3.1 Coordinates and task motifs

The benchmark is built in log-coordinates. This is a modeling choice. It reduces dynamic range and makes the intended invariances explicit.

Define a small stabilizer $\epsilon > 0$. In the implementation, ϵ is a single global constant reused across coordinates within a dataset. In ratio coordinates, ϵ is added inside each sum. In product coordinates, ϵ is added after forming the product.

This paper uses two coordinate types. The first type is a log-ratio of sums:

$$u_R(A, B) = \log\left(\frac{\sum_{i\in A} x_i + \epsilon}{\sum_{j\in B} x_j + \epsilon}\right). \tag{3}$$

The second type is a log-product:

$$u_P(A) = \log\left(\prod_{i\in A} x_i + \epsilon\right). \tag{4}$$

A task then applies a simple motif to one or more coordinates. The benchmark assigns each task a level label that indexes a motif family. The level label is not a formal ordering by function-class complexity. Appendix A gives representative task definitions.

Level families are:

- **Level 1 (single coordinate).** $s = u$ where u is one log-ratio or one log-product coordinate.
- **Level 2 (aggregated coordinate).** $s = u$ where u is a log-ratio of sums with larger index sets.
- **Level 3 (contrast).** $s = u_1 - u_2$ for two coordinates of the same type.
- **Level 4 (interaction).** $s = u_1 u_2$ for two coordinates. This level includes ratio$\times$ratio and product$\times$product subtypes.
- **Level 5 (hybrid interaction).** $s = u_1 u_2$ where one coordinate is ratio-type and one is product-type.
- **Level 6 (non-monotone shape).** $s = g(u)$ where g is non-monotone in u.
- **Level 7 (gating).** s selects between two coordinates based on a gate coordinate. Level 7 is provisional in the current implementation.

3.2 Distribution regimes and shifts

This paper generates features under several regimes. Some regimes are static and change tails or correlations. Other regimes impose explicit train-test shifts.

This paper includes a *preserve shift*. The preserve shift changes raw magnitudes while preserving the intended coordinate. For ratio tasks, the shift multiplies numerator and denominator features by a common factor. For product tasks, the shift applies inverse scaling that keeps the product constant.

The preserve shift is an identifiability device. Without it, a raw-only model can score well by exploiting magnitude cues that correlate with the label under a specific regime. Those cues can disappear when units or scales change. A model that truly uses the intended coordinate should be stable under preserve shifts.

The stabilizer ϵ makes the invariance approximate in finite samples. The approximation is tight when feature magnitudes are large relative to ϵ. Appendix B states the preserve shift more precisely.

3.3 Experimental protocol

This paper uses a fixed train-validation-test split within each dataset. The main experiment uses $n = 30{,}000$ training examples and a similarly sized test set. This paper repeats each configuration over three random seeds.

3.4 Models

This paper trains XGBoost classifiers with a small set of capacity settings. The main settings vary maximum tree depth. This paper uses a validation set for early stopping. Appendix D lists the training configuration.

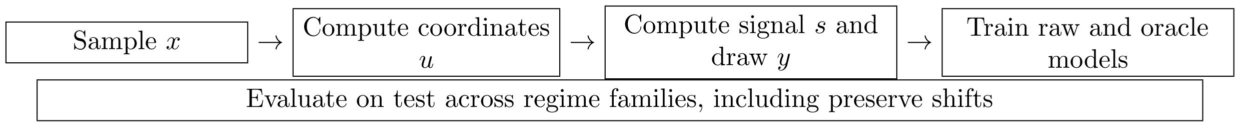
Sample x
Compute coordinates u
Compute signal s and draw y
Train raw and oracle models
Evaluate on test across regime families, including preserve shifts

Figure 1: Benchmark pipeline.

4 Results

Unless stated otherwise, tables in the main text use the baseline XGBoost configuration. Appendix C reports additional breakdowns.

4.1 Regime-family breakdown motivates preserve shifts

Table 1 summarizes ΔPRAUC by regime family. Two oracle variants are shown. The coordinate oracle is the feature engineering analogue. The signal oracle is an upper bound.

Preserve shifts produce substantially larger gaps than the other regime families. This is consistent with raw models using scale-dependent shortcuts that do not transfer when the coordinate is preserved but magnitudes change.

Regime family	ΔPRAUC (coords)	[p10, p90]	ΔPRAUC (signal)	[p10, p90]
mixture_extremes	0.016	[0.003, 0.046]	0.025	[0.004, 0.070]
shift_naive	0.016	[-0.010, 0.095]	0.023	[-0.008, 0.092]
shift_preserve	0.146	[0.017, 0.392]	0.151	[0.019, 0.418]
tail_corr	0.007	[-0.006, 0.037]	0.011	[-0.004, 0.057]

Table 1: Median ΔPRAUC by regime family for the baseline XGBoost configuration. “Coords” means oracle coordinate augmentation. “Signal” means oracle signal augmentation.

4.2 Feature engineering realism: oracle coordinates versus oracle signal

Table 2 summarizes results by motif family. The coordinate oracle is consistently closer to the raw model than the signal oracle. This is expected because the coordinate oracle is a partial disclosure of structure. The main qualitative pattern is the same. Contrast and interaction motifs have larger gaps than single-coordinate motifs.

Level 7 is included for completeness. It is provisional in the current implementation. Headline claims in this paper focus on Levels 1 to 6.

Level	Raw med PRAUC	Oracle med PRAUC (coords)	Δ (coords)	Oracle med PRAUC (signal)	Δ (signal)
1	0.205	0.244	0.006	0.239	0.008
2	0.132	0.161	0.011	0.157	0.014
3	0.211	0.319	0.028	0.330	0.047
4	0.116	0.236	0.035	0.246	0.052
5	0.376	0.453	0.050	0.477	0.062
6	0.054	0.061	0.006	0.065	0.008
7[†]	0.333	0.332	0.008	0.346	0.016

Table 2: Results by level for the baseline XGBoost configuration. Deltas are paired per run and then aggregated. [†] Level 7 is provisional.

4.3 Vignette: largest gaps under preserve shifts

Table 3 lists the five largest ΔPRAUC cases under the preserve shift family for the coordinate oracle. These cases are not representative. They show that raw models can collapse even when an invariant structure exists and is easy to express in engineered coordinates.

Task	Level	Kind	Seed	Raw PRAUC	Oracle PRAUC	ΔPRAUC
l4_product_x_product	4	product_x_product	0	0.144	0.903	0.759
l4_product_x_product	4	product_x_product	2	0.261	0.899	0.638
l5_ratio_x_product	5	ratio_x_product	0	0.048	0.444	0.396
l5_ratio_x_product	5	ratio_x_product	2	0.051	0.446	0.395
l5_ratio_x_product	5	ratio_x_product	1	0.066	0.455	0.390

Table 3: Top five ΔPRAUC runs under preserve shifts for the baseline configuration with oracle coordinate augmentation.

4.4 Preserve shifts expose noninvariant learning

A preserve shift should not hurt a predictor that uses the correct coordinate. However, preserve shifts can cause large failures for the raw feature model.

The product×product interaction case in Table 3 shows a raw model PRAUC near 0.14 while the coordinate oracle reaches about 0.90. This is one example of a failure mode. The raw feature model relied on scale-dependent cues. Those cues did not transfer.

Across tasks, preserve shifts tend to produce larger median ΔPRAUC than static regimes that only change tails or correlations. Table 1 quantifies this pattern.

4.5 Diagnostics align with performance gaps

This paper uses diagnostics that test invariance directly. This paper also measures iso-coordinate variation.

For a ratio coordinate, this paper scales numerator and denominator together. This keeps the coordinate u_R fixed up to the stabilizer ϵ. A perfectly invariant predictor should not change under this perturbation. This paper reports a ratio invariance error based on prediction differences under such perturbations. This paper also reports an iso variation score that measures prediction variability along iso-coordinate directions.

These diagnostics are mechanistic checks. Preserve shift is the performance stress test. Runs with larger invariance error tend to have larger ΔPRAUC.

5 Discussion and future work

This benchmark isolates a specific risk. A raw-only model can fit data using magnitude cues that are cheap for trees to exploit. Those cues can be unstable under unit and scale changes. Oracle coordinates reduce this risk because they expose the invariant structure directly.

This paper has limitations. It uses synthetic data. It uses a small set of model settings. It does not yet include a wide set of baseline learners.

This paper plans several extensions. These are written as reader-facing planned experiments. Appendix E provides an implementation checklist.

Planned experiments include:

- A raw baseline that trains on $\log(x+\epsilon)$ to test whether failures are primarily dynamic-range effects.
- A seed expansion on a reduced grid that focuses on Levels 3 to 5 under preserve shifts.
- A dataset size sweep to estimate sample complexity for learning invariant coordinates.
- Additional learners on oracle coordinates, including linear models and small neural nets.

A Task definitions

This appendix gives representative task definitions. Each task uses disjoint index sets unless stated otherwise. All tasks use the coordinate definitions in Eqs. (3) and (4).

Level 1 examples

Ratio: $s = u_R(\{0,1\},\{2,3\})$. Product: $s = u_P(\{0,1\})$.

Level 2 examples

Ratio of larger sums: $s = u_R(\{0,1,2,3\},\{4,5,6,7\})$.

Level 3 examples

Contrast of ratios: $s = u_R(\{0,1\},\{2,3\}) - u_R(\{4,5\},\{6,7\})$. Contrast of products: $s = u_P(\{0,1\}) - u_P(\{2,3\})$.

Level 4 examples

Ratio×ratio: $s = u_R(\{0,1\},\{2,3\}) \cdot u_R(\{4,5\},\{6,7\})$. Product×product: $s = u_P(\{0,1\}) \cdot u_P(\{2,3\})$.

Level 5 examples

Hybrid interaction: $s = u_R(\{0,1\},\{2,3\}) \cdot u_P(\{4,5\})$.

Level 6 examples

Band-pass shape on a coordinate:

$$s = \exp\left(-\frac{(u-\mu)^2}{2\sigma^2}\right), \qquad u = u_R(\{0,1\},\{2,3\}). \tag{5}$$

This creates a non-monotone relationship between u and the label.

Level 7 examples (provisional)

Gating:

$$s = \begin{cases} u_a, & u_g > 0, \\ u_b, & u_g \le 0, \end{cases} \qquad u_g = u_R(\{0,1\},\{2,3\}), \quad u_a = u_R(\{4,5\},\{6,7\}), \quad u_b = u_P(\{8,9\}). \tag{6}$$

B Preserve shift construction

This appendix states the preserve shift at the coordinate level.

For a ratio coordinate $u_R(A, B)$, define a positive scalar c. Apply $x_i \leftarrow cx_i$ for $i \in A \cup B$. If $\epsilon = 0$, this leaves $u_R(A, B)$ unchanged. With $\epsilon > 0$, the coordinate is approximately preserved when sums dominate ϵ.

For a product coordinate $u_P(A)$, preserve shifts use inverse scaling. For example, for $A = \{i, j\}$ apply $x_i \leftarrow cx_i$ and $x_j \leftarrow x_j/c$. This keeps $\prod_{k \in A} x_k$ fixed. With $\epsilon > 0$, the coordinate is approximately preserved when products dominate ϵ.

C Additional result breakdowns

This appendix reports a compact level×regime-family breakdown for the coordinate oracle. All values are median ΔPRAUC for the baseline configuration.

Level	mixture_extremes	tail_corr	shift_naive	shift_preserve
1	0.011	0.002	-0.003	0.190
2	0.018	0.005	0.005	0.096
3	0.034	0.018	0.019	0.178
4	0.057	0.013	0.087	0.190
5	0.041	0.028	0.060	0.395
6	0.004	0.006	0.016	0.017
7[†]	0.018	0.009	-0.004	0.019

Table 4: Median ΔPRAUC by level and regime family for oracle coordinate augmentation. [†] Level 7 is provisional.

D Training configuration

This paper uses standard XGBoost binary classification training. This paper uses early stopping on a validation split. This paper varies maximum depth in the main grid. Other settings follow common defaults. The next iteration will record a complete configuration table produced from the experiment harness.

E Implementation checklist for the next iteration

This appendix is an implementation-oriented checklist intended to support the next revision cycle.

- Add a raw baseline that trains on $\log(x + \epsilon)$. Restrict to preserve shifts and a small subset of tasks. Use one XGBoost configuration and a small seed count.
- Expand seeds on a reduced grid. Focus on Levels 3 to 5 under preserve shifts. Report uncertainty bands for ΔPRAUC.
- Add an explicit sweep that varies correlation while holding tail-heaviness fixed. Add a separate sweep that varies tail-heaviness while holding correlation fixed.
- Add a dataset size sweep. Use at least three training sizes. Keep the same test set.

- Add baseline learners on oracle coordinates. Include a linear model and a small neural net baseline.
- Revisit Level 7 gating. Ensure the gate definition and preserve shift are aligned. Keep Level 7 marked as provisional until this check is complete.
- Add a compact schematic figure of the benchmark pipeline if Fig. 1 is replaced or expanded.

References

[1] L. Breiman. Random forests. *Machine Learning*, 45:5–32, 2001.

[2] J. H. Friedman. Greedy function approximation: A gradient boosting machine. *Annals of Statistics*, 29(5):1189–1232, 2001.

[3] T. Chen and C. Guestrin. XGBoost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2016.

[4] J. Peters, P. Bühlmann, and N. Meinshausen. Causal inference using invariant prediction: Identification and confidence intervals. *Journal of the Royal Statistical Society: Series B*, 78(5):947–1012, 2016.

[5] M. Arjovsky, L. Bottou, I. Gulrajani, and D. Lopez-Paz. Invariant risk minimization. In *International Conference on Learning Representations*, 2019.

[6] R. Geirhos, J.-H. Jacobsen, C. Michaelis, R. Zemel, W. Brendel, M. Bethge, and F. A. Wichmann. Shortcut learning in deep neural networks. *Nature Machine Intelligence*, 2:665–673, 2020.

[7] A. D'Amour, K. Heller, D. Moldovan, B. Adlam, B. Alipanahi, A. Beutel, and others. Underspecification presents challenges for credibility in modern machine learning. In *Advances in Neural Information Processing Systems*, 2020.