

FORTE: A Production Framework for Feature Evaluation and Deprecation in Industrial Recommender Systems

Vishnu Arun
Pinterest
San Francisco, CA, USA
vishnuarun@pinterest.com

Adam Obeng
Pinterest
San Francisco, CA, USA
aobeng@pinterest.com

Tianyou Wang*
Pinterest
San Francisco, CA, USA
tianyouwang@pinterest.com

Zilin Ren
Pinterest
San Francisco, CA, USA
zren@pinterest.com

Mukuntha Narayanan S
Pinterest
San Francisco, CA, USA
mukuntha@pinterest.com

Han Wang
Pinterest
San Francisco, CA, USA
hanwang@pinterest.com

Shivin Thukral
Pinterest
San Francisco, CA, USA
sthukral@pinterest.com

Zhixiang Wang
Pinterest
San Francisco, CA, USA
zhixiangwang@pinterest.com

Saurabh Vishwas Joshi*
Pinterest
San Francisco, CA, USA
saurabhsvj@gmail.com

Darren Reger
Pinterest
San Francisco, CA, USA
dreger@pinterest.com

Abstract

Production recommender systems have feature-lifecycle asymmetry: adding new features need only produce gains on some metric, while feature removal must not regress any metric. In practice, feature deprecation is hard to prioritize as a feature can easily fail to be removed: if it interacts with other features, its importance changes daily, it cannot be evaluated before production models change or data drift occurs, it is in contention for A/B test space with other launches, it causes regressions in an online A/B test, or it is consumed by multiple models that need to coordinate its deprecation.

We present FORTE, a model-agnostic production framework for discovering, validating, and removing features. Its efficient perturbation implementation allows daily feature evaluation. To account for interactions, it evaluates candidate feature sets constructed from features with the weakest measured importance through grouped perturbation, providing an offline gate before online testing. Lineage tracing identifies all downstream consumer models to realize the full savings of removing a feature from the training set.

We validate FORTE over 2 large-scale production recommender systems and find that it identifies deprecable feature sets which do not cause online regressions, while the impact on efficiency depends on model characteristics. On a search-relevance model, FORTE resulted in 3 nested candidate sets of 32, 48, and 59 features. During offline evaluation and retraining, all 3 candidates appeared neutral, but online guardrails rejected the largest treatment. The 48-feature set was selected for deprecation, removing 43.2% of the model’s features. On a large ranking model, FORTE identified 8 candidates for removal. Retraining and A/B testing confirmed no regression. Its launch reduced the trainable parameters by 29.9%,

improved serving throughput by 19.8–25.5%, and reduced annual training costs by hundreds of thousands of dollars.

CCS Concepts

• **Information systems** → **Recommender systems**; • **Computing methodologies** → *Machine learning*.

Keywords

Feature Deprecation, Feature Importance, Recommender Systems, Production Machine Learning, Distributed Inference

1 Introduction

1.1 Feature-lifecycle asymmetry

Modern recommender systems drive engagement and revenue across billions of daily interactions. Pinterest deploys hundreds of models for retrieval, ranking, and engagement prediction, each ingesting tens to thousands of features.

Model evolution requires experimenting with dozens of features at any given time. Adding a new feature is generally justified by a gain in a particular user behavioral metric, assuming adherence to a strict set of model performance guardrails. Removing one is instead justified by an efficiency gain, and its criteria are harder to satisfy: beyond the same guardrails, deprecation generally requires validating that no user metric regresses.

Validating a feature removal requires both retraining a model and running an online A/B test of the treatment against the current production model. A/B tests are a limited resource and generally prioritize high-impact treatments, which feature removals do not promise. Deprecation also creates a combinatorial allocation problem: a model with d features has $2^d - 1$ nonempty candidate removal

*Work done while at Pinterest

sets. Although only a small fraction will be plausible, this candidate space is substantially larger than the set of new feature launches competing for the same A/B capacity. This incentive structure tends to lead to the persistence of ineffective features.

1.2 Deprecation in practice

In industry, model development typically follows the same iteration cycle for both adding and removing a feature: propose the change, retrain a model with it enabled, backtest offline on evaluation data, run an online A/B test, validate online user behavioral metrics and model performance guardrails against the baseline, then audit and ship. As a change progresses through the cycle, the associated costs increase as resources constrain.

Deprecation only realizes its benefits after the end of this cycle: removing a feature reduces trainable parameters, producing a leaner model with lower training and serving cost, and once no model references the feature it can stop being collected in training and evaluation sets altogether.

We define deprecation specifically as removing a feature from a model which is already deployed. This differs from the related concept of feature selection, which is typically performed during model development to identify and remove low-signal features to improve model generalization performance.

We identify the following failure modes for feature deprecation in an industry setting. Each can prevent the feature-deprecation cycle from completing, cause offline validation results to diverge from online outcomes, or both:

- Feature interactions.
- Data drift.
- Model versioning and drift.
- Online A/B bandwidth contention.
- Online regression.
- Multi-model coordination during deprecation.

1.3 FORTE overview

We identify the predominant bottleneck to deprecation as limited online A/B bandwidth. FORTE introduces a perturbation-based feature importance system which is sufficiently efficient to run perturbation experiments on a daily cadence, generates and ablates candidate groups, and traces feature lineage to provide end-to-end feature removal.

1.4 Contributions

- **Problem formulation.** We define complete feature deprecation in two parts: (1) the ability to identify unimportant feature sets from a model, and (2) tracking a feature’s complete lineage and deprecating it once it is deemed unimportant.
- **FORTE, an end-to-end framework spanning discovery, validation, and removal.** We combine temporal feature importance, pooled perturbation, online A/B validation, and feature lineage tracking into a single workflow.

- **Efficient perturbation implementation.** We define an efficient perturbation implementation with a minimal marginal cost per additional perturbation experiment. This enables running individual-feature perturbation experiments daily.
- **Case studies from two production recommender systems.** We walk through a Search Relevance model and a Pin Ranker model.

2 Related Work

Permutation importance is a model-agnostic way to measure model reliance. Breiman [3] introduced it for Random Forests, and Fisher, Rudin, and Dominici [9] generalized it to Model Class Reliance. These estimators match our operational question: how much does model quality change when a feature’s signal is destroyed? Existing implementations evaluate features sequentially and lack native support for heterogeneous tensor types found in production ML models.

The main statistical limitation of marginal permutation is feature dependence. Unconditional permutation can create out-of-distribution inputs and distort measured importance for correlated predictors [11, 19]. Conditional PFI [23] addresses this by conditioning on the other predictors when permuting, but requires estimating conditional structure. Covert, Lundberg, and Lee [7] distinguish fixed-value removal from distributional marginalization; the latter may use marginal or conditional replacements. FORTE’s ablation uses fixed nulls, while permutation uses batch-shuffled values as empirical marginal replacements. We do not claim conditional importance, instead using online A/B validation for feature removal.

Stronger attribution alternatives are prohibitive at daily cadence. LOCO [14] retrains without each feature, requiring p retrains. ROAR [12] evaluates rankings by cumulative removal and retraining; its findings motivate external validation. SHAP [16], KernelSHAP [5], and SAGE [6] use Shapley-style coalitions, but sampling cost is prohibitive at thousands of features. TreeSHAP [15] cuts this cost for tree ensembles but does not apply to the neural rankers we study.

Backpropagation-based attribution methods such as Integrated Gradients [25] and DeepLIFT [22] require access to model internals and a reference input; baseline choice can materially affect the resulting attribution [24]. FORTE targets global, perturbation-based importance.

Streaming and online importance estimation [10, 18] update with each sample; FORTE uses scheduled model evaluations for operational monitoring.

Unmanaged feature dependencies contribute to ML technical debt [20]. Before FORTE, feature importance at Pinterest required one inference job per feature, making daily evaluation infeasible. FORTE combines pooled inference, type-aware perturbation, temporal tracking, and online validation in one workflow.

3 Problem Formulation

Terminology and sign convention. In an offline perturbation experiment, the *baseline condition* is the frozen model evaluated on the intact examples, and the *perturbed condition* is the same model evaluated after a feature group G has been permuted or ablated.

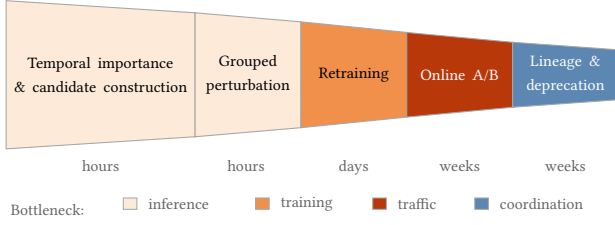


Figure 1: FORTE’s deprecation funnel. Candidate sets narrow through validation and deprecation; fill identifies each stage’s bottleneck.

Let M_0 and M_G denote the resulting baseline and perturbed metrics. FORTE reports the direction-adjusted normalized change

$$\Delta_M(G) = 100 s_M \cdot \frac{M_G - M_0}{M_0}. \quad (1)$$

Here $s_M = +1$ when higher values of M are better and $s_M = -1$ when lower values are better. We refer to this signed normalized change as the *feature-importance score*. Thus, a negative score means that perturbing G degrades model quality and the model relies on the feature group; a score near zero indicates weak measured reliance; and a positive score means that perturbation improves model quality. Under this convention, scores increase as measured reliance decreases, so deprecation candidates are the features with scores near zero or above. We report a signed change rather than a metric drop so that the score carries the same orientation as the online A/B results it is compared against. Classical permutation importance [3, 9] reports the negated quantity, so reliance appears there as a positive degradation; readers should keep this difference in mind when comparing magnitudes across the two conventions.

In the retraining and online stages, the *baseline* (or online *control*) is the incumbent production model, while the *treatment* is a model retrained with a candidate set removed. To avoid overloading these terms we separate offline from online work throughout. A *perturbation experiment* is one configured offline comparison of the baseline and perturbed conditions on a frozen model, and we report offline results per experiment. A *treatment arm* is a retrained candidate model allocated online traffic in an A/B test, and we report online results per arm; arms are the scarce resource the funnel of Section 4 is ordered around. A *candidate set* is a collection of features proposed for joint removal.

Given a production model f with feature set $\{X_1, \dots, X_d\}$ and evaluation metric M , we aim to:

- (1) Compute daily feature importance scores $FI_j(t)$ for all features j at each day t , producing temporal series $TFI_j = \{FI_j(1), \dots, FI_j(T)\}$.
- (2) Combine temporal importance into a ranked deprecation candidate set; validate via retraining and online A/B testing.
- (3) Identify feature lineage and create a method for safe deprecation.

4 System Design and Methodology

FORTE assembles deprecation candidates through the deprecation funnel illustrated in Figure 1. Its stages differ in which resource they consume, and online A/B traffic is the bottleneck: it admits only a handful of arms. The funnel is therefore ordered so that stages

bound by amortized inference screen the full feature set before any candidate reaches a stage bound by training capacity or online traffic.

4.1 Temporal feature importance

We set up a scheduled perturbation experiment for each model. A resilient workflow reads a partition of the evaluation dataset and evaluates the current production model.

At industry scale across a diverse model fleet we implement several safeguards to ensure continuous data collection. Upstream issues can lead to partitions being delayed or missing; we include partition sensors in our design to skip and backfill. In a production setting, teams have variable model iteration cycles. We define the current production model by implementing a production model sensor to automatically update the model reference as serving traffic diverts to a newer checkpoint.

For feature X_j on day t , temporal feature importance is the signed normalized change $FI_j(t) = \Delta_M(\{X_j\}; t)$ defined in Equation (1). The baseline and perturbed conditions use the same checkpoint and the same day’s examples. This paired comparison controls level shifts shared by both conditions, while normalization by the baseline expresses the result as a relative percentage. It does not remove permutation noise or genuine changes in the model’s sensitivity to a feature.

Candidate construction. We express the temporal gate in the conventional reliance orientation, $I_j(t) = -FI_j(t)$, so larger values indicate greater measured reliance. Across each configured lookback window, we use mean importance, rank stability, and fitted slope as indicators for deprecation candidates. By default, a candidate must have low mean importance, remain stably near the bottom of the feature ranking, and have a non-positive slope, with these conditions required to hold across every output head and lookback window. The gate is a per-model configuration rather than a fixed rule: owning teams set the window lengths, the filter thresholds, and which heads a candidate must satisfy, according to operational concerns such as model release frequency and the amount of post-release evidence required before nominating a feature. The default is deliberately conservative because a candidate that survives it still has to clear the more expensive gates downstream. Additional signals include estimated feature costs. We use these signals to generate multiple nested candidate sets.

Two examples in Figure 2 show how rank changes over time. Feature A remains relatively low-ranked but varies substantially. Feature B is generally higher-ranked before moving sharply near the end of the window. This variation motivates evaluating mean rank, stability, and trend over a lookback window rather than using one observation.

We do not assert that this method rigorously identifies all non-correlated candidates, but we assert that it will provide a signal that is validated in a future gate.

4.2 Grouped perturbation

We take the nested feature sets and perform grouped perturbation on each to validate their joint perturbation effect, capturing

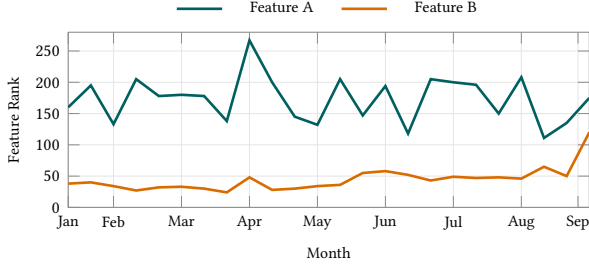


Figure 2: Feature-rank trajectories for two features across scheduled observations. Lower ranks denote more highly ranked features. The observations retain their original order; the horizontal labels show the corresponding abbreviated month names.

interactions and dependence. This joint test addresses one operational consequence of feature dependence [11, 19]: correlated candidates can substitute for one another under individual permutation, whereas perturbing them together measures reliance on their combined signal. Grouping does not produce conditional importance or eliminate out-of-distribution perturbations. The pooled inference run shares its principal fixed costs across candidate sets. At this stage we seek to reduce the number of candidate sets carried into retraining because each surviving set typically takes days to retrain, considerably longer than offline ablation. We first filter candidate sets for metric neutrality, then retain up to N passing sets ranked by their potential cost savings from feature deprecation. The value of N and the sizes of these sets are team-specific choices reflecting cost priorities, retraining capacity, and available A/B traffic.

4.3 Retraining, online validation, launch review

We retrain a model with each candidate set removed and compare it with the production baseline. Neutral performance metrics are the criterion to move to the online validation stage.

We promote S of the candidate sets that passed retraining to treatment arms. A/B validation consists of running a test for multiple weeks and benchmarking each arm against the production control. A/B test completion has the following criteria:

- Online metric neutrality.
- User behavioral guardrail adherence.

Passing the above two allows the team to move this to human audit and launch review to finally ship the deprecation.

4.4 Feature lineage and deprecation

FORTE automatically detects dereferenced features by maintaining a live feature-model mapping. Once a feature has been removed from the model, FORTE determines whether it is entirely dereferenced. Teams can then initiate the formal deprecation process, which progressively blocks the feature and prevents any model that still references it from training. Blocking and removing features can have real costs if they need to be restored or revived, so deprecation proceeds through a staged manual audit process:

- **Dataloader block.** If a feature is attempted to be added back into a model it will trigger here.
- **Dataset block.** The feature is hidden from reads, and the block propagates to downstream leaf datasets.

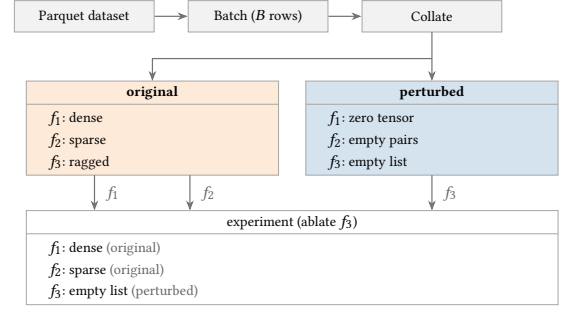


Figure 3: Dataset virtualization (ablation): the *original* dictionary holds typed batches; *perturbed* holds type-appropriate nulls. Experiments draw each feature column (f_n) from one of these dictionaries.

- **Ingestion block.** Evaluation and training datasets block a blocked feature from being written.
- **Removal from logging.** The final step, which removes the feature entirely from logging.

4.5 Dataset Virtualization and Pooled Execution

Individual-feature importance permutes or ablates a feature after reading in and materializing the dataset, once per experiment. We instead store an ablated/permuted in-memory copy of the entire batched dataset, so that all experiments for a model resolve against a single materialization in one distributed inference pass.

The evaluation dataset is laid out on disk in a columnar storage format. Rows are read from these files and collated into batches, converting each raw column into the tensor format its feature registry schema declares: dense values, sparse (index, value) pairs, or ragged variable-length lists. Collation yields a tensor dictionary D_{original} keyed by feature name. We build $D_{\text{perturbed}}$ alongside it, mirroring D_{original} in keys and batch dimensions, each value the perturbed form of that feature: a permuted batch under permutation, a type-appropriate null under ablation.

As Figure 3 illustrates, an experiment is a per-feature selection between the two dictionaries. Each inference worker receives both once per batch and iterates over the configured experiment set: the baseline draws every feature from D_{original} , and each subsequent experiment draws its feature group G from $D_{\text{perturbed}}$ and the remainder from D_{original} , runs the already-loaded model, and tags the output with an experiment index. Permutation and ablation share this loop, differing only in how $D_{\text{perturbed}}$ is constructed. Model loading, data I/O, actor startup, and GPU allocation amortize across every experiment in the batch, so p experiments cost one materialization rather than p . Section 6 measures the resulting runtime behavior (Figure 5).

5 Practical Considerations for Production Deployment

5.1 Single-Permutation Estimation at Scale

Repeating permutations reduces Monte Carlo noise but multiplies inference cost, which could instead be spent on inference over a larger dataset. We use a theory-to-empirical bridge: following tractable ROC-AUC analyses [1], we first establish a reference

variance-scaling law for ROC-AUC, whose pairwise decomposition supports a clean bounded-differences argument. We then test whether the resulting operating conclusion extends empirically to PR-AUC and production feature-ranking decisions. This second step supports an operational generalization; it does not prove a PR-AUC scaling law.

Fix an evaluation dataset with n_+ positive and n_- negative examples, $n = n_+ + n_-$, a feature group G , and a frozen model. Let Π_n denote the finite set of permutations allowed by the evaluation implementation. For a global permutation, $\Pi_n = S_n$; because FORTE permutes within inference batches, its implemented space is the product of the corresponding within-batch permutation sets. For $\pi \in \Pi_n$, let $A_G(\pi)$ denote the tie-adjusted AUC after permuting only G .

The analysis is conditional on both the fixed evaluation data and the frozen model architecture and evaluation state. By *row-wise inference*, we mean that each score has the form $s_i = f(x_i)$: once model state is fixed, changing the features assigned to row i cannot change another row's score. This condition excludes inference-time batch normalization that recomputes statistics from the current batch, joint listwise or slate-level scoring and normalization, cross-candidate attention, and GNN neighborhood aggregation when perturbing one row changes neighboring representations or scores. A model trained with a listwise objective still satisfies the condition if its deployed inference is row-wise. The Pinterest models evaluated here use frozen evaluation state and score examples independently.

Each admissible permutation breaks the alignment between G and the remaining features in a different way. Since no admissible permutation is privileged, the natural fixed-dataset estimand is their uniform average,

$$\mu_{n,G} = \mathbb{E}_{\pi \sim \text{Unif}(\Pi_n)} [A_G(\pi)].$$

This estimand is defined over the permutation space implemented by the system; it need not equal the average over all of S_n when permutation is restricted to batches.

For R independent uniform draws with replacement $\pi_1, \dots, \pi_R$ from Π_n , define

$$\hat{A}_{R,G} = \frac{1}{R} \sum_{r=1}^R A_G(\pi_r).$$

The standard Monte Carlo identities give

$$\mathbb{E}[\hat{A}_{R,G}] = \mu_{n,G}, \quad \text{Var}(\hat{A}_{R,G}) = \frac{\sigma_{n,G}^2}{R}, \quad \sigma_{n,G}^2 = \text{Var}_{\pi}(A_G(\pi)).$$

If $A_0 > 0$ is the fixed, unpermuted AUC, FORTE's normalized importance estimate is

$$\hat{\Delta}_R(G) = 100 \frac{\hat{A}_{R,G} - A_0}{A_0}.$$

It has expectation $100(\mu_{n,G} - A_0)/A_0$ and variance $10^4 \sigma_{n,G}^2 / (R A_0^2)$. A single draw is therefore already unbiased for $\mu_{n,G}$, and averaging R draws only reduces variance by a factor of $1/R$ without changing the fixed-dataset estimand.

The remaining question is how large $\sigma_{n,G}^2$ itself is, since that is what sets the precision a single draw achieves.

PROPOSITION 5.1. *Conditional on the fixed evaluation dataset and frozen model state, assume $n_+, n_- = \Theta(n)$ and row-wise inference as*

defined above. Then, for either global permutation or FORTE's product of within-batch permutation spaces,

$$\sigma_{n,G}^2 = \text{Var}_{\pi \sim \text{Unif}(\Pi_n)} [A_G(\pi)] = O\left(\frac{1}{n}\right).$$

PROOF. Fix the dataset and frozen model state, and view $A_G : \Pi_n \rightarrow [0, 1]$ as a function of the permutation alone. For a transposition $\tau = (ab)$ acting within one permutation block, π and $\pi \circ \tau$ differ only in the values assigned to rows a and b ; under row-wise inference only those two scores change. A pairwise indicator can therefore change only for pairs $(k, l) \in I^+ \times I^-$ with $\{k, l\} \cap \{a, b\} \neq \emptyset$, of which there are at most $2 \max(n_+, n_-) \leq 2n$. Each contributes weight $1/(n_+ n_-)$, giving the per-swap bound

$$|A_G(\pi) - A_G(\pi \circ \tau)| \leq c_n = \frac{2n}{n_+ n_-} = O\left(\frac{1}{n}\right)$$

under $n_+, n_- = \Theta(n)$. A bounded-differences inequality for uniform random permutations [17], applied globally or independently across the within-batch blocks, then gives $\sigma_{n,G}^2 \leq C n c_n^2 = O(1/n)$ for a universal constant C . $\square$

Two levers therefore reduce the variance of a permutation estimate, and they act on different axes: repetition divides it by R at fixed evaluation size, while a larger evaluation sample shrinks $\sigma_{n,G}^2$ itself. Repeated permutations are best understood as variance reduction rather than a change of estimand, and one whose marginal benefit is small in the large- n regime characteristic of production evaluation tables.

Empirical bridge to PR-AUC. The Monte Carlo identities extend directly to PR-AUC. Let $P_G(\pi)$ denote PR-AUC after permutation and $\bar{P}_{R,G} = R^{-1} \sum_{r=1}^R P_G(\pi_r)$. Then $\mathbb{E}[\bar{P}_{R,G}] = \mathbb{E}_{\pi}[P_G(\pi)]$ and $\text{Var}(\bar{P}_{R,G}) = \text{Var}_{\pi}(P_G(\pi))/R$; the same statements hold after subtracting and normalizing by the fixed baseline. However, the $O(1/n)$ rate in Proposition 5.1 uses AUC's pairwise-comparison representation and does not establish a corresponding sample-size rate for PR-AUC, particularly for highly imbalanced heads. We evaluate that behavior empirically below.

The $1/R$ Monte Carlo identities hold for any fixed architecture because they concern averages of independent permutation draws. Only the $O(1/n)$ reference rate depends on row-wise sensitivity; architectures outside that scope require a separate sensitivity analysis.

We test this rate directly. We evaluated ten permutations for 381 feature groups using 15k and 150k examples. Across two output heads and two metrics, increasing the sample size by an order of magnitude reduced the median per-feature permutation standard deviation by 2.58–3.16 $\times$. The comparison is descriptive rather than a direct test of Proposition 5.1: the proposition bounds the variance rather than fixing its constant, the two samples may differ in composition and class balance, and the proposition does not establish a corresponding rate for PR-AUC. This comparison forms the empirical bridge: PR-AUC does not inherit the proposition, but its observed variance reduction tracks the ROC-AUC reference closely enough that it does not alter the downstream feature-ranking decisions reported below.

At 150k examples, a single permutation had a median leave-one-out rank correlation of 0.907–0.959 with the estimate obtained by

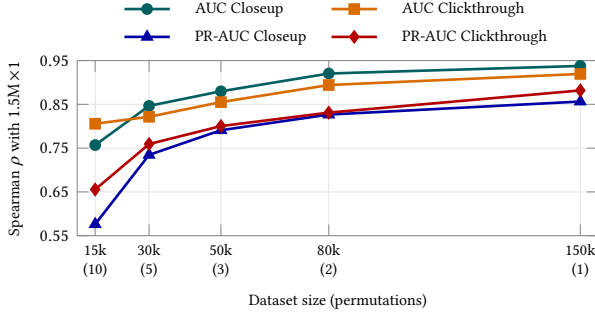


Figure 4: AUC and PR-AUC feature-ranking correlations across approximately matched row-evaluation budgets, measured against a separate $1.5\text{M} \times 1$ reference. Tick labels show dataset size followed by the number of permutations in parentheses.

averaging the other nine permutations, and median top-20 overlap of 92–100%. On PR-AUC, Clickthrough had the weaker standard-deviation reduction, $2.580\times$ compared with Closeup’s $3.082\times$. However, Closeup had the weaker leave-one-out correlation, 0.907 compared with Clickthrough’s 0.942. Thus, at the larger evaluation size, permutation randomness had limited effect on the feature-ranking decisions for which FORTE is used.

Figure 4 shows that feature-ranking agreement with the larger reference rises monotonically as the row-to-permutation ratio increases. All four metric–head combinations agree more at $150\text{k} \times 1$ than at $15\text{k} \times 10$, which processes the same number of row-evaluations, by between 0.11 (AUC Clickthrough) and 0.28 (PR-AUC Closeup) in Spearman ρ , with no reversal at any intermediate configuration.

Across both metrics and heads, the evidence supports $R = 1$ at the evaluated production-scale sample size of roughly 150k examples when the goal is feature ranking, and allocating compute to more representative rows when possible. Our measurements bear on rank correlation and top-20 overlap, not on the accuracy of any individual feature’s importance value. The 15k and 150k samples are not confirmed to be nested, only ten repeats are available, and the 1.5M reference uses one permutation, so it is a larger-sample comparison rather than ground truth. Three to five repeats remain useful for smaller samples or noisy tail metrics when more representative rows are unavailable.

5.2 Engineering for Production-Scale Evaluation

5.2.1 Feature storage and type-awareness during perturbation. Standard libraries define permutation as a row-level operation. FORTE instead applies it to the batched representation the model actually consumes, which requires a custom permutation kernel per feature type. Datasets are defined at a row level and batched at the dataloader–model boundary, so operating after that boundary is what lets a single materialization serve every experiment (Section 4.5).

Ablation raises the same requirement in a different form: each of those feature types needs a null or empty representation. Where feature producers and consumers span an organization, these definitions cannot live in the evaluation job alone. We formalize them in a

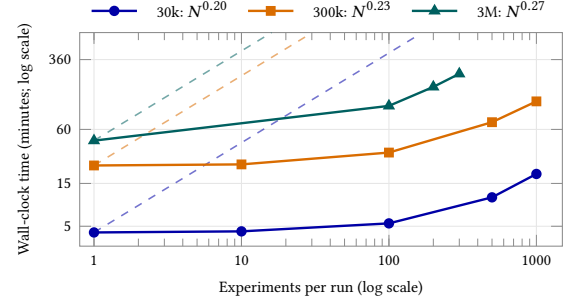


Figure 5: Runtime grows sublinearly with experiment count for all three evaluation-set sizes. Dashed lines show linear scaling from each series’ single-experiment runtime; the 3M-row set was measured through 300 experiments.

Feature Registry that categorizes every supported feature across the ML stack, so a type-appropriate null resolves identically in training and in evaluation.

5.2.2 Production model sensor. Models are iterated on regularly. To support automatic daily feature importance, FORTE resolves the model from serving traffic, ensuring we can track importance on the latest model across our model fleet.

5.2.3 Randomization. Production evaluation tables are generally ordered by ingestion time, partition key, or some upstream join order. They do not guarantee random ordering. Randomizing row order before taking a subset reduces dependence on this physical layout. A full distributed shuffle does so at the cost of communicating data across workers [2]. We define a two-stage shuffle, related to block-aware hierarchical shuffling [29], to reduce layout correlation: a Spark job randomizes rows within each output file, and the inference layer dispatches files to workers in randomized order. Together, the arrival order of inference batches is decoupled from the source table’s physical layout.

For international data we take care not to oversample any single market; the fairness implications are discussed in the ethical considerations section.

6 Evaluation

Unless otherwise noted, the following evaluations use the same frozen production Pin Ranker checkpoint (Section 7.2) and sample rows from its associated evaluation dataset using the two-stage randomization procedure in Section 5.2.3. Each subsection reports its evaluation-set size and experiment-specific configuration.

6.1 Scaling behavior

We compare a legacy (one-pass-per-feature) implementation of feature perturbation experiments against our pooled execution implementation, isolating perturbation experiments from dataset sampling. We vary two axes, dataset size and number of perturbation experiments, choosing a random subset of features for each experiment. Evaluation-set sizes are 30k (small), 300k (medium), and 3M (large) rows, and all runs use the same compute allocation.

For our largest dataset (3M rows), Figure 5 shows a $54\times$ reduction in elapsed time over 300 experiments. The empirical scaling trend

shows clear amortization: runtime grows as $N^{0.20}$, $N^{0.23}$, and $N^{0.27}$ in experiment count for the small, medium, and large evaluation sets respectively, against the N^1 scaling of one evaluation pass per feature. For the 30k-row set, a $1000\times$ increase in experiments costs only $4.5\times$ the wall-clock time.

Adding an additional experiment costs 41 s at 3M rows, against 45 minutes for a separate pass under the one-pass-per-feature approach. The exponent rises with evaluation-set size because forward-pass cost takes a larger share of the total as rows grow: model loading, data I/O, and actor startup are paid once per run, while forward passes and final writes grow with the experiment count. It remains far below linear throughout.

6.2 Grouped perturbation validation

We use 1.5M matched evaluation examples and measure AUC for the Closeup head across 189 feature groups that passed the non-negative Individual-Feature Permutation gate. The individual, fixed-order grouped, and order-averaged estimates use the same examples.

We then jointly perturb nested candidate sets before retraining. For set G_k , let $D_k = 100(\text{AUC}(G_k) - \text{AUC}_0)/\text{AUC}_0$ and $m_k = D_k - D_{k-1}$, with $D_0 = 0$. Because higher AUC is better, a positive m_k indicates less model reliance, whereas a negative m_k reveals greater joint reliance. We order groups from least to most important by their Individual-Feature Permutation score. Pooled inference keeps the marginal cost of each additional group low.

We find 68 of 176 nonzero fixed-order marginals (38.6%) were negative. These negative marginals show why the joint gate is necessary: feature groups that pass individually can reveal model reliance when evaluated in a shared perturbation context. Without making use of grouped perturbation, we might have expected the effect on AUC of removing these features to be the sum of the Individual-Feature Permutation deltas: 6.554%. However, not only is the grouped effect substantially smaller: 3.084%, the interactions between features are sufficiently negative that the peak cumulative delta of 3.899% occurs when only 64 of the candidates which pass the individual gate have been removed.

Order-averaged marginal contributions. The fixed cumulative path measures one addition order. To evaluate sensitivity to coalition context, we average each group’s marginal contribution across 20 reverse-paired randomized orders, yielding an order-averaged marginal contribution for each of the same 189 groups. This is the permutation-sampling estimator [4] of the Shapley value [21]; we report it as an order-sensitivity diagnostic rather than an attribution method, since 20 orders cannot resolve individual Shapley values across 189 groups. Reverse pairing lowers weighted attribution variance by 83–90% relative to independent orders. Because this requires 20 cumulative passes rather than the funnel’s one, it is not a production stage. Figure 6 shows the resulting per-group estimates and uncertainty.

The group total is well determined while its per-group decomposition is not. The order-averaged attributions sum to 3.079%, against the fixed path’s 3.084% endpoint, with a standard error of 0.008% (0.26% of the total). Within a single order the marginals telescope to the full-coalition effect exactly, so the two totals estimate the same quantity and differ only because each path re-measures it under

independent permutation draws; the reported standard error is that measurement noise, not variance across orders. Attribution error cancels in the sum, so the composed set is estimated an order of magnitude more precisely than a typical member.

Two structural effects explain why per-group shares resist identification. The model reads several features in more than one slot, and the order-averaged estimates assign opposing signs to the two copies: one embedding group scores +0.080% in one slot and −0.050% in another, and an impression-count group +0.049% against −0.002%. Since both copies are permuted together, no split of their shared signal is preferred, and only the joint effect is well defined. Reliance also differs by head, sometimes in direction. For the same 189 groups the measured full-coalition effect is removable on Closeup AUC and PR-AUC and on Clickthrough AUC, but load-bearing on Clickthrough PR-AUC: the four attribution sums are +3.079%, +2.348%, +1.996%, and −3.769%, respectively. Individual groups swing much further: one content-type group attributes −0.042% on Closeup AUC, where the model relies on it, and +8.385% on Clickthrough AUC, where it is removable. A per-feature screen on one head cannot see either effect.

Individual-Feature Permutation deltas also overcount additive value. Closeup retains only 47–48% of their sum once the full coalition is measured: 3.079% versus 6.554% for AUC and 2.348% versus 4.909% for PR-AUC. In addition, Clickthrough AUC reverses sign outright, from −10.698% summed to +1.996% measured. Value saturates with coalition size as well. The mean Closeup AUC marginal falls from 0.0327% in positions 1–38 to 0.0030% in positions 152–189, a difference of −0.0297% with 95% CI [−0.0358, −0.0236]. Reverse pairing matches the two bins on Individual-Feature Permutation score, so this decline reflects position in the order and not feature strength. It corroborates the fixed cumulative path’s peak and subsequent decline.

As Figure 6 shows, order averaging broadly preserves the top of the Individual-Feature Permutation ranking but not the zero threshold, with disagreements concentrated in the tail. Across individual orders, fewer than one-third of candidates retain a positive marginal in every order, so agreement in the mean does not imply stable per-feature effects.

These results support FORTE’s staged design: per-feature estimates screen and rank candidates, while grouped perturbation measures the composed candidate set before retraining. Estimates from either a single permutation or 20 coalition orders supported relative ranking but were not sufficiently stable to serve as removal criteria: across the 176 candidates with a positive Individual-Feature Permutation score, 2,496 of 3,520 order-specific marginals (70.9%) retained that direction, an average of 14.2 of 20 orders per group (median 16), and only 53 candidates retained it in all 20. The composed-set effect was estimated more precisely, and grouped perturbation measures it directly in one pooled pass. Order averaging characterizes order sensitivity but does not replace grouped validation; a 10-order run reproduced the 20-order ranking closely (0.975 rank correlation and 95% top-20 overlap).

7 Industrial Applications

FORTE is deployed across Pinterest’s production model fleet. We examine two in depth: a ranking model and a Search Relevance model.

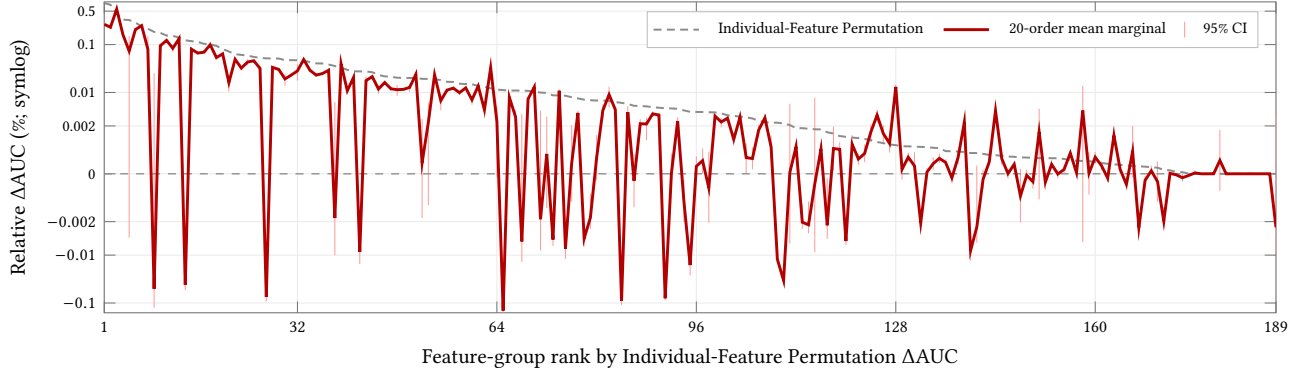


Figure 6: Closeup AUC effects for 189 feature groups, ranked by Individual-Feature Permutation relative Δ AUC. The dashed curve shows individual permutation; the solid curve and bars show the mean and 95% confidence interval over 20 randomized addition orders, computed as t_9 intervals over the ten reverse-pair means. The vertical axis uses a symmetric-logarithmic scale.

Results are reported in each model’s own section. The two models differ in architecture affecting the impact of the deprecations. The Search Relevance model deprecated a larger share of its features yet realized a small parameter reduction because its cross-encoder backbone holds orders of magnitude more parameters.

7.1 Search Relevance

The Search Relevance application uses *segmented evaluation*: a post-inference filter applied at the metric layer that partitions the evaluation set into subpopulations and computes importance scores per segment. Because segmentation is a metric-layer filter, segmented importance reuses the same inference pipeline, perturbation logic, and aggregation framework as global importance. Search Relevance serves multilingual traffic, so the segments of interest are US and non-US markets, and we use segmented A/B results to confirm that deprecated features did not carry localized signal for non-English queries.

The Search Relevance model [26] predicts the relevance of a candidate Pin to a user’s search query on a five-point ordinal scale (L1–L5); the score is consumed alongside engagement predictions to determine the final ranking on Pinterest Search.

The production system is a real-time servable student network distilled from a cross-encoder LLM teacher fine-tuned from a multilingual LLM backbone, ingesting a heterogeneous feature set spanning query-level signals, Pin-level signals (text, graph, and visual embeddings), and query–Pin interaction features; that work describes the full architecture, feature catalog, and distillation pipeline. The checkpoint that FORTE evaluates extends it with a smaller cross-encoder layered over the feed-forward student, strengthening query–Pin interaction modeling within the real-time serving budget. The model serves multilingual traffic globally in real time across more than 45 languages.

Evaluation setup. The combination of heterogeneous tensor types, breadth of input signals, and tight serving budget makes Search Relevance a natural target for FORTE: many features were added incrementally over multiple model iterations, and per-feature serving cost compounds at search QPS (queries per second). We ran FORTE on the production Search Relevance checkpoint, which ingests 111

Table 1: Search Relevance: offline results and deployment outcomes for the three nested candidate sets, one treatment arm each. All three cleared every offline gate; only the online guardrails separated them. Efficiency gains are reported for the launched 48-feature configuration.

Features removed	Teacher acc. Δ	Human acc. Δ	Outcome	Trainable params. Δ	Serving throughput Δ
32	−0.0010	−0.0012	Superseded	—	—
48	−0.0009	+0.0042	Launched*	−2.3%	+2.0%
59	−0.0011	−0.0012	Rejected	—	—

*The launched configuration also realized network and storage savings.

features, computing both permutation and ablation importance. We transformed the five-point relevance output into a single scalar evaluation value and computed the signed normalized importance score from its perturbation-induced change. The candidate gate used an importance-score threshold of 0.01, after which candidates were ranked jointly with a cost analysis. Cut points of 32, 48, and 59 features were chosen to span likelihood of passing A/B. The candidate filter (low mean importance, non-positive fitted slope, and rank stability near the bottom of the feature list) was applied across every head and lookback window. Segments were defined as US and non-US traffic, with segmented results aggregated separately to confirm neutrality across the multilingual scope.

Results. The candidate filter surfaced 59 features for deprecation, drawn from across feature types and entity types (text-match features, some Pin and query embeddings, and several sparse features). Grouped ablation of the candidate sets produced an offline 5-scale accuracy change of 0.1% relative to the production baseline. We then retrained the student model with the candidate features removed and evaluated against the production baseline on the same human-annotated relevance test set used in that work: 5-scale accuracy fell within the offline-neutral band for this model, for all three candidate sets.

The retrained models were then allocated traffic as treatment arms and A/B tested against production using relevance measurements obtained on queries from search traffic. Table 1 reports each arm. We observed no statistically significant regression in search feed relevance for any arm; this evaluation uses sDCG@25 from an

LLM-as-a-judge labeling system [27]. Segment-level results across US and non-US markets were also neutral, indicating that the deprecated features did not carry localized signal for non-English queries despite the model’s multilingual scope.

The offline gates did not separate the three candidate sets: teacher-test accuracy moved by at most 0.0011 in every case, and teacher-test RMSE improved in every case (-0.0003 , -0.0037 , -0.0044). The online stage did. Search processing time improved significantly for all three arms; for the 48-feature arm, page wait time decreased by 0.1696% ($p=0.0012$). The most aggressive 59-feature arm triggered a regression in a secondary guardrail and was not launched; the 48-feature set was selected as the safer operating point. Shipping the smaller model reduced the trainable parameter count by 2.3%, increased serving throughput by 2%, and reduced network ingress and egress for our model serving system, as well as enabling savings on storage and KV feature store costs.

7.2 Pin Ranker

The Pin Ranker is a multi-head L2 ranking model powering a visual-search-like surface that serves roughly half of Pinterest traffic. Given a (user, query Pin, candidate Pin) triplet, the production checkpoint predicts closeups, clickthroughs, repins, and downloads. It also includes a revisitation-based retention head [13].

The architecture combines a transformer-based user-sequence encoder, including lifelong user-history features [28], with a DCNv2 feature-crossing stage and an MMoE module that produces task-specific engagement scores [8]. It has hundreds of millions of dense parameters spanning multi-modal content, numerical, categorical, and user-behavior features. These components enable the system to model rich dependencies among user intent, history, content semantics, and candidate relevance.

Evaluation setup. We ran FORTE on the production Pin Ranker checkpoint, computing the signed normalized change in PR-AUC under permutation across all output heads. Following the deprecation pilot, we now run feature importance daily on the latest production model. This ongoing practice provides a consistent benchmark for the team’s feature utilization and assists in monitoring feature health and adoption over time. Temporal importance was combined with a cost analysis; an audit identified two nested feature sets, of 2 and 8 features, that yielded substantial cost savings at weak measured reliance.

Results. FORTE identified eight features as deprecation candidates: each had a non-negative permutation importance score against PR-AUC across all output heads, indicating no predictive signal the model used. We retrained the model without these features. Both candidate sets cleared every gate. Retraining produced an absolute pairwise-accuracy change of $+0.0001$ for both sets, while the absolute $\Delta\text{NDCG@3}$ was -0.0015 and -0.0010 for the 2- and 8-feature sets, respectively. These results are averaged across output heads and remained within the offline-neutral band. As treatment arms over a three-week online A/B test, neither showed a statistically significant regression in clickthroughs, closeups, or repins. The 8-feature set is a superset of the 2-feature set, so with both arms neutral the larger set was launched and the smaller one superseded.

After launch, FORTE’s final lineage-and-deprecation stage (Figure 1) queried the model registry for remaining consumers of the

Table 2: Pin Ranker: Retraining results for the two nested candidate sets and efficiency gains from launching the 8-feature removal. Throughput is measured on a CUDA Graph benchmark.

Features removed	Pairwise		Outcome	Serving		
	Accuracy Δ^*	NDCG@3 Δ^*		Trainable params. Δ	throughput Δ	Inference cost Δ
2	$+0.0001$	-0.0015	Superseded	—	—	—
8	$+0.0001$	-0.0010	Launched	-29.9%	$+19.8\text{--}25.5\%$	$\sim -10\%$

*Retraining results are absolute changes averaged across output heads.

eight retired features. None remained; after owner review, the corresponding dataset columns were removed and their feature-materialization jobs decommissioned.

Table 2 reports the offline outcomes and resulting efficiency gains. The parameter, throughput, and inference-cost gains follow directly from the smaller model, and together reduced annual infrastructure cost by hundreds of thousands of dollars. Completing deprecation across the feature lineage added dataset-storage and training-loop savings beyond those available from a single-model launch.

Diagnostic use. Beyond its utility in identifying candidates for feature deprecation, the feature-importance report serves as a critical diagnostic mechanism for explaining why a deep learning architectural change had its observed effect. For example, in the residual-connection layer design for non-engagement features in the ranker model, the objective is to reduce the model’s over-reliance on engagement-biased signals and elevate content and quality signals in the decision process. FORTE quantifies the over-reliance directly: in the baseline ranker, the ratio of $\Delta\text{PR-AUC}$ from ablating non-historical features versus historical features sits well below 100%, confirming that the model leans disproportionately on historical signals [8, Fig. 2(a)]. The per-feature $\Delta\text{PR-AUC}$ distribution under the residual variant shifts more negative for non-historical features relative to the baseline, indicating that the residual connection raises the model’s reliance on those features as designed [8, Fig. 6(a)]. Reading the full importance ranking turns FORTE into a tool for implementation validation, not only post-hoc interpretation.

8 Discussion and Conclusion

FORTE combines scalable daily feature evaluation with validation and lineage tracking to support production feature deprecation and infrastructure savings.

8.1 Limitations

FORTE prioritizes scalable feature-deprecation decisions over full causal attribution. This choice creates limitations. Marginal permutation does not achieve conditional PFI [23], so correlated features can receive inflated or deflated importance scores. FORTE also measures first-order importance for individual features or manually configured groups, which can miss features that matter only through interactions.

Temporal feature importance is computed against a model held fixed within each analysis window, making changes interpretable as data drift. New windows refresh to the current production checkpoint, so trend comparisons are scoped to a single checkpoint.

The two-stage downsample provides an approximate shuffling, trading exactness for runtime; we do not claim it is equivalent to a uniform shuffle.

We apply a single random permutation ($R = 1$) to each batch by default. R is exposed as a hyper-parameter, and each additional repeat multiplies the number of perturbation experiments; batch size and total dataset size also bear on the resulting variance. Section 5.1 provides a finite-sample analysis and empirical stability measurements supporting this operating choice; the measurements cover two heads of one production model and do not establish the choice for every model or metric.

We share two examples on Pinterest models; the efficiency findings in particular depend on model characteristics and should not be read as architecture-independent.

The production workflow is designed around these limitations. Candidate features are selected conservatively across all heads, evaluated jointly through grouped perturbation, removed through retraining, and validated through online A/B testing before removal. Future work will focus on automatic interaction discovery, conditional importance estimates for correlated features, and alerts for feature failures and data-quality changes over temporal importance series.

Ethical Considerations

A feature removal that is neutral in aggregate can still degrade quality for subpopulations the aggregate does not reflect. FORTE mitigates this with metric-layer segmented evaluation, applied in this work as US versus non-US traffic. Two coarse geographic segments cannot resolve effects across the more than 45 languages the Search Relevance model serves, and can conceal within-segment harm; this is one mitigation with limited resolution, not a fairness result. Deployments should monitor at the granularity their traffic warrants, whether per-language, per-geography, or per-traffic-slice, chosen for the population actually served.

We note a risk the framework itself introduces. Deprecation errors surface as aggregate metric regressions, which is precisely the measurement that hides distributional harm; making removal cheap and routine increases the throughput of a decision whose failure mode is unevenly distributed.

Removing unused features reduces the compute, storage, and serving footprint of a model. Our measured effects are a reduced parameter count, higher serving throughput, and lower storage footprint; removal is expected to reduce energy use as a consequence, but we did not measure power, accelerator-hours, or emissions and make no quantitative energy claim.

References

- [1] Shivani Agarwal, Sarel Har-Peled, and Dan Roth. 2005. A Uniform Convergence Bound for the Area Under the ROC Curve. In *Proceedings of the Tenth International Workshop on Artificial Intelligence and Statistics*. 1–8.
- [2] Mohamed Adel Attia and Ravi Tandon. 2016. On the Worst-Case Communication Overhead for Distributed Data Shuffling. In *2016 54th Annual Allerton Conference on Communication, Control, and Computing*. IEEE, 961–968. doi:10.1109/ALLERTON.2016.7852338
- [3] Leo Breiman. 2001. Random Forests. *Machine Learning* 45, 1 (2001), 5–32. doi:10.1023/A:1010933404324
- [4] Javier Castro, Daniel Gómez, and Juan Tejada. 2009. Polynomial Calculation of the Shapley Value Based on Sampling. *Computers & Operations Research* 36, 5 (2009), 1726–1730. doi:10.1016/j.cor.2008.04.004
- [5] Ian Covert and Su-In Lee. 2021. Improving KernelSHAP: Practical Shapley Value Estimation Using Linear Regression. In *Proceedings of the 24th International Conference on Artificial Intelligence and Statistics (Proceedings of Machine Learning Research, Vol. 130)*. 3457–3465.
- [6] Ian Covert, Scott Lundberg, and Su-In Lee. 2020. Understanding Global Feature Contributions With Additive Importance Measures. In *Advances in Neural Information Processing Systems*, Vol. 33.
- [7] Ian Covert, Scott Lundberg, and Su-In Lee. 2021. Explaining by Removing: A Unified Framework for Model Explanation. *Journal of Machine Learning Research* 22, 209 (2021), 1–90.
- [8] Saeed Ebrahimi, Weijie Jiang, Jaewon Yang, Olafur Gudmundsson, Yucheng Tu, and Huizhong Duan. 2025. Warmer for Less: A Cost-Efficient Strategy for Cold-Start Recommendations at Pinterest. arXiv:2512.17277. arXiv:2512.17277 [cs.LG] doi:10.48550/arXiv.2512.17277 Submitted to The Web Conference 2026.
- [9] Aaron Fisher, Cynthia Rudin, and Francesca Dominici. 2019. All Models are Wrong, but Many are Useful: Learning a Variable’s Importance by Studying an Entire Class of Prediction Models Simultaneously. *Journal of Machine Learning Research* 20, 177 (2019), 1–81.
- [10] Fabian Fumagalli, Maximilian Muschalik, Eyke Hüllermeier, and Barbara Hammer. 2022. Incremental Permutation Feature Importance (iPFI): Towards Online Explanations on Data Streams. arXiv:2209.01939. arXiv:2209.01939 [cs.LG]
- [11] Giles Hooker, Lucas Mentch, and Siyu Zhou. 2021. Unrestricted Permutation forces Extrapolation: Variable Importance Requires at least One More Model, or There Is No Free Variable Importance. *Statistics and Computing* 31, 6, Article 82 (2021). doi:10.1007/s11222-021-10057-z
- [12] Sara Hooker, Dumitru Erhan, Pieter-Jan Keldermans, and Been Kim. 2019. A Benchmark for Interpretability Methods in Deep Neural Networks. In *Advances in Neural Information Processing Systems*, Vol. 32.
- [13] Weijie Jiang, Armando Ordorica, Jaewon Yang, Olafur Gudmundsson, Yucheng Tu, and Huizhong Duan. 2026. Save, Revisit, Retain: A Scalable Framework for Enhancing User Retention in Large-Scale Recommender Systems. *Proceedings of the AAAI Conference on Artificial Intelligence* 40, 47 (2026), 39996–40005. doi:10.1609/aaai.v40i47.41434
- [14] Jing Lei, Max G’Sell, Alessandro Rinaldo, Ryan J. Tibshirani, and Larry Wasserman. 2018. Distribution-Free Predictive Inference for Regression. *J. Amer. Statist. Assoc.* 113, 523 (2018), 1094–1111. doi:10.1080/01621459.2017.1307116
- [15] Scott M. Lundberg, Gabriel G. Erion, and Su-In Lee. 2018. Consistent Individualized Feature Attribution for Tree Ensembles. arXiv:1802.03888. arXiv:1802.03888 [cs.LG]
- [16] Scott M. Lundberg and Su-In Lee. 2017. A Unified Approach to Interpreting Model Predictions. In *Advances in Neural Information Processing Systems*, Vol. 30. Colin McDiarmid. 2002. Concentration for Independent Permutations. *Combinatorics, Probability and Computing* 11, 2 (2002), 163–178. doi:10.1017/S0963548301005089
- [17] Maximilian Muschalik, Fabian Fumagalli, Barbara Hammer, and Eyke Hüllermeier. 2023. iSAGE: An Incremental Version of SAGE for Online Explanation on Data Streams. arXiv:2303.01181. arXiv:2303.01181 [cs.LG]
- [18] Kristin K. Nicodemus, James D. Malley, Carolin Strobl, and Andreas Ziegler. 2010. The behaviour of random forest permutation-based variable importance measures under predictor correlation. *BMC Bioinformatics* 11, Article 110 (2010). doi:10.1186/1471-2105-11-110
- [19] D. Sculley, Gary Holt, Daniel Golovin, Eugene Davydov, Todd Phillips, Dietmar Ebner, Vinay Chaudhary, Michael Young, Jean-François Crespo, and Dan Dennison. 2015. Hidden Technical Debt in Machine Learning Systems. In *Advances in Neural Information Processing Systems*, Vol. 28.
- [20] Lloyd S. Shapley. 1953. A Value for n -Person Games. In *Contributions to the Theory of Games, Volume II*, Harold W. Kuhn and Albert W. Tucker (Eds.). Annals of Mathematics Studies, Vol. 28. Princeton University Press, Princeton, NJ, 307–317. doi:10.1515/9781400881970-018
- [21] Avanti Shrikumar, Peyton Greenside, and Anshul Kundaje. 2017. Learning Important Features Through Propagating Activation Differences. In *Proceedings of the 34th International Conference on Machine Learning (ICML’17)*. 3145–3153.
- [22] Carolin Strobl, Anne-Laure Boulesteix, Thomas Kneib, Thomas Augustin, and Achim Zeileis. 2008. Conditional variable importance for random forests. *BMC Bioinformatics* 9, Article 307 (2008). doi:10.1186/1471-2105-9-307
- [23] Pascal Sturmfels, Scott Lundberg, and Su-In Lee. 2020. Visualizing the Impact of Feature Attribution Baselines. *Distill* (2020). doi:10.23915/distill.00022
- [24] Mukund Sundararajan, Ankur Taly, and Qiqi Yan. 2017. Axiomatic Attribution for Deep Networks. In *Proceedings of the 34th International Conference on Machine Learning (ICML’17)*. 3319–3328.
- [25] Han Wang, Mukuntha Narayanan Sundararaman, Onur Gungor, Yu Xu, Krishna Kamath, Rakesh Chalasani, Kurchi Subhra Hazra, and Jinfeng Rao. 2024. Improving Pinterest Search Relevance Using Large Language Models. In *Proceedings of the 3rd International Workshop on Industrial Recommendation Systems (IRS) at CIKM 2024*. arXiv:2410.17152 [cs.LG] doi:10.48550/arXiv.2410.17152
- [26] Han Wang, Alex Whitworth, Pak Ming Cheung, Zhenjie Zhang, and Krishna Kamath. 2025. LLM-based Relevance Assessment for Web-Scale Search Evaluation at Pinterest. In *Proceedings of the RecSys 2025 EARL Workshop*.

- arXiv:2509.03764 [cs.LR] doi:10.48550/arXiv.2509.03764
- [28] Xue Xia, Saurabh Vishwas Joshi, Kousik Rajesh, Kangnan Li, Yangyi Lu, Nikil Pancha, Dhruvil Deven Badani, Jiajing Xu, and Pong Eksombatchai. 2025. Trans-Act V2: Lifelong User Action Sequence Modeling on Pinterest Recommendation. *arXiv preprint arXiv:2506.02267* (2025). doi:10.48550/arXiv.2506.02267
- [29] Lijie Xu, Shuang Qiu, Binhang Yuan, Jiawei Jiang, Cedric Renggli, Shaoduo Gan, Kaan Kara, Guoliang Li, Ji Liu, Wentao Wu, Jieping Ye, and Ce Zhang. 2022. In-Database Machine Learning with CorgiPile: Stochastic Gradient Descent without Full Data Shuffle. In *Proceedings of the 2022 International Conference on Management of Data*. ACM, 1286–1300. doi:10.1145/3514221.3526150