LEARNING THE COUNT, KEEPING THE UNIT: TASK-ALGEBRA HEADS FOR RELATIONAL REGRESSION

Anonymous authors
Paper under double-blind review

ABSTRACT

Relational prediction tasks define their targets through programs: a future sum, a count, or an event condition. These programs can specify how a model should predict, as well as what it should predict. We develop TASK-ALGEBRA, a compiler for a restricted aggregation-query fragment that extracts a query-time unit factor, transforms the training target, and reconstructs predictions with that factor kept exact. For a static entity-owned unit, a future sum becomes $Y = sN$; monetary absolute error becomes exactly s-weighted count error. A conditional-median approximation result explains why this parameterization can save finite model capacity even when both models receive identical inputs. On four chronological Amazon item-LTV development evaluations with matched 67-feature CatBoost heads, the factorized head reduces mean absolute error from 44.481 to 42.071, a 5.42% reduction, with a category-clustered gain interval of $[1.133, 4.466]$. Larger trees, numerical loss controls, a history-selected population, and continuous decoders reveal how approximation and optimization contribute. The resulting account connects a task's aggregation semantics to an executable prediction head and to the regimes in which keeping a known unit outside the learner is useful.

1 INTRODUCTION

A ten-dollar product and a hundred-dollar product may each attract three future reviews. Their price-weighted review targets are thirty and three hundred dollars. A model given price as an input can learn this multiplication, but a finite tree ensemble may spend partitions approximating a unit conversion that the target program already specifies. The query has already told us how the unit enters the answer.

This distinction is particularly concrete in relational prediction. A task is often a query over future database records: sum a value over transactions, count interactions within a horizon, or test whether an event occurs. RelBench makes these tasks explicit through temporal relational data and task definitions (Robinson et al., 2024). Declarative languages further turn predictive tasks into executable specifications (Kocijan et al., 2026; Kolesnichenko et al., 2026). Once a program is available, it can do more than generate labels. Its joins, keys, aggregation, and time predicates can constrain the model's output.

Consider a future sum of product price over review rows. If the query joins each review to the same product and product price is a declared static attribute of the forecast entity, then the sum is exactly price times the future number of review rows. The learner can predict the count, while a small deterministic head applies the known price. The training objective must change with the target: treating all count errors equally would optimize a different metric. For monetary mean absolute error (MAE), the correct sample weight is the price itself.

We develop TASK-ALGEBRA, an aggregation-head compiler that combines these two deductions. It checks the lineage and temporal availability of the sum operand, distinguishes a static entity factor from an event-varying unit, and emits a target transform, a loss weight, an inverse prediction map, and a strictly historical mirror of the task query. The ordinary feature matrix and decoder can remain the same. Figure 1 shows the complete route from a task program to a fitted predictor.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

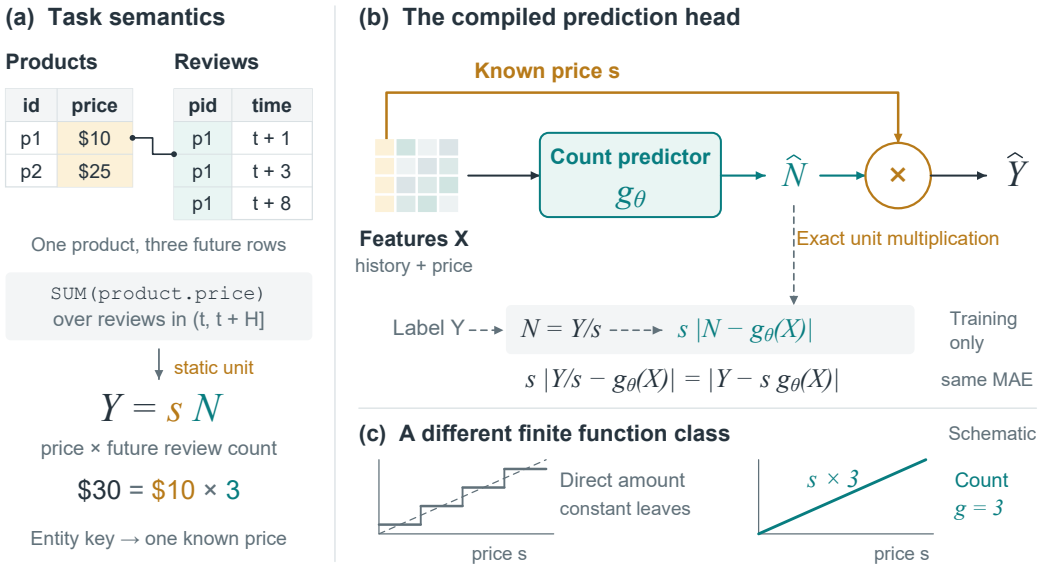


Figure 1: **Learn the count, keep the unit.** (a) A static, entity-owned price factors a future sum over review rows into $Y = sN$. (b) The decoder receives historical features, including price; the known price also bypasses the decoder and multiplies its count prediction. Dashed arrows carry training supervision, with price-weighted count error equal to monetary MAE before the common support projection. (c) For an illustrative conditional count median of three, a constant count head yields a linear amount function, while direct constant leaves approximate that relationship.

Our central question is when this use of task semantics improves a finite learner. With the unit already present in the input, the algebra changes which functions are easy to represent and which residuals the optimizer sees. We make that claim concrete in three ways. First, we derive a conditional-median capacity separation for direct piecewise-constant prediction. Second, we implement query analysis and head execution, including temporal and population checks. Third, we evaluate direct and factorized heads with fixed input interfaces across chronological origins, stronger capacities, richer inputs, and different decoder families.

The main result is a consistent tree-model benefit on Amazon item-LTV: with the same 67 input features and the same CatBoost budget, the factorized head lowers MAE at all four evaluation origins. The larger model and continuous-decoder comparisons make the explanation more precise. A unit factor is most valuable when it removes a costly scale relationship from a restricted predictor; its benefit depends on the function class and on the numerical implementation of the loss. This makes task algebra an experimentally testable inductive bias.

The work contributes an executable aggregation-head compiler, an approximation argument tied to conditional medians, and a temporal empirical study that separates output representation, metric weighting, and optimizer scale effects. Together these components provide a way to ask a concrete modeling question: *which parts of a target should be learned, and which parts are already specified by the task?*

2 From a Target Query to a Prediction Head

2.1 Temporal Relational Tasks

Let e be a forecast entity, t its query time, and H the prediction horizon. A task program defines both a query population $\mathcal{P}_t$ and an aggregate over future event rows. For a sum target,

$$Y_{e,t} = \sum_{r:\, r \to e,\ t < \tau_r \le t+H} v_r. \tag{1}$$

Inputs $X_{e,t}$ are computed from records strictly before t in our conservative experimental track. A labeled source query is eligible for fitting only once its entire horizon has matured: $t_i + H < t$. These conditions concern different objects—input records and training outcomes—and both are enforced.

Static entity factor. Suppose the sum operand resolves through a key-preserving join to a declared static entity attribute $s_e > 0$. The schema then supplies the functional dependency $e \to s_e$, and every aggregated row has the same operand. Equation 1 becomes

$$Y_{e,t} = s_e N_{e,t}, \qquad N_{e,t} = \#\{r : r \to e,\ t < \tau_r \le t + H\}. \tag{2}$$

In Amazon item-LTV, s_e is the product-table price and N counts future review rows. The quantity being forecast is the benchmark's price-weighted review aggregate.

Event-varying operand. If the unit belongs to an event row, a static functional dependency does not hold. H&M item-sales sums transaction-specific prices. A positive historical estimate $u_{e,t}$ can still define a normalized regression head, but Y/u is then a normalized amount rather than an exact event count. The compiler emits an estimated-unit candidate and requires an explicit historical estimator.

2.2 Metric-aligned target transformation

For any positive query-time factor s and any query-time anchor B, define

$$Z = \frac{Y - B}{s}, \qquad \widehat{Y} = B + s\, g_\theta(X). \tag{3}$$

For a power error with $q \ge 1$,

$$|Y - \widehat{Y}|^q = s^q |Z - g_\theta(X)|^q. \tag{4}$$

Thus MAE uses target $(Y - B)/s$ and sample weight s; squared error uses weight s^2. For the exact static case with $B = 0$, the transformed target is N. This pointwise identity holds before fitting and for every unprojected prediction. Population support is applied as a common inference projection after reconstruction. The identity also explains why unweighted count regression is a distinct ablation.

The predictor retains price in its input: count demand may depend on price even though multiplication is handled outside the decoder. Head comparisons fix the input interface, decoder capacity, and population support. On a supplied future-event-conditioned population, $N \ge 1$ gives a common price floor. On a population selected only from historical activity, future count may be zero, so the common lower bound is zero. The population rule is an input to the head specification and is never silently changed by the compiler. Section 4.1 specifies the implemented source-row eligibility rules.

2.3 Executable compilation

The implemented analyzer supports a single-column `SUM`, optionally wrapped in `COALESCE`, over a directly joined future event relation. It performs the following operations:

1. Resolve aliases and the sum operand to a unique table and column; identify the forecast entity key and event foreign key.
2. Check the event–entity join and the future interval against the declared horizon. Read the entity attribute's temporal declaration.
3. Emit an exact static-factor recipe when the operand is entity-owned and declared static; otherwise emit an estimated-unit candidate.
4. Generate the target transform, metric weight, inverse map, and population-specific support rule. Emit a historical sum query over a caller-supplied roster, using $[t - H, t)$.

The recipe is independent of training labels. It can be attached to an existing tabular backend through three operations: transform a training target, supply its sample weights, and invert predictions. The recipe records whether its count interpretation is exact. Our checks cover three real task manifests and rejection cases including ambiguous lineage, an unmatched foreign key, an inconsistent horizon, and a temporally varying attribute falsely proposed as static. Historical queries were executed on raw Amazon and H&M relations and agreed with independently generated amount features on 1,000 entities per task (maximum discrepancy 0 and 4.2×10^{-14}). The appendix gives the supported syntax and replay checks.

3 WHY KEEPING THE UNIT CAN SAVE MODEL CAPACITY

Under absolute-error risk, any conditional median is Bayes-optimal (Koenker & Bassett, 1978; Gneiting, 2011). If $s > 0$ is already available in X, multiplying a conditional median of N by s gives a conditional median of $Y = sN$. Direct and factorized prediction therefore have the same unrestricted optimum. Their finite function classes are different: a constant count leaf becomes a linear function of the known unit after reconstruction, whereas a direct monetary leaf remains constant.

The following result captures this distinction for noisy counts. It requires a common positive median within an unresolved input cell, not independence of count and price.

Proposition 1 (A conditional-median capacity separation)**.** *Let P be uniform on $[a, b]$, with $0 < a < b$, and let $Y = PN$, where N is a nonnegative integer count with $\mathbb{E}[N] < \infty$. Suppose N has the same integer conditional median $m > 0$ for almost every p, with a uniform margin $\gamma > 0$:*

$$\Pr(N \le m - 1 \mid P = p) \le \tfrac{1}{2} - \gamma, \quad \Pr(N \le m \mid P = p) \ge \tfrac{1}{2} + \gamma. \tag{5}$$

A constant count head $g = m$ followed by multiplication attains Bayes MAE. Any direct piecewise-constant monetary predictor with at most L price intervals has excess MAE at least

$$\frac{\gamma m(b - a)}{2L}. \tag{6}$$

Proof sketch. The conditional absolute-risk slopes on either side of m have magnitude at least 2γ. Hence its excess risk is at least $2\gamma|v - m|$. Multiplying by p gives a monetary excess-risk bound $2\gamma|f(p) - pm|$. On a price interval of length d, the best constant approximation to pm has integrated absolute error $md^2/4$. Summing the intervals and using $\sum_j d_j^2 \ge (b - a)^2/L$ proves Equation 6. Appendix A gives the full proof and the effect of the benchmark price floor.

This result predicts an interaction between scale variation and effective capacity. A wider unresolved unit range and a larger positive count median make direct prediction harder; more price intervals reduce that burden. When the conditional count median is zero, a zero monetary leaf already represents the optimum. A continuous decoder can also express multiplication more efficiently than a constant-leaf model. Section 4 measures these interactions in fitted predictors. Here L counts intervals of the final predictor, not leaves in one constituent tree.

An estimated unit has a different error mechanism. For $Y = N\mu_{\text{future}}$ and a historical estimate $u > 0$,

$$|Y - ug| \le u|N - g| + N|\mu_{\text{future}} - u|. \tag{7}$$

The second term measures unit drift. Training on Y/u with weight u still exactly optimizes monetary MAE within the chosen head class; interpreting that target as count, however, depends on unit stability. This distinguishes algebraic correctness from estimation convenience and motivates the static–dynamic comparison.

4 EXPERIMENTS: IDENTIFYING WHAT THE HEAD CHANGES

We organize the study around three questions: does moving the known unit outside a finite decoder improve prediction; which part of the gain comes from representation, metric weighting, or optimization; and how does the comparison change with the decoder and query population?

4.1 TASKS, TEMPORAL PROTOCOL, AND INPUT INTERFACES

The main task is RelBench Amazon item-LTV: a product's static price summed once per review row in the next 91 days. The official task roster contains products with at least one future review. We evaluate four late TRAIN origins, October 2014 through July 2015, using the latest 16 source origins whose 91-day outcomes have strictly matured before each query. These are chronological development evaluations. Event inputs satisfy $\tau < t$, including windows $[t - w, t)$. No evaluation-origin labels enter the fitted source data.

Table 1: **Static-factor Amazon item-LTV, 67 inputs.** CatBoost500/depth6, strictly historical features, official query population. Smaller MAE is better; gain is direct minus factor.

Origin	Queries	Direct amount	Unit × count	Gain
2014-10-02	182,220	45.758	43.377	2.380
2015-01-01	190,059	44.464	44.010	0.454
2015-04-02	179,118	45.583	41.647	3.936
2015-07-02	177,689	42.119	39.252	2.867
Equal-origin mean		44.481	42.071	2.410

The main interface has 67 columns: product price and category/brand, historical amount and count windows, rating, verified-review and text-length summaries, and reviewer summaries. Both heads receive the same columns, including price and past monetary amount. CatBoost (Prokhorenkova et al., 2018) uses 500 depth-6 trees, learning rate 0.05, regularization 3, and seed 42, with no development early stopping. The main tree and continuous comparisons use identical positive-price source rows within each pair, a common zero rule for nonpositive-price queries, and the same official-roster price floor for positive-price queries. Appendix B records the exact counts, inherited auxiliary-control eligibility, and neural training policy.

We report monetary MAE and direct-minus-factor paired gains, averaging origins equally. The 95% intervals jointly resample 57 broad book categories across origins, with 3,000 replicates. They describe category variation conditional on the fitted models; the chronological origins remain distinct evaluation periods.

4.2 A FINITE TREE BENEFITS FROM KEEPING THE UNIT EXACT

Table 1 gives the four-origin result. The count head improves at every origin, reducing equal-origin MAE from 44.481 to 42.071, or 5.42%. Its mean paired gain is 2.410, with interval $[1.133, 4.466]$. The January effect is smaller; the other three origins supply most of the reduction. The same known price is available to both heads, so this comparison concerns how that information enters prediction.

Increasing capacity to 1,000 depth-8 trees gives October direct MAE 43.774. Its corresponding factor head reaches 42.362: a smaller gap, still positive at 1.413 $[0.410, 3.178]$. This follows the capacity argument's direction while comparing both heads within the larger decoder; the auxiliary source policy is recorded in the appendix.

Representation and weighting are separate operations. With a compact 21-column interface at the October origin, direct monetary MAE is 46.490. Predicting unweighted count and multiplying by price gives 45.340; using the metric-derived price weight gives 43.731. The two steps reduce MAE by 1.150 $[0.189, 2.739]$ and 1.609 $[0.734, 2.709]$, respectively. A useful output representation and the correct monetary objective contribute separately.

4.3 SCALE ALSO ENTERS THE OPTIMIZER

CatBoost's quantile derivative includes a residual tolerance δ, defaulting to 10^{-6} in the inspected implementation.[1] A source-only probe changes monetary units by dividing targets by 12.99 and then maps forecasts back. With the default tolerance, the mean prediction departure is 3.537; scaling the tolerance with the target unit reduces it to 6.4×10^{-7}. Algebraically equivalent objectives can therefore follow different numerical training paths.

We test this mechanism in full forecasting fits using a fixed tiny positive tolerance 10^{-12} and identical positive-price source rows. Across October and January, the factor head retains mean gain 1.178 $[0.290, 2.728]$. The recorded $\delta = 0$ control gives 0.507 $[-0.328, 1.944]$. Zero tolerance also changes exact-tie subgradients; it is a different optimization control. These results identify both function-class structure and numerical loss conventions as relevant to the observed head comparison. Appendix C supplies all pairs and source-only diagnostics.

[1] CatBoost 1.2.10, `error_functions.h`, fixed commit `b1bd2a6`.

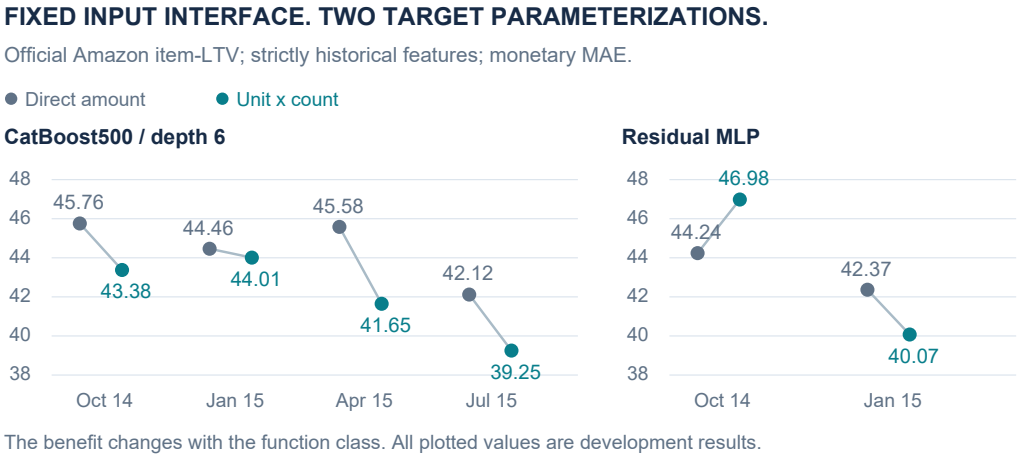


Figure 2: **The decoder changes the value of the head.** Official-roster Amazon evaluations with the same 67-column interface. The tree comparison improves at four origins; the residual MLP comparison reverses direction between its two evaluated origins. Values are read directly from archived paired results.

4.4 Decoder and population determine the practical comparison

A residual MLP provides a continuous function class, inspired by established tabular residual-network design (Gorishniy et al., 2021). It has width 256, three residual blocks, and source-only epoch selection. On the official roster, its direct/factor MAEs are 44.240/46.984 in October and 42.367/40.074 in January. The equal-origin factor gain is -0.226 $[-0.759, 0.287]$. The strong continuous direct model changes the comparison substantially: known multiplication is especially useful when the underlying decoder has difficulty expressing it.

We also define a history-selected population: positive-price products with at least one review in the preceding 91 days. This roster can be constructed at prediction time, includes future zeros (35.30% and 34.12%), and uses only a zero floor. Within CatBoost67 with $\delta = 0$, the factor head improves by 5.884 $[3.346, 9.503]$. The continuous ordinary model, however, achieves 37.279/35.572, below the factor tree's 39.237/38.558. Within the continuous pair, factor MAE is 37.706/34.493, giving gain 0.326 $[0.021, 0.723]$; a common nearest-count support projection yields 0.253 $[-0.052, 0.632]$. Table 4 in the appendix reports the wider capacity controls. These contrasts distinguish a favorable same-decoder parameterization from the choice of the best predictor.

The scale signature can be measured. On the saved main tree models, we multiply price, historical monetary inputs, and the target by $k \in \{0.5, 2\}$ while keeping review counts fixed. The normalized departure from scale-equivariant prediction is 8.52–9.09 for direct prediction and 0.97–2.09 for the factor head across October and January. The count decoder still receives price, so this is measured approximate equivariance. The diagnostic is an artificial unit transport, not a causal price intervention. An exploratory price-trim analysis also locates much of the monetary gain in the tail: removing the highest 1% of prices leaves gains of 0.463 in October and -0.213 in January. This connects the aggregate result to the unit variation that motivates the method.

Static factors and estimated units have different roles. H&M item-sales provides a seven-day sum of event-specific prices. Under strictly historical 54-column inputs, adding a historical unit to an ordinary residual head changes equal-origin MAE by only 1.13×10^{-5} $[-1.86 \times 10^{-4}, 2.91 \times 10^{-4}]$. Table 6 reports the direct, residual, unit, and historical-reference heads. Here the unit must be estimated, and the task algebra leads to a different empirical regime, as Equation 7 anticipates.

5 Related work

Relational prediction and task specification. Relational deep learning represents database rows and primary–foreign key relations as a temporal heterogeneous graph (Fey et al., 2024); RelBench supplies temporal tasks and evaluation infrastructure (Robinson et al., 2024). PQL and RTGL make predictive-task semantics declarative (Kocijan et al., 2026; Kolesnichenko et al., 2026). PQL also infers broad task types to enable appropriate model choice. Relational foundation models scale learning across databases (Hudovernik et al., 2026). Our focus is algebraic parameterization within aggregate regression. Keys, operand lineage, time predicates, and roster support together determine which output transform is justified.

Known scales, aggregate decomposition, and residual prediction. Frequency–severity models decompose aggregate insurance costs and use exposure in their prediction objectives (Holvoet et al., 2025). Persistence-based residual forecasting retains an observed reference outside the learned correction (Haugsdal et al., 2022). These precedents motivate treating available scale and anchors explicitly. The algebraic factorization and residual connection are established modeling ideas. Our contribution connects an aggregation program to an executable metric-preserving recipe and studies the resulting finite-decoder mechanism.

Tabular decoders and decision functionals. Boosted trees such as LightGBM and CatBoost remain important tabular modeling approaches (Ke et al., 2017; Prokhorenkova et al., 2018). Residual networks provide a useful continuous comparison (Gorishniy et al., 2021). Quantile regression and forecast evaluation establish the relationship between absolute loss and conditional medians (Koenker & Bassett, 1978; Gneiting, 2011). We use that relationship to derive a restricted approximation separation, then test output representation, weighting, and numerical optimization with fixed relational inputs.

6 Conclusion

A target query contains modeling information. When a future sum repeats a known entity-owned unit, a predictor can learn the remaining count and keep that unit exact. Task-Algebraturns this observation into a checked target, weight, and inverse map. The conditional-median result explains a finite-capacity benefit, and the experiments connect it to tree capacity, monetary weighting, and optimizer conventions. The useful question is which part of a target still needs to be learned.

The present evidence is concentrated on a static-factor task and chronological development origins. Continuous decoders substantially change the comparison, and the estimated-unit transfer supplies no reliable independent gain over anchoring. The compiler covers a restricted query fragment, assumes declared metadata availability, and uses task-specific feature and training wrappers. Broader published baselines, multiple seeds, and frozen official validation/test evaluation remain necessary to establish submission-level predictive superiority. These extensions can test the current mechanism while preserving its central distinction: task algebra specifies a function class; its predictive value depends on the learner and the population.

Reproducibility statement

The appendix specifies the task populations, chronological source policy, input interfaces, decoders, and paired uncertainty estimates. The source package includes the figure generator, an experimental evidence ledger, and a reference audit. Each displayed empirical value is linked to an existing result artifact. The implemented query analyzer and prediction-head code are identified in Appendix E; raw benchmark data and model checkpoints are not embedded in this manuscript package.

Ethics statement

The experiments use public relational benchmark data. Review-based monetary targets are defined by benchmark queries and are not interpreted as observed purchases or realized business revenue.

Population eligibility and temporal information rules are stated explicitly. The method is intended for predictive modeling; it does not identify the causal effect of changing a price or an event rate.

AI USE STATEMENT

Generative AI tools assisted research ideation, experiment design, implementation, result interpretation, mathematical formulation, proof drafting, literature retrieval, figure creation, and manuscript writing. Empirical statements were checked against recorded experimental outputs, and bibliographic records were checked against primary sources by a separate reference-verification agent. Automated checks verify source-file consistency and citation keys. This AI-assisted manuscript has undergone human author review. The submitting authors take responsibility for its final content.

REFERENCES

Matthias Fey, Weihua Hu, Kexin Huang, Jan Eric Lenssen, Rishabh Ranjan, Joshua Robinson, Rex Ying, Jiaxuan You, and Jure Leskovec. Position: Relational deep learning – graph representation learning on relational databases. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 13592–13607. PMLR, 2024. URL https://proceedings.mlr.press/v235/fey24a.html.

Tilmann Gneiting. Making and evaluating point forecasts. *Journal of the American Statistical Association*, 106(494):746–762, 2011. doi: 10.1198/jasa.2011.r10138. URL https://doi.org/10.1198/jasa.2011.r10138.

Yury Gorishniy, Ivan Rubachev, Valentin Khrulkov, and Artem Babenko. Revisiting deep learning models for tabular data. In *Advances in Neural Information Processing Systems*, volume 34, pp. 18932–18943. Curran Associates, Inc., 2021. URL https://proceedings.neurips.cc/paper/2021/hash/9d86d83f925f2149e9edb0ac3b49229c-Abstract.html.

Espen Haugsdal, Erlend Aune, and Massimiliano Ruocco. Persistence initialization: A novel adaptation of the transformer architecture for time series forecasting, 2022. URL https://arxiv.org/abs/2208.14236.

Freek Holvoet, Katrien Antonio, and Roel Henckaerts. Neural networks for insurance pricing with frequency and severity data: A benchmark study from data preprocessing to technical tariff, 2025. URL https://arxiv.org/abs/2310.12671v4. Version 4.

Valter Hudovernik, Federico López, Vid Kocijan, Akihiro Nitta, Jan Eric Lenssen, Jure Leskovec, and Matthias Fey. KumoRFM-2: Scaling foundation models for relational learning, 2026. URL https://arxiv.org/abs/2604.12596v1.

Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. LightGBM: A highly efficient gradient boosting decision tree. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper/2017/hash/6449f44a102fde848669bdd9eb6b76fa-Abstract.html.

Vid Kocijan, Jinu Sunil, Jan Eric Lenssen, Viman Deb, Xinwei He, Federico Reyes Gomez, Matthias Fey, and Jure Leskovec. Predictive query language: A domain-specific language for predictive modeling on relational databases, 2026. URL https://arxiv.org/abs/2602.09572v3. Version 3; presented at TaDA@VLDB 2026.

Roger Koenker and Gilbert Bassett, Jr. Regression quantiles. *Econometrica*, 46(1):33–50, 1978. doi: 10.2307/1913643. URL https://doi.org/10.2307/1913643.

Oleksii Kolesnichenko, Jakub Peleška, and Gustav Šír. Relational task generation language: A declarative specification framework for relational deep learning, 2026. URL https://arxiv.org/abs/2609.01292v1. Accepted at MLG 2026.

Liudmila Prokhorenkova, Gleb Gusev, Aleksandr Vorobev, Anna Veronika Dorogush, and Andrey Gulin. CatBoost: Unbiased boosting with categorical features. In *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018. URL https://proceedings.neurips.cc/paper/2018/hash/14491b756b3a51daac41c24863285549-Abstract.html.

Joshua Robinson, Rishabh Ranjan, Weihua Hu, Kexin Huang, Jiaqi Han, Alejandro Dobles, Matthias Fey, Jan E. Lenssen, Yiwen Yuan, Zecheng Zhang, Xinwei He, and Jure Leskovec. RelBench: A benchmark for deep learning on relational databases. In *Advances in Neural Information Processing Systems*, volume 37, pp. 21330–21341. Curran Associates, Inc., 2024. doi: 10.52202/079017-0672. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/25cd345233c65fac1fec0ce61d0f7836-Abstract-Datasets_and_Benchmarks_Track.html.

A PROOF AND THE POPULATION SUPPORT RULE

Let $r_p(v) = \mathbb{E}[|N - v| \mid P = p]$. The two conditional CDF inequalities in Proposition 1 imply right slopes at least 2γ for $v > m$ and left slopes at most -2γ for $v < m$. Convexity of absolute risk and integration of these slopes give

$$r_p(v) - r_p(m) \geq 2\gamma|v - m|. \tag{8}$$

For any monetary prediction $f(p)$, substitute $v = f(p)/p$ and multiply by $p > 0$:

$$\mathbb{E}[|PN - f(P)| - |PN - Pm| \mid P = p] \geq 2\gamma|f(p) - pm|. \tag{9}$$

On an interval $[l, u]$ of length d, the constant minimizing $\int_l^u |c - mp|\, dp$ is $c = m(l + u)/2$ and the minimum is $md^2/4$. For at most L intervals covering $[a, b]$, Jensen's inequality gives $\sum_j d_j^2 \geq (b - a)^2/L$. Integrating over the uniform price distribution establishes the lower bound. The factorized prediction Pm attains the conditional median at every price and hence Bayes MAE.

Effect of a common official-roster price floor. The official Amazon roster ensures at least one future review. The supported direct prediction in a price cell is then $\max(c, p)$. If $m = 1$, taking $c = 0$ already produces the Bayes line p; the constant-predictor separation does not apply to this supported class. For integer $m \geq 2$, the geometric minimum remains $m(u - l)^2/4$. For $c \leq l$, the forecast is just p. For $l < c < u$, differentiating $\int_l^u |\max(c, p) - mp| dp$ gives a negative derivative: either $l - c < 0$, or $(2/m - 1)c - l < 0$, according to the location of c/m. Thus the optimum is at or above u. In that region it is the ordinary constant optimum $m(l + u)/2 \geq u$. The same lower bound follows for $m \geq 2$. On a history-selected population the common floor is zero; a positive count median is unaffected by it.

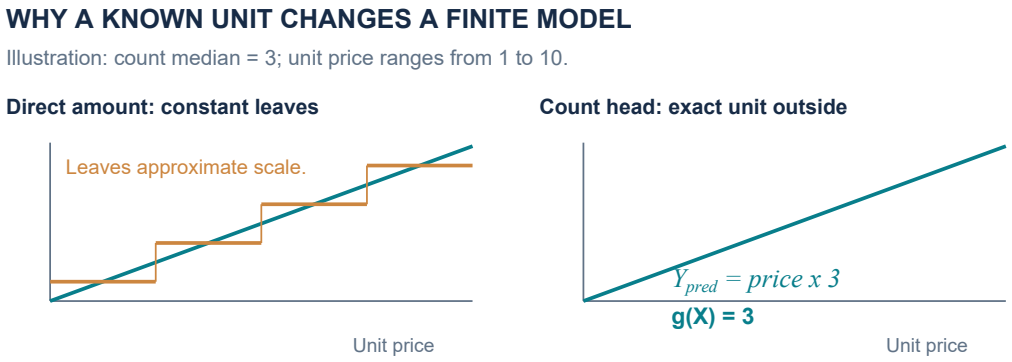


Figure 3: **A schematic of the approximation mechanism.** With a common count median of three, a constant count prediction followed by the known unit represents the Bayes line. A direct constant-leaf function approximates that line using intervals. This is an illustration of the proposition, not a fitted experimental result.

B EXPERIMENTAL SPECIFICATION

Source origins and row eligibility. Each Amazon evaluation uses the most recent 16 official TRAIN source origins satisfying $t_i + 91\text{d} < t$. The main tree source row counts are shown below. Both heads use the same positive-price rows and column order, with prediction zero on nonpositive-price evaluation queries. At the first two evaluation origins there are 12 and 11 such queries. The main factor models are reused from their completed fits after verifying the positive-price source roster, settings and feature order; replayed inference has maximum departure zero. The direct models are refitted under the identical source mask and common zero rule. The fixed tiny-positive pair follows the same policy. The continuous comparisons also share positive-price eligibility and the zero rule.

The inherited 21-column, larger-capacity and zero-tolerance auxiliary controls retain their original direct-source eligibility: the direct arm includes nonpositive-price rows while the factor excludes

Table 2: Main tree source populations.

Evaluation origin	Direct source rows	Factor source rows
2014-10-02	1,433,608	1,433,608
2015-01-01	1,569,589	1,569,589
2015-04-02	1,707,339	1,707,339
2015-07-02	1,847,849	1,847,849

them. In the October source population this is 112 rows among approximately 1.43 million; the original direct loss contribution of nonpositive-price evaluation queries is 12.569 across 182,220 queries. These auxiliary fits are kept separate from the corrected main and tiny-positive pairs. The original four-origin main comparison and the source-mask correction receipts remain in the evidence snapshot for provenance.

Continuous decoder. Numerical columns are expanded to source-standardized values and missing indicators (130 channels), with two 16-dimensional embeddings for category and brand. The residual MLP has width 256 and three blocks, BatchNorm and dropout 0.1. AdamW uses learning rate 10^{-3}, weight decay 10^{-5} and batches of 8,192. Training runs for at most 30 epochs with patience 8. The first 15 sources fit the model and the 16th selects an epoch by original monetary MAE; the chosen epoch count is then refitted on all 16 sources. Each head selects its own source-validation epoch. Development outcomes do not select epochs. Direct targets use a global source-median price unit; the count head uses price-weighted absolute loss. Both use seed 42.

Uncertainty. For Amazon, each paired comparison uses aligned predictions and labels, draws 57 book-category clusters with replacement, and applies that cluster draw jointly across origins. Each origin's MAE difference is recomputed before equal-origin averaging. H&M uses the analogous 132 product-type clusters. Reported intervals are percentile intervals from 3,000 draws. They quantify the specified clustered resampling distribution conditional on the trained models.

Evaluation status. All reported forecasting origins are already-evaluated TRAIN development periods. Amazon item-LTV official validation/test outcomes and H&M item-sales at 2020-09-07 are not used in this draft. The source package's evidence snapshot records this distinction. Static product price is available under the benchmark schema contract; historical event inputs follow the strict prequery rule.

C NUMERICAL AND CAPACITY CONTROLS

Table 3: Official-roster Amazon67 controls. Entries are direct/factor monetary MAE. Default and tiny-positive rows use identical positive-price sources; the larger and zero-tolerance rows retain the auxiliary eligibility described above. The larger-capacity experiment evaluates October only.

Model/loss setting	October 2014	January 2015
500 trees, depth 6, default	45.758 / 43.377	44.464 / 44.010
1,000 trees, depth 8, default	43.774 / 42.362	—
500 trees, depth 6, $\delta = 10^{-12}$	45.741 / 43.482	44.564 / 44.467
500 trees, depth 6, $\delta = 0$	45.693 / 46.358	44.771 / 43.091
Residual MLP	44.240 / 46.984	42.367 / 40.074

The source-only numerical probe fits 100-tree models on the same 10,000 positive-price April 2014 source rows. There are 16 fits, including global target division by 12.99, default tolerance, scaled tolerance, zero tolerance, and a tiny positive tolerance. Default money-unit conversion changes 85% of sampled tree partitions; scaling the tolerance restores matching sampled partitions and forecast departure 6.38×10^{-7}. Zero and tiny-positive tolerance also restore ordinary global-unit agreement to approximately 7×10^{-7}.

Zero versus tiny-positive tolerance changes native count predictions by 0.110826 count units and 27% of sampled partitions. At the initial count bias of two, 18.2% of labels are exact ties. Tiny positive versus default native count forecasts agree to 3.10×10^{-7}. Count-unit transformations retain additional floating-point/leaf-optimization effects. These probes measure changes in fitted functions, not out-of-time forecast improvements.

D POPULATION, SCALE, AND DYNAMIC-UNIT COMPARISONS

Table 4: History-selected Amazon67 population. The roster uses prior 91-day activity and positive price, with no future-event selection. The wide pair is a separate matched capacity control.

Predictor	2014-10-02	2015-01-01
Direct Cat500, $\delta = 0$	45.045	44.519
Factor Cat500, $\delta = 0$	39.237	38.558
Direct residual MLP	37.279	35.572
Factor residual MLP	37.706	34.493
Wide direct residual MLP	36.837	37.567
Wide factor residual MLP	38.455	36.444
Direct MLP, nearest-count projection	37.176	35.404
Factor MLP, nearest-count projection	37.673	34.402

Table 5: Artificial scale transport on the saved main official-roster Amazon67 fits with identical positive-price sources. Departure is $\mathbb{E}[|f(T_k X) - kf(X)|]/k$ over positive-price queries. Inputs $T_k X$ scale price and deterministic monetary history while leaving counts fixed.

Origin	k	Direct departure	Factor departure
2014-10-02	0.5	8.517	0.965
2014-10-02	2	9.034	1.708
2015-01-01	0.5	9.022	1.535
2015-01-01	2	9.086	2.087

The exploratory highest-1%-price exclusion leaves paired gains 0.463 $[0.238, 0.741]$ in October and -0.213 $[-0.543, 0.227]$ in January. The cut is defined from query-time price, and this is a development subgroup diagnostic. It identifies the contribution of the price tail to monetary MAE without interpreting the price transform as an intervention.

Table 6: Dynamic-unit H&M item-sales, strict prequery inputs, 54 columns. All four learned heads use Cat500/depth6 and the same mature-source policy. B is prior seven-day amount; u is a historical unit estimate. The scaled historical reference is calibrated on the latest mature source.

Head / reference	2020-08-24	2020-08-31
Direct amount	0.057677	0.044856
Unit-normalized ug	0.056698	0.042739
Ordinary residual $B + g$	0.053796	0.038867
Unit-normalized residual $B + ug$	0.053461	0.039180
Activity-bin scaled history	0.054325	0.040452
Activity-bin median-shift history	0.052830	0.042290
Fixed half direct/scaled-history blend	0.055139	0.041449

The unit estimator first uses the item's prior 28-day mean price, then falls back to item 91-day, product 91-day, type 91-day, department 91-day, and global seven-day means. All inputs are strictly historical. The normalized-direct head improves over direct by 0.001548, but the scaled historical reference is better by 0.002330 $[0.000306, 0.004818]$. Adding normalization to the ordinary residual gives 0.000011 $[-0.000186, 0.000291]$. This table isolates the historical unit from the historical amount anchor. Earlier experiments with inclusive timestamp features are not pooled into this comparison.

E COMPILER INTERFACE AND EXECUTABLE CHECKS

The implementation uses SQLGlot 30.19.0 to analyze the restricted task fragment. `task_algebra_compiler.py` accepts the actual SQL and a schema/temporal manifest; `task_algebra_head.py` executes the emitted recipe. The source snapshot is included under `method_source/`.

The analyzer accepts a single column sum or `COALESCE(SUM(column),0)`, one identified outer entity with a declared primary key, a directly foreign-keyed event relation, and a future interval consistent with the manifest horizon. Its static factor is the same entity alias's declared-static operand. It emits an estimated-unit recipe for a varying operand. An explicit experiment supplies any historical unit estimator and anchor. The historical mirror uses a caller-supplied entity roster and $[t - H, t)$; it preserves zero-history entities.

The verification snapshot contains 14 checks: exact static MAE, squared weights for MSE, undeclared keys, undeclared-static attributes, ambiguous operand lineage, past intervals, disjunctive time predicates, horizon mismatch, extra target arithmetic, a missing query join, wrong time aliases, a different entity alias, a real dynamic user aggregate, and executable head/loss/support semantics. All pass their specified expected status. The compiler reads no task target values. On 1,000 sampled entities in each of Amazon and H&M, its historical SQL agrees with independent raw-history amounts to maximum error zero and 4.17×10^{-14}, respectively. Replaying a saved Amazon model through the emitted head gives maximum prediction discrepancy zero.

The source package includes the experimental number snapshot, per-origin CSV, reference audit, and vector figure generator. Figure values are extracted from result JSON fields; schematic curves are labeled as illustrations. The evidence ledger maps manuscript claims to their result artifacts and distinguishes forecasting fits, saved-prediction diagnostics, and query-execution checks.