TABULARMATH: EVALUATING COMPUTATIONAL EXTRAPOLATION IN TABULAR LEARNING VIA PROGRAM-VERIFIED SYNTHESIS

Anonymous authors
Paper under double-blind review

ABSTRACT

Can a tabular predictor learn a calculation from examples and return its answers when outputs exceed the training range? TABULARMATH tests this question using 100 GSM8K word problems and 14 AIME competition problems, compiled into reusable computations with 233,472 program-labeled rows. Nine tabular methods and GPT-OSS-120B predict randomly held-out rows or learn from lower-output rows and predict held-out upper-output rows. We measure both regression fit and answer accuracy after rounding to integers. At a 128-row budget under output shift, TabPFN v2.5 achieves median $R^2 = 0.76$ but only 7.8% mean rounded accuracy on GSM8K. The gap occurs within individual computations: 33 families have $R^2 > 0.8$ and accuracy below 20%. Increasing the budget to 2,048 rows improves random-split prediction but leaves shifted accuracy near 8%. GPT-OSS-120B attains 39.2% shifted accuracy at 128 rows, despite negative median R^2 from large errors elsewhere. These results make precise answers and controlled error a joint target for computational generalization. We release the executable families and evaluation records.

1 INTRODUCTION

Can a model learn a calculation from a table of examples? A table of quantities, prices, and tax rates may record the outputs of a fixed rule. Reproducing that rule requires answers at a specified numerical precision, including when new outputs exceed the training range. Explaining most of the target variance does not guarantee that precision.

Consider a one-unit error. Suppose integer targets have variance σ_y^2, and a predictor returns $\hat{y} = y+1$ for every test row. Its rounded accuracy is zero, while $R^2 = 1 - 1/\sigma_y^2$. As target variance grows, the regression score approaches one without a single correct rounded answer. Do modern tabular predictors exhibit this behavior when learning mathematical computations?

Recent tabular foundation models make this question timely. Prior-fitted networks achieve strong small-data prediction (Hollmann et al., 2022; 2025; Grinsztajn et al., 2025), alongside competitive tree ensembles and neural architectures (Grinsztajn et al., 2022; Gorishniy et al., 2025; McElfresh et al., 2023). Arithmetic networks and symbolic regression explicitly seek numerical rules that extend beyond training examples (Trask et al., 2018; Madsen & Johansen, 2020; Cranmer, 2023). TABULARMATH connects these objectives: deterministic, human-motivated computations test whether modern tabular predictors extend a numerical relationship with answer-level precision.

TABULARMATH starts from mathematical problems in GSM8K (Cobbe et al., 2021) and AIME. Each original problem, or seed, becomes a *task family*: a computation with variable inputs and a program that supplies its answers. Inputs become columns, and execution labels 114 families with 2,048 rows each. A sorted-target split trains on the lower 80% of outputs and tests on the upper 20%. A random split provides the comparison in which training and test examples come from the same sampled distribution, at the same row budget.

TabPFN v2.5 fits the numerical relationship well but rarely returns the correct rounded answer under output-range shift. GPT-OSS-120B answers more queries correctly on the same small-table tasks,

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.

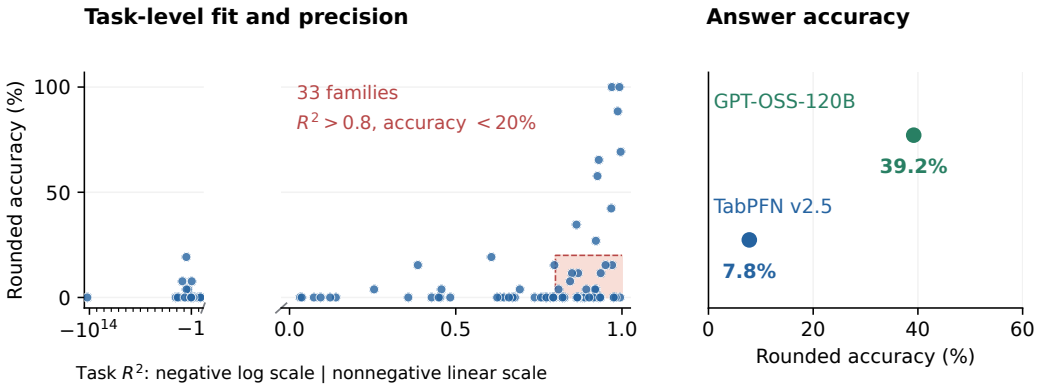


Figure 1: **Fit and answer precision separate within the same task.** Left: all 100 GSM8K families for TabPFN v2.5 at a 128-row cap under output shift; each point is one task. The broken horizontal axis uses a logarithmic negative segment and a linear nonnegative segment. The highlighted region contains 33 families with $R^2 > 0.8$ and accuracy below 20%. Right: mean rounded accuracy for GPT-OSS-120B and TabPFN v2.5 in this evaluation setting.

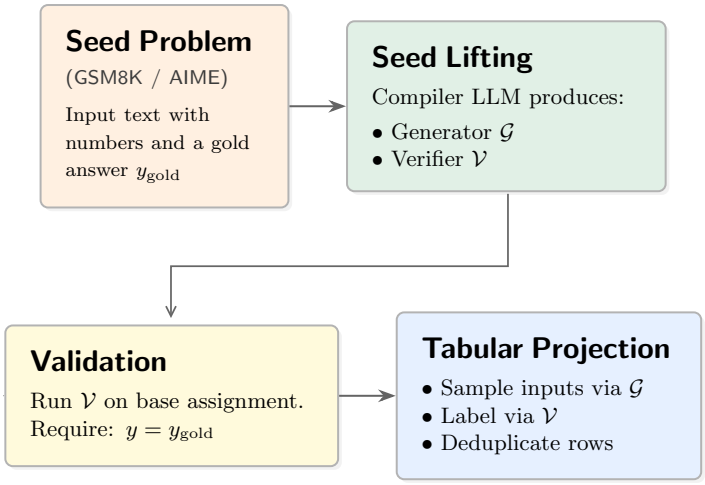


Figure 2: **From mathematical seeds to tabular prediction.** Compilation produces a generator and verifier, validation anchors the verifier to the seed answer, and tabular projection yields labeled rows. The two evaluation splits vary the prediction regime while retaining this executable task definition.

yet its negative median R^2 exposes large errors elsewhere. More training rows substantially improve random-split prediction without closing the shifted precision gap.

TABULARMATH contributes executable families from everyday and competition mathematics, controlled row-budget and split comparisons, and joint task-level analysis of fit and precision. These resources support a specific generalization problem: *preserve a computation's answers as the queried output range expands.*

2 EXECUTABLE TABULAR TASK FAMILIES

From a mathematical problem to a table. A seed supplies a natural-language problem and its answer. Following VeRA's executable task family approach (Cheng et al., 2026), a language-model teacher identifies variable quantities and writes a question template. It also writes a generator G

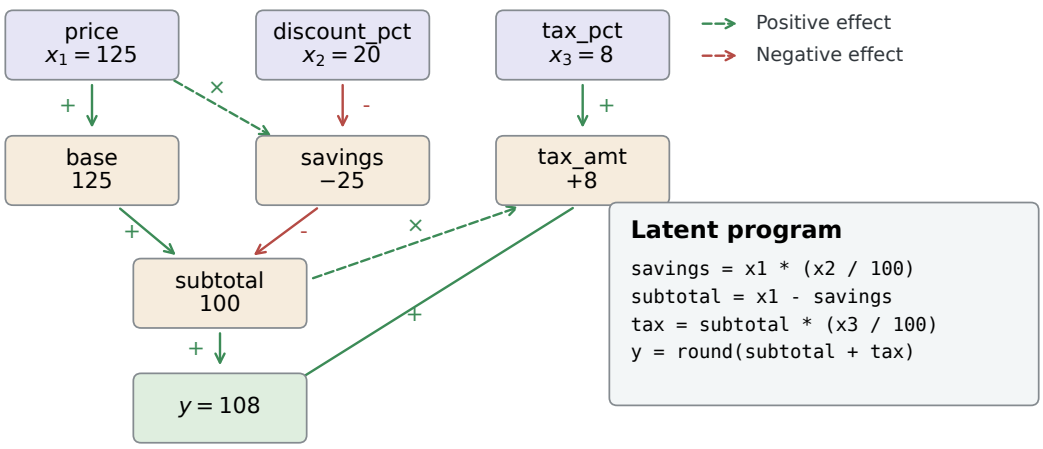


Figure 3: **The target is an executed computation.** The pricing example expresses the same calculation through intermediate quantities and a short program. The diagram marks the discount's signed contribution as -25; the program stores the positive savings amount and subtracts it. Both yield a discounted subtotal of 100 and a taxed total of 108. Learners observe input–output rows, while the reference program determines the labels.

that samples valid inputs x and an answer program, called the verifier V, that computes targets y (Figure 2). Each family produces the dataset

$$\mathcal{D}_i = \{(x_{ij}, V_i(x_{ij}))\}_{j=1}^{2048}, \qquad x_{ij} \sim G_i. \tag{1}$$

The learner receives the table rather than the verifier. Each family is a separate regression task, so learning concerns the relationship within a family rather than predicting across unrelated mathematical problems.

For example, a billing rule maps a base price p, discount d, and tax t to the target $p(1-d)(1+t)$. Each row records the rule's inputs and computed bill. Larger prices or rates can produce totals beyond the training range, while the calculation stays fixed. This illustrative family isolates the problem of learning a deterministic calculation from its input–output rows. The benchmark also includes multi-step and competition-mathematics computations whose dependencies are less direct.

What the learner observes. Figure 3 expresses the pricing calculation as a graph and as code. Other families use conditionals, iteration, integer constraints, or search, but all expose the same tabular interface. Predictors receive examples, not intermediate execution steps or the answer program. They may interpolate, infer a formula, or use another representation; evaluation concerns only their outputs. Conditional label variance is zero relative to the reference program, so prediction errors can be traced to concrete inputs rather than uncertain observational labels.

Validation. The verifier must reproduce the seed answer, respect type and range constraints, and complete within the execution budget. Schema, generation, and execution failures return corrective feedback, with at most 20 synthesis attempts. Accepted programs produce deduplicated rows. Each released specification contains the seed assignment, input constraints, and answer program. These checks anchor executable labels to the seed; the program then defines the reference computation on sampled inputs.

Representation. Numeric quantities form the primary columns. Generic arithmetic transforms add log-magnitude, sign, parity, modular features, and categorical encodings where applicable. Every method receives the same task columns, with their feature counts reported in Table 1. These

Table 1: **Benchmark construction.** Each seed defines a separate task with 2,048 rows. Feature counts include the constructed representations supplied to the predictors.

Source	Tasks	Rows	Mean features	Feature range
GSM8K	100	204,800	45.3	25–98
AIME 2024	14	28,672	40.6	18–88

Table 2: **A seed becomes a family of labeled rows.** Worked assignments from the flower-counting construction example, before generic feature transforms. The final row is the original seed. Values follow the same generator constraints and answer program.

Yellow a	Extra purple (%)	Green (%)	Purple p	Green g	Target y
20	50	40	30	20	70
15	100	20	30	9	54
40	50	12	60	12	112
8	25	50	10	9	27
10	80	25	18	7	35

transforms expose numerical structure without supplying the computation that combines the quantities into an answer.

2.1 A Worked Generator–Verifier Family

Consider the flower-counting example used in the construction specification. A garden has a yellow flowers, b percent more purple flowers than yellow flowers, and green flowers numbering c percent of the combined yellow and purple count. The seed uses $(a, b, c) = (10, 80, 25)$ and has answer 35. Parameterization turns the three quantities into slots while retaining the relations between them:

$$p = a(1 + b/100), \qquad g = (a + p)c/100, \qquad y = a + p + g. \tag{2}$$

The generator proposes $a \in \{1, \ldots, 100\}$ and $b, c \in \{0, \ldots, 100\}$, accepting assignments for which both p and g are integers. These domain constraints are part of the task. They preserve the meaning of counting flowers rather than producing fractional objects and rounding away the inconsistency. The verifier evaluates the accepted assignment, and the resulting row contains the inputs and its computed target.

The rows in Table 2 change the counts while preserving the same three relations. A random split samples training and test rows throughout this distribution. Sorting by the answer instead trains on smaller totals and tests larger ones. Different combinations of inputs can produce the same total, so a larger answer need not require every input to be larger.

For example, the row with target 112 has more yellow flowers than the row with target 70, but its green percentage is smaller, at 12% rather than 40%. The larger answer comes from the combined calculation, not from increasing every coordinate. A predictor must therefore account for how the quantities interact. The upper-output-tail split orders rows by the result of this calculation. Individual input coordinates can overlap between training and test rows; held-out rows occupy the upper end of the output distribution. The executable family keeps the relationship fixed across the split.

Not every family has integer-valued answers. A distance–rate–time example samples a base speed v, headwind reduction h, and travel time t, then constructs distance $d = (v - h)t$. The input row (d, h, t) has target $v = d/t + h$, which can be nonintegral. Constructing the distance from the other quantities keeps the assignment coherent. Appendix E gives its sampling ranges. This contrasts with the flower family's integrality constraints and motivates defining the scoring resolution explicitly rather than assuming every program returns a count.

3 EVALUATION DESIGN

Row budgets and splits. We subsample to a row cap $r \in \{32, 64, 128, 256, 512, 1024, 2048\}$ before an 80/20 split. The random condition samples both sets from the table; the output-shift condition sorts by y and holds out its upper 20%. At cap 128, approximately 102 rows provide training examples and 26 provide queries. Selection on $y = V(x)$ also changes the input distribution through the fixed program's dependencies; individual coordinates may overlap, and ties may straddle the boundary. The measured regime is upper-output-tail prediction under this joint shift.

Predictors. The nine tabular methods are XGBoost (Chen & Guestrin, 2016), CatBoost (Prokhorenkova et al., 2018), LightGBM (Ke et al., 2017), Random Forest (Breiman, 2001), TabPFN v2 and v2.5 (Hollmann et al., 2025; Grinsztajn et al., 2025), RealMLP-TD-S (Holzmüller et al., 2024), TabM (Gorishniy et al., 2025), and xRFM (Beaglehole et al., 2025). They use fixed configurations; TabPFN v2.5 uses `v2.5_default` with eight estimators. Input and target standardization use training-split statistics.

GPT-OSS-120B (OpenAI et al., 2025) receives key–value serializations of context rows followed by query rows and a request for numeric predictions. It uses temperature zero and row caps 32, 64, and 128. The harness permits up to ten attempts to obtain a parseable response and extracts the required numeric values. This serialization exposes the same task columns through a language-model interface. Appendix A gives the evaluation budgets and parsing protocol.

Two complementary scores. For task i, rounded accuracy is

$$A_i = \frac{100}{|Q_i|} \sum_{j \in Q_i} \mathbf{1}\{\mathrm{round}(\hat{y}_{ij}) = \mathrm{round}(y_{ij})\}. \tag{3}$$

Rounded accuracy uses a common unit-width answer resolution across families, including those with nonintegral program outputs. We report the equal-task mean of A_i and the median task-level R^2, a robust measure of central regression performance. The joint task-level analysis below retains individual fits and answer accuracies.

We examine the scores together for each family before drawing conclusions from benchmark averages. Accuracy records how often a prediction falls in the target's rounding bin. Its R^2 records squared error relative to the variation among that family's query targets. A family can have high R^2 while many predictions miss the required bin, or frequent correct answers alongside a few large errors. The joint analysis identifies these behaviors on the same computation. The benchmark summaries retain those complementary views across the task bank, with each family receiving equal weight.

Answer scale and aggregation. Standardization and answer scoring serve different purposes. Standardization makes the numerical inputs and targets suitable for regression; rounded accuracy evaluates predictions on the original answer scale. The target of a flower-counting task remains a count, while the distance–rate–time task is scored at the same unit-width resolution. Keeping that resolution fixed makes the accuracy criterion explicit across the task bank. RMSE and MAE provide additional information about the magnitude of errors, with their scale specified in Appendix F.

The evaluation covers 100 GSM8K and 14 AIME task families. Each task contributes equally to its source-level accuracy, regardless of its number of columns or output scale. The regression summaries take the median across those task-level scores. We present the full row-budget profiles for both sources and retain individual task results for diagnosing the relationship between fit and precision. A benchmark mean describes answer frequency across computations; the joint plot asks whether good fit and poor precision occur on the same computation.

Comparisons at a fixed budget. Cross-model comparisons hold the row cap and split regime fixed. The 128-row comparison provides a common operating point for all ten methods. Scaling profiles then show what changes as more rows become available, with the language-model comparison restricted to its evaluated budgets. The larger tabular budgets measure additional-data performance rather than equal-budget performance against GPT-OSS-120B. Reading these comparisons together separates sample efficiency from the gains obtained by supplying more examples.

Table 3: **GSM8K at a 128-row cap.** Nine tabular methods and GPT-OSS-120B on the GSM8K benchmark. Rounded accuracy is an equal-task mean (%); R^2 is a task-level median. The table presents the complete method comparison under both split regimes.

Model	Random acc.	Random R^2	Shifted acc.	Shifted R^2
GPT-OSS-120B	49.6	0.84	39.2	-1.30
TabPFN v2.5	39.5	1.00	7.8	0.76
TabPFN v2	25.3	0.99	3.8	-1.30
TabM	9.8	0.96	0.2	-1.50
CatBoost	4.9	0.81	0.0	-5.20
XGBoost	5.2	0.82	0.0	-4.00
LightGBM	5.2	0.81	0.2	-3.40
Random Forest	4.6	0.79	0.0	-4.30
RealMLP-TD-S	4.0	0.77	0.3	-4.50
xRFM	1.7	0.90	0.5	-2.70

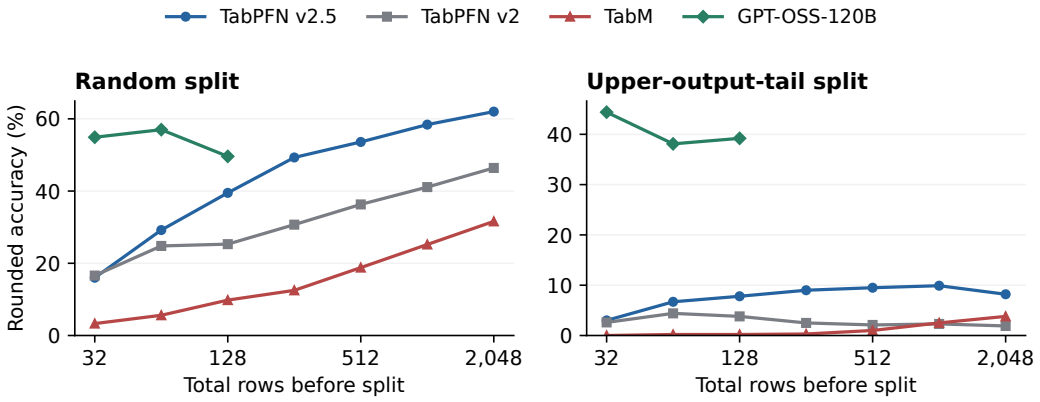


Figure 4: **Data scaling changes the two split regimes differently.** Rounded accuracy on the GSM8K benchmark across evaluated row budgets. GPT-OSS-120B is evaluated at caps 32, 64, and 128; tabular methods extend to 2,048 rows. Lines connect the reported budgets.

4 Results

4.1 Strong Regression Fit Conceals Low Answer Precision

At a 128-row cap under output-range shift, TabPFN v2.5 obtains median $R^2 = 0.76$ and mean rounded accuracy 7.8% on GSM8K. Figure 1 links both scores within each task: 42 families achieve $R^2 > 0.8$, and 33 of those have accuracy below 20%. GPT-OSS-120B reaches 39.2% rounded accuracy at the same row budget, approximately five times the TabPFN v2.5 rate.

GPT-OSS-120B's higher rounded accuracy comes with negative median shifted R^2: more queries receive correct answers, but large errors elsewhere damage regression fit.

The comparison identifies two concrete targets: tabular methods need to return more exact answers, and the language model needs to control its largest errors. Figure 1 makes both visible: the task-level plot locates computations with good fit but poor precision, and the accuracy comparison measures answer frequency. Table 3 places these outcomes within the full method panel. The following analyses examine how much of the precision gap is reduced by additional rows and how it changes on competition-level mathematics.

Table 4: **Scaling across the complete method panel.** GSM8K rounded accuracy at three evaluated row budgets. Tasks receive equal weight. A dash denotes a row budget outside the language-model evaluation. Additional rows primarily improve random-split performance.

Model	Rand. 32	128	2,048	Shift 32	128	2,048
GPT-OSS-120B	54.9	49.6	–	44.4	39.2	–
TabPFN v2.5	16.0	39.5	62.0	3.0	7.8	8.2
TabPFN v2	16.6	25.3	46.4	2.6	3.8	1.9
TabM	3.3	9.8	31.6	0.0	0.2	3.8
CatBoost	2.0	4.9	29.6	0.0	0.0	0.5
XGBoost	1.6	5.2	24.0	0.1	0.0	0.4
LightGBM	2.3	5.2	23.6	0.0	0.2	0.3
Random Forest	2.3	4.6	17.9	0.0	0.0	0.1
RealMLP-TD-S	2.0	4.0	19.3	0.1	0.3	1.0
xRFM	1.6	1.7	1.7	0.1	0.5	2.0

4.2 More Rows Help Interpolation More Than Output-Range Shift

TabPFN v2.5's random-split rounded accuracy rises from 16.0% at 32 rows to 62.0% at 2,048 rows. Its shifted accuracy rises from 3.0% to a peak of 9.9% at 1,024 rows, then reaches 8.2% at 2,048 rows. More within-family examples substantially improve interpolation without producing a comparable rise in upper-tail accuracy. Figure 4 shows the full observed budget range rather than only the endpoints.

TabM illustrates why regression fit alone remains incomplete even when it improves under shift. Its median shifted R^2 becomes positive at 1,024 rows and reaches 0.17 at 2,048 rows, while rounded accuracy at the latter budget is 3.8%. The precision gap consequently extends beyond a single prior-fitted model. Tree-ensemble and other neural baselines also achieve much higher random-split accuracy than shifted accuracy as the row budget grows.

Table 4 gives the corresponding results for all ten methods. GPT-OSS-120B is evaluated at caps 32, 64, and 128, allowing direct comparison at those budgets. The larger caps show how the tabular methods scale with additional data.

The split determines what additional examples can show the learner. Random-split training rows and queries span the same sampled input–output distribution, so a larger budget supplies more examples throughout that range. Under output shift, the extra rows still train the predictor below the held-out output region. The comparison therefore tests whether denser examples of a computation also improve predictions beyond the observed answers. The observed scaling favors the first setting: interpolation improves substantially, while the upper-tail precision challenge remains. The reference program keeps the target computation fixed as both the budget and split vary.

4.3 Output Shift Reduces Accuracy Across the Method Panel

How much accuracy does moving to the upper output tail cost? Table 5 compares each method's random-split and shifted means at cap 128. Both settings use the same computational task bank and row budget, changing which part of each sampled table supplies the queries.

Every method loses accuracy. TabPFN v2.5 falls from 39.5% to 7.8%, a 31.7-point decline, while GPT-OSS-120B falls from 49.6% to 39.2%, a 10.4-point decline. The smaller losses of several conventional baselines accompany low random-split accuracy: a system near the floor has less accuracy to lose. Both the initial level and the change matter when assessing how well a predictor preserves answer precision under output-range shift.

4.4 Competition Mathematics Remains Difficult

At cap 32, GPT-OSS-120B reaches random and shifted accuracies of 25.5% and 14.3% on AIME. The tabular methods maintain particularly low shifted accuracy across budgets (Figure 5); additional rows mainly improve prediction within the sampled distribution. TabPFN v2.5 reaches 17.3% random-split accuracy at 2,048 rows, while its shifted accuracy is 0.1%. Thus the competition prob-

Table 5: **Precision losses under upper-output-tail evaluation.** GSM8K rounded accuracy at cap 128. Change is the shifted mean minus the random-split mean, in percentage points, calculated from the displayed scores.

Model	Random (%)	Upper tail (%)	Change (pp)
GPT-OSS-120B	49.6	39.2	-10.4
TabPFN v2.5	39.5	7.8	-31.7
TabPFN v2	25.3	3.8	-21.5
TabM	9.8	0.2	-9.6
CatBoost	4.9	0.0	-4.9
XGBoost	5.2	0.0	-5.2
LightGBM	5.2	0.2	-5.0
Random Forest	4.6	0.0	-4.6
RealMLP-TD-S	4.0	0.3	-3.7
xRFM	1.7	0.5	-1.2

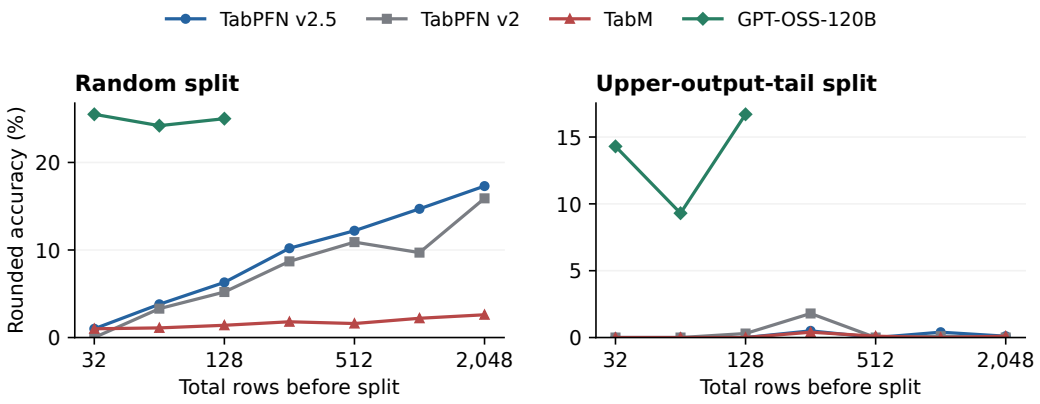


Figure 5: **AIME-derived families expose a harder precision regime.** Rounded accuracy on the 14-task competition-mathematics benchmark. Each task receives equal weight. The row-budget limits are the same as in the GSM8K evaluation.

lems preserve the distinction between learning a numerical relationship within the sampled range and returning precise answers beyond that range.

4.5 Few-Example Accuracy and Data Scaling Favor Different Methods

At a 32-row cap on GSM8K, GPT-OSS-120B reaches 54.9% random-split accuracy, compared with 16.0% for TabPFN v2.5. The latter reaches 53.6% at 512 rows and 62.0% at 2,048 rows. The language model supplies strong low-budget answer accuracy, while TabPFN v2.5 benefits substantially from more in-distribution examples. Under output shift, however, its 2,048-row accuracy is 8.2%, compared with GPT-OSS-120B's 44.4% at 32 rows. Additional data narrows the random-split gap without resolving the extrapolation problem.

5 What the Benchmark Reveals

Why the metrics disagree. For an integer target y, a prediction strictly within half a unit has the same rounded value. The squared-error contribution at that boundary is only 0.25, whereas the denominator of R^2 follows the spread of the task's targets. Tasks with large target variation can therefore assign a small relative penalty to errors that cross several integer bins. For nonintegral targets, the relevant boundary is the edge of the target's rounding bin, which can be closer than half a unit. Reporting rounded accuracy fixes the answer resolution explicitly; reporting R^2 retains information about the size of errors beyond that boundary.

Rounded accuracy assigns the same penalty to a small miss and an extreme error. Retaining R^2 prevents improvements in answer frequency from hiding such increases in error magnitude.

A predictor can answer most rows exactly yet make a few very large mistakes. Rounded accuracy counts each mistake once, whereas squared error grows with its magnitude. Another predictor can miss more often but stay close to the targets. Joint task-level analysis distinguishes these behaviors and makes frequent correct answers with controlled error a concrete modeling target.

Learning rules that extend to larger outputs. The smaller gains in upper-tail precision motivate methods that exploit numerical structure, including program induction, arithmetic modules, and hybrid predictors. TABULARMATH tests their predictions through answer accuracy, error magnitude, and improvement as the row budget grows; recovering the underlying rule is a modeling objective, not inferred from fit alone.

A rule-learning method can infer a candidate relationship from training rows and apply it to held-out inputs. The hidden reference program supplies the targets. Testing both splits measures whether the relationship improves interpolation, upper-tail prediction, or both. The candidate must handle valid input combinations, including nonintegral answers, at the specified scoring resolution.

An executable family also supports diagnosis of individual errors. In the flower example, the inputs determine the purple count, the green count, and the total. Replaying these steps locates the target for a failed row; testing nearby valid assignments reveals whether a candidate expression respects the dependencies. A repaired predictor can then be evaluated on held-out rows. The same procedure supports targeted tests of branching rules, divisibility conditions, and dependencies between quantities, using fresh valid inputs from the family's generator.

6 RELATED WORK

Tabular prediction and learned priors. Comparative studies establish the strengths of tree ensembles and neural predictors (Grinsztajn et al., 2022; McElfresh et al., 2023; Borisov et al., 2024). Feature representations, attention, and parameter-efficient ensembles improve tabular learning (Gorishniy et al., 2021; 2022; Arik & Pfister, 2021; Huang et al., 2020; Gorishniy et al., 2025). Prior-fitted networks instead learn prediction from synthetic task distributions (Müller et al., 2022; Hollmann et al., 2022; 2025; Grinsztajn et al., 2025). TABULARMATH gives these approaches a common deterministic target and tests precision as the queried output range expands.

Arithmetic extrapolation and executable reasoning. Systematic generalization evaluates extension beyond observed combinations (Lake & Baroni, 2018; Hupkes et al., 2019). Arithmetic studies link extrapolation to representations and training data (Jelassi et al., 2023; McLeish et al., 2024); NALU and neural arithmetic units encode numerical structure (Trask et al., 2018; Madsen & Johansen, 2020). PySR, SRBench, AI Feynman, and neural symbolic regression study equation recovery (Cranmer, 2023; La Cava et al., 2021; Udrescu & Tegmark, 2020; Biggio et al., 2021). Mathematical benchmarks and program-aided reasoning supply complementary interfaces (Hendrycks et al., 2021; Gao et al., 2023; Chen et al., 2023). VeRA supplies executable task families (Cheng et al., 2026); GSM-Ranges probes numerical-scale robustness (Shrestha et al., 2025). TABULARMATH projects computations into tables: fit and rounded accuracy measure whether predictions track and correctly evaluate the relationship. Appendix G connects this interface to in-context learning and symbolic induction.

7 CONCLUSION

TABULARMATH makes answer precision testable in tabular prediction. Strong regression fit often accompanies low rounded accuracy, while additional examples mainly improve random-split prediction. The released programs and records support the next modeling target: precise upper-tail answers with controlled error across the query set.

AI USAGE STATEMENT

Language models proposed benchmark generators and verifiers, as described in Section 2. **With human in the loop for supervision and quality control**, coding agents assisted with writing, better result presentation, and experiment execution. **All research ideas, structure of the manuscript, and result analysis strategies originated with the human authors**, who have reviewed and revised the codebase and the manuscript, and human authors take full responsibility for all the scientific claims and artifacts.

REPRODUCIBILITY STATEMENT

Benchmark code and data are available at `https://anonymous.4open.science/r/TabularMath-ICLR2027-BE93`. The repository contains the GSM8K and AIME tables, their curation inputs, and evaluation drivers for tabular predictors and language models. Its README documents dependencies, split and row-budget settings, and provider credentials for API-backed runs. Sections 2 and 3 define construction and metrics; Appendices A, E, and F detail the pipeline and comparison sets. The manuscript source archive supplies task-level metrics, the numerical-resolution audit, and scripts that reproduce the tables and figures.

REFERENCES

Ekin Akyürek, Dale Schuurmans, Jacob Andreas, Tengyu Ma, and Denny Zhou. What learning algorithm is in-context learning? Investigations with linear models. In *International Conference on Learning Representations*, 2023. URL https://arxiv.org/abs/2211.15661.

Aida Amini, Saadia Gabriel, Peter Lin, Rik Koncel-Kedziorski, Yejin Choi, and Hannaneh Hajishirzi. MathQA: Towards Interpretable Math Word Problem Solving with Operation-Based Formalisms. *arXiv preprint arXiv:1905.13319*, 2019. URL https://arxiv.org/abs/1905.13319.

Sercan O. Arik and Tomas Pfister. TabNet: Attentive Interpretable Tabular Learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2021. URL https://arxiv.org/abs/1908.07442.

Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, et al. Program Synthesis with Large Language Models. *arXiv preprint arXiv:2108.07732*, 2021. URL https://arxiv.org/abs/2108.07732.

Zhangir Azerbayev, Hailey Schoelkopf, Keiran Paster, Marco Dos Santos, Stephen McAleer, Albert Q. Jiang, et al. Llemma: An Open Language Model For Mathematics. *arXiv preprint arXiv:2310.10631*, 2023. URL https://arxiv.org/abs/2310.10631.

Daniel Beaglehole, David Holzmüller, Adityanarayanan Radhakrishnan, and Mikhail Belkin. xRFM: Accurate, scalable, and interpretable feature learning models for tabular data. *arXiv preprint arXiv:2508.10053*, 2025. URL https://arxiv.org/abs/2508.10053.

Luca Biggio, Tommaso Bendinelli, Alexander Neitz, Aurelien Lucchi, and Giambattista Parascandolo. Neural Symbolic Regression that Scales. In *International Conference on Machine Learning*, 2021. URL https://arxiv.org/abs/2106.06427.

Vadim Borisov, Tobias Leemann, Kathrin Seßler, Johannes Haug, Martin Pawelczyk, and Gjergji Kasneci. Deep Neural Networks and Tabular Data: A Survey. *IEEE Transactions on Neural Networks and Learning Systems*, 35(6):7499–7519, 2024. doi: 10.1109/TNNLS.2022.3229161. URL https://arxiv.org/abs/2110.01889.

Leo Breiman. Random Forests. *Machine Learning*, 45(1):5–32, 2001. doi: 10.1023/A:1010933404324. URL https://link.springer.com/article/10.1023/A:1010933404324.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, et al. Evaluating Large Language Models Trained on Code. *arXiv preprint arXiv:2107.03374*, 2021. URL https://arxiv.org/abs/2107.03374.

Tianqi Chen and Carlos Guestrin. XGBoost: A Scalable Tree Boosting System. In *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 785–794, 2016. doi: 10.1145/2939672.2939785. URL https://arxiv.org/abs/1603.02754.

Wenhu Chen, Xueguang Ma, Xinyi Wang, and William W. Cohen. Program of Thoughts Prompting: Disentangling Computation from Reasoning for Numerical Reasoning Tasks. *Transactions on Machine Learning Research*, 2023. URL https://arxiv.org/abs/2211.12588.

Zerui Cheng, Jiashuo Liu, Chunjie Wu, Jianzhu Yao, Pramod Viswanath, Ge Zhang, et al. VeRA: Verified Reasoning Data Augmentation at Scale. *arXiv preprint arXiv:2602.13217*, 2026. URL https://arxiv.org/abs/2602.13217.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, et al. Training Verifiers to Solve Math Word Problems. *arXiv preprint arXiv:2110.14168*, 2021. URL https://arxiv.org/abs/2110.14168.

Miles Cranmer. Interpretable Machine Learning for Science with PySR and SymbolicRegression.jl. *arXiv preprint arXiv:2305.01582*, 2023. URL https://arxiv.org/abs/2305.01582.

Miles Cranmer, Alvaro Sanchez-Gonzalez, Peter Battaglia, Rui Xu, Kyle Cranmer, David Spergel, et al. Discovering Symbolic Models from Deep Learning with Inductive Biases. In *Advances in Neural Information Processing Systems*, 2020. URL https://arxiv.org/abs/2006.11287.

Dheeru Dua, Yizhong Wang, Pradeep Dasigi, Gabriel Stanovsky, Sameer Singh, and Matt Gardner. DROP: A Reading Comprehension Benchmark Requiring Discrete Reasoning Over Paragraphs. *arXiv preprint arXiv:1903.00161*, 2019. URL https://arxiv.org/abs/1903.00161.

Nouha Dziri, Ximing Lu, Melanie Sclar, Xiang Lorraine Li, Liwei Jiang, Bill Yuchen Lin, et al. Faith and Fate: Limits of Transformers on Compositionality. *arXiv preprint arXiv:2305.18654*, 2023. URL https://arxiv.org/abs/2305.18654.

Luyu Gao, Aman Madaan, Shuyan Zhou, Uri Alon, Pengfei Liu, Yiming Yang, et al. PAL: Program-aided Language Models. In *Proceedings of the International Conference on Machine Learning*, 2023. URL https://arxiv.org/abs/2211.10435.

Shivam Garg, Dimitris Tsipras, Percy Liang, and Gregory Valiant. What Can Transformers Learn In-Context? A Case Study of Simple Function Classes. In *Advances in Neural Information Processing Systems*, 2022. URL https://arxiv.org/abs/2208.01066.

Yury Gorishniy, Ivan Rubachev, Valentin Khrulkov, and Artem Babenko. Revisiting Deep Learning Models for Tabular Data. In *Advances in Neural Information Processing Systems*, 2021. URL https://arxiv.org/abs/2106.11959.

Yury Gorishniy, Ivan Rubachev, and Artem Babenko. On Embeddings for Numerical Features in Tabular Deep Learning. In *Advances in Neural Information Processing Systems*, 2022. URL https://arxiv.org/abs/2203.05556.

Yury Gorishniy, Akim Kotelnikov, and Artem Babenko. TabM: Advancing Tabular Deep Learning with Parameter-Efficient Ensembling. In *International Conference on Learning Representations*, 2025. URL https://arxiv.org/abs/2410.24210.

Léo Grinsztajn, Edouard Oyallon, and Gaël Varoquaux. Why do tree-based models still outperform deep learning on tabular data? *arXiv preprint arXiv:2207.08815*, 2022. URL https://arxiv.org/abs/2207.08815.

Léo Grinsztajn, Klemens Flöge, Oscar Key, Felix Birkel, Philipp Jund, Brendan Roof, et al. TabPFN-2.5: Advancing the State of the Art in Tabular Foundation Models. *arXiv preprint arXiv:2511.08667*, 2025. URL https://arxiv.org/abs/2511.08667.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, et al. Measuring Mathematical Problem Solving With the MATH Dataset. In *NeurIPS Datasets and Benchmarks*, 2021. URL https://arxiv.org/abs/2103.03874.

Noah Hollmann, Samuel Müller, Katharina Eggensperger, and Frank Hutter. TabPFN: A Transformer That Solves Small Tabular Classification Problems in a Second. *arXiv preprint arXiv:2207.01848*, 2022. URL https://arxiv.org/abs/2207.01848.

Noah Hollmann, Samuel Müller, Lennart Purucker, Arjun Krishnakumar, Max Körfer, Shi Bin Hoo, et al. Accurate predictions on small data with a tabular foundation model. *Nature*, 637(8045): 319–326, 2025. doi: 10.1038/s41586-024-08328-6. URL https://www.nature.com/articles/s41586-024-08328-6.

David Holzmüller, Léo Grinsztajn, and Ingo Steinwart. Better by Default: Strong Pre-Tuned MLPs and Boosted Trees on Tabular Data. In *Advances in Neural Information Processing Systems*, 2024. URL https://arxiv.org/abs/2407.04491.

Xin Huang, Ashish Khetan, Milan Cvitkovic, and Zohar Karnin. TabTransformer: Tabular Data Modeling Using Contextual Embeddings. *arXiv preprint arXiv:2012.06678*, 2020. URL https://arxiv.org/abs/2012.06678.

Dieuwke Hupkes, Verna Dankers, Mathijs Mul, and Elia Bruni. Compositionality decomposed: how do neural networks generalise? *arXiv preprint arXiv:1908.08351*, 2019. URL https://arxiv.org/abs/1908.08351.

Samy Jelassi, Stéphane d'Ascoli, Carles Domingo-Enrich, Yuhuai Wu, Yuanzhi Li, and François Charton. Length Generalization in Arithmetic Transformers. *arXiv preprint arXiv:2306.15400*, 2023. URL https://arxiv.org/abs/2306.15400.

Pierre-Alexandre Kamienny, Stéphane d'Ascoli, Guillaume Lample, and François Charton. End-to-end symbolic regression with transformers. *arXiv preprint arXiv:2204.10532*, 2022. URL https://arxiv.org/abs/2204.10532.

Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, et al. LightGBM: A Highly Efficient Gradient Boosting Decision Tree. *Advances in Neural Information Processing Systems*, 30, 2017. URL https://proceedings.neurips.cc/paper/2017/hash/6449f44a102fde848669bdd9eb6b76fa-Abstract.html.

Daniel Keysers, Nathanael Schärli, Nathan Scales, Hylke Buisman, Daniel Furrer, Sergii Kashubin, et al. Measuring Compositional Generalization: A Comprehensive Method on Realistic Data. In *International Conference on Learning Representations*, 2020. URL https://arxiv.org/abs/1912.09713.

Najoung Kim and Tal Linzen. COGS: A Compositional Generalization Challenge Based on Semantic Interpretation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, 2020. URL https://arxiv.org/abs/2010.05465.

Jannik Kossen, Neil Band, Clare Lyle, Aidan N. Gomez, Tom Rainforth, and Yarin Gal. Self-Attention Between Datapoints: Going Beyond Individual Input-Output Pairs in Deep Learning. In *Advances in Neural Information Processing Systems*, 2021. URL https://arxiv.org/abs/2106.02584.

William La Cava, Patryk Orzechowski, Bogdan Burlacu, Fabrício Olivetti de França, Marco Virgolin, Ying Jin, et al. Contemporary Symbolic Regression Methods and their Relative Performance. In *NeurIPS Datasets and Benchmarks*, 2021. URL https://arxiv.org/abs/2107.14351.

Brenden M. Lake and Marco Baroni. Generalization without systematicity: On the compositional skills of sequence-to-sequence recurrent networks. In *International Conference on Machine Learning*, 2018. URL https://arxiv.org/abs/1711.00350.

Nayoung Lee, Kartik Sreenivasan, Jason D. Lee, Kangwook Lee, and Dimitris Papailiopoulos. Teaching Arithmetic to Small Transformers. *arXiv preprint arXiv:2307.03381*, 2023. URL https://arxiv.org/abs/2307.03381.

Aitor Lewkowycz, Anders Andreassen, David Dohan, Ethan Dyer, Henryk Michalewski, Vinay Ramasesh, et al. Solving Quantitative Reasoning Problems with Language Models. *arXiv preprint arXiv:2206.14858*, 2022. URL https://arxiv.org/abs/2206.14858.

Andreas Madsen and Alexander Rosenberg Johansen. Neural Arithmetic Units. In *International Conference on Learning Representations*, 2020. URL https://arxiv.org/abs/2001.05016.

Duncan McElfresh, Sujay Khandagale, Jonathan Valverde, Vishak Prasad C, Benjamin Feuer, Chinmay Hegde, et al. When Do Neural Nets Outperform Boosted Trees on Tabular Data? In *Advances in Neural Information Processing Systems*, 2023. URL https://arxiv.org/abs/2305.02997.

Sean McLeish, Arpit Bansal, Alex Stein, Neel Jain, John Kirchenbauer, Brian R. Bartoldson, et al. Transformers Can Do Arithmetic with the Right Embeddings. *arXiv preprint arXiv:2405.17399*, 2024. URL https://arxiv.org/abs/2405.17399.

Samuel Müller, Noah Hollmann, Sebastian Pineda Arango, Josif Grabocka, and Frank Hutter. Transformers Can Do Bayesian Inference. In *International Conference on Learning Representations*, 2022. URL https://arxiv.org/abs/2112.10510.

Rodrigo Nogueira, Zhiying Jiang, and Jimmy Lin. Investigating the Limitations of Transformers with Simple Arithmetic Tasks. *arXiv preprint arXiv:2102.13019*, 2021. URL https://arxiv.org/abs/2102.13019.

OpenAI, Sandhini Agarwal, Lama Ahmad, Jason Ai, Sam Altman, Andy Applebaum, et al. gpt-oss-120b & gpt-oss-20b Model Card. *arXiv preprint arXiv:2508.10925*, 2025. URL https://arxiv.org/abs/2508.10925.

Arkil Patel, Satwik Bhattamishra, and Navin Goyal. Are NLP Models really able to Solve Simple Math Word Problems? In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics*, 2021. URL https://arxiv.org/abs/2103.07191.

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, et al. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, 12(85):2825–2830, 2011. URL https://jmlr.org/papers/v12/pedregosa11a.html.

Brenden K. Petersen, Mikel Landajuela, T. Nathan Mundhenk, Claudio P. Santiago, Soo K. Kim, and Joanne T. Kim. Deep symbolic regression: Recovering mathematical expressions from data via risk-seeking policy gradients. In *International Conference on Learning Representations*, 2021. URL https://arxiv.org/abs/1912.04871.

Sergei Popov, Stanislav Morozov, and Artem Babenko. Neural Oblivious Decision Ensembles for Deep Learning on Tabular Data. *arXiv preprint arXiv:1909.06312*, 2019. URL https://arxiv.org/abs/1909.06312.

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*, 2018. URL https://arxiv.org/abs/1706.09516.

Ivan Rubachev, Artem Alekberov, Yury Gorishniy, and Artem Babenko. Revisiting Pretraining Objectives for Tabular Deep Learning. *arXiv preprint arXiv:2207.03208*, 2022. URL https://arxiv.org/abs/2207.03208.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, et al. DeepSeekMath: Pushing the Limits of Mathematical Reasoning in Open Language Models. *arXiv preprint arXiv:2402.03300*, 2024. URL https://arxiv.org/abs/2402.03300.

Safal Shrestha, Minwu Kim, and Keith Ross. Mathematical Reasoning in Large Language Models: Assessing Logical and Arithmetic Errors across Wide Numerical Ranges. *arXiv preprint arXiv:2502.08680*, 2025. URL https://arxiv.org/abs/2502.08680.

Ravid Shwartz-Ziv and Amitai Armon. Tabular Data: Deep Learning is Not All You Need. *arXiv preprint arXiv:2106.03253*, 2021. URL https://arxiv.org/abs/2106.03253.

Gowthami Somepalli, Micah Goldblum, Avi Schwarzschild, C. Bayan Bruss, and Tom Goldstein. SAINT: Improved Neural Networks for Tabular Data via Row Attention and Contrastive Pre-Training. *arXiv preprint arXiv:2106.01342*, 2021. URL https://arxiv.org/abs/2106.01342.

Andrew Trask, Felix Hill, Scott Reed, Jack Rae, Chris Dyer, and Phil Blunsom. Neural Arithmetic Logic Units. In *Advances in Neural Information Processing Systems*, 2018. URL https://arxiv.org/abs/1808.00508.

Silviu-Marian Udrescu and Max Tegmark. AI Feynman: a Physics-Inspired Method for Symbolic Regression. *Science Advances*, 2020. URL https://arxiv.org/abs/1905.11481.

Silviu-Marian Udrescu, Andrew Tan, Jiahai Feng, Orisvaldo Neto, Tailin Wu, and Max Tegmark. AI Feynman 2.0: Pareto-optimal symbolic regression exploiting graph modularity. In *Advances in Neural Information Processing Systems*, 2020. URL https://arxiv.org/abs/2006.10782.

Joaquin Vanschoren, Jan N. van Rijn, Bernd Bischl, and Luis Torgo. OpenML: networked science in machine learning. *ACM SIGKDD Explorations Newsletter*, 15(2):49–60, 2014. doi: 10.1145/2641190.2641198. URL https://arxiv.org/abs/1407.7722.

Johannes von Oswald, Eyvind Niklasson, Ettore Randazzo, João Sacramento, Alexander Mordvintsev, Andrey Zhmoginov, et al. Transformers learn in-context by gradient descent. In *Proceedings of the International Conference on Machine Learning*, 2023. URL https://arxiv.org/abs/2212.07677.

Sang Michael Xie, Aditi Raghunathan, Percy Liang, and Tengyu Ma. An Explanation of In-context Learning as Implicit Bayesian Inference. In *International Conference on Learning Representations*, 2022. URL https://arxiv.org/abs/2111.02080.

Keyulu Xu, Mozhi Zhang, Jingling Li, Simon S. Du, Ken-ichi Kawarabayashi, and Stefanie Jegelka. How Neural Networks Extrapolate: From Feedforward to Graph Neural Networks. *arXiv preprint arXiv:2009.11848*, 2020. URL https://arxiv.org/abs/2009.11848.

A Evaluation Protocol

The unit of aggregation is a task family. The benchmark contains 100 GSM8K and 14 AIME families, each with 2,048 generated rows. Tabular methods are evaluated at caps 32, 64, 128, 256, 512, 1,024, and 2,048; GPT-OSS-120B uses caps 32, 64, and 128 because of context-length constraints. Both methods receive examples and queries from the same tabular interface. Table 6 summarizes the evaluation grid.

Table 6: Task banks and evaluated row budgets. Both random and upper-output-tail splits are evaluated at each listed budget.

Source	Families	Tabular row caps	GPT-OSS row caps
GSM8K	100	32–2,048	32, 64, 128
AIME	14	32–2,048	32, 64, 128

The language-model harness requests a JSON prediction list after serialized context and query rows. Parsing failures are retried, up to ten attempts, with logging. Temperature is zero. Preprocessing parameters for the tabular methods are fit on the training split, and predictions are returned to the original target scale for rounded-accuracy scoring.

B Aggregation and Answer Resolution

Rounded accuracy first averages correctness over a task's query rows, then weights the task means equally within GSM8K or AIME. Regression summaries take the median of per-task scores. This preserves the distinction between answer frequency and the scale of prediction error across heterogeneous computations. Table 5 subtracts the displayed random-split mean from the upper-tail mean; Table 4 gives the multi-budget context.

The accuracy criterion rounds both predictions and reference values on the original answer scale. Some reference programs return continuous quantities, so rounded accuracy expresses unit-width answer agreement rather than exact real-number equality. Regression errors retain the distances discarded by that discrete criterion. Both metrics are needed to distinguish a small miss from a large numerical error.

C Extended Results

Table 7: AIME rounded accuracy across selected row caps for all ten methods. The task bank contains 14 families; dashes denote unevaluated language-model budgets.

Model	Rand. 32	128	2,048	Shift 32	128	2,048
GPT-OSS-120B	25.5	25.0	–	14.3	16.7	–
TabPFN v2.5	1.0	6.3	17.3	0.0	0.0	0.1
TabPFN v2	0.0	5.2	15.9	0.0	0.3	0.0
TabM	1.0	1.4	2.6	0.0	0.0	0.1
CatBoost	1.0	0.5	3.5	0.0	0.0	0.0
XGBoost	1.0	1.1	9.8	0.0	0.0	5.4
LightGBM	0.0	0.3	9.8	0.0	0.0	1.8
Random Forest	2.0	4.4	16.6	0.0	3.8	7.1
RealMLP-TD-S	0.0	0.3	1.4	0.0	0.0	0.0
xRFM	0.1	0.3	1.3	0.0	0.0	0.2

D ALL-TASK SCALING PROFILES

The full-panel plots extend Figures 4 and 5 to all ten methods and evaluated budgets. The regression comparison complements Figure 1, placing answer frequency alongside the relative magnitude of prediction errors.

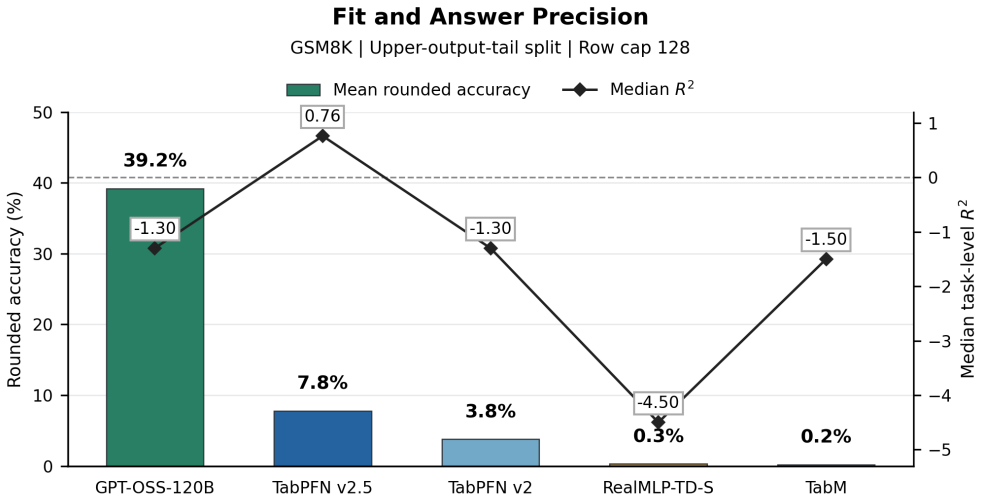


Figure 6: **Fit and answer precision under output-range shift.** GSM8K at a 128-row cap. Bars show mean rounded accuracy and diamonds show median R^2. GPT-OSS-120B answers more queries correctly while its larger errors yield negative median R^2; TabPFN v2.5 shows the converse pattern.

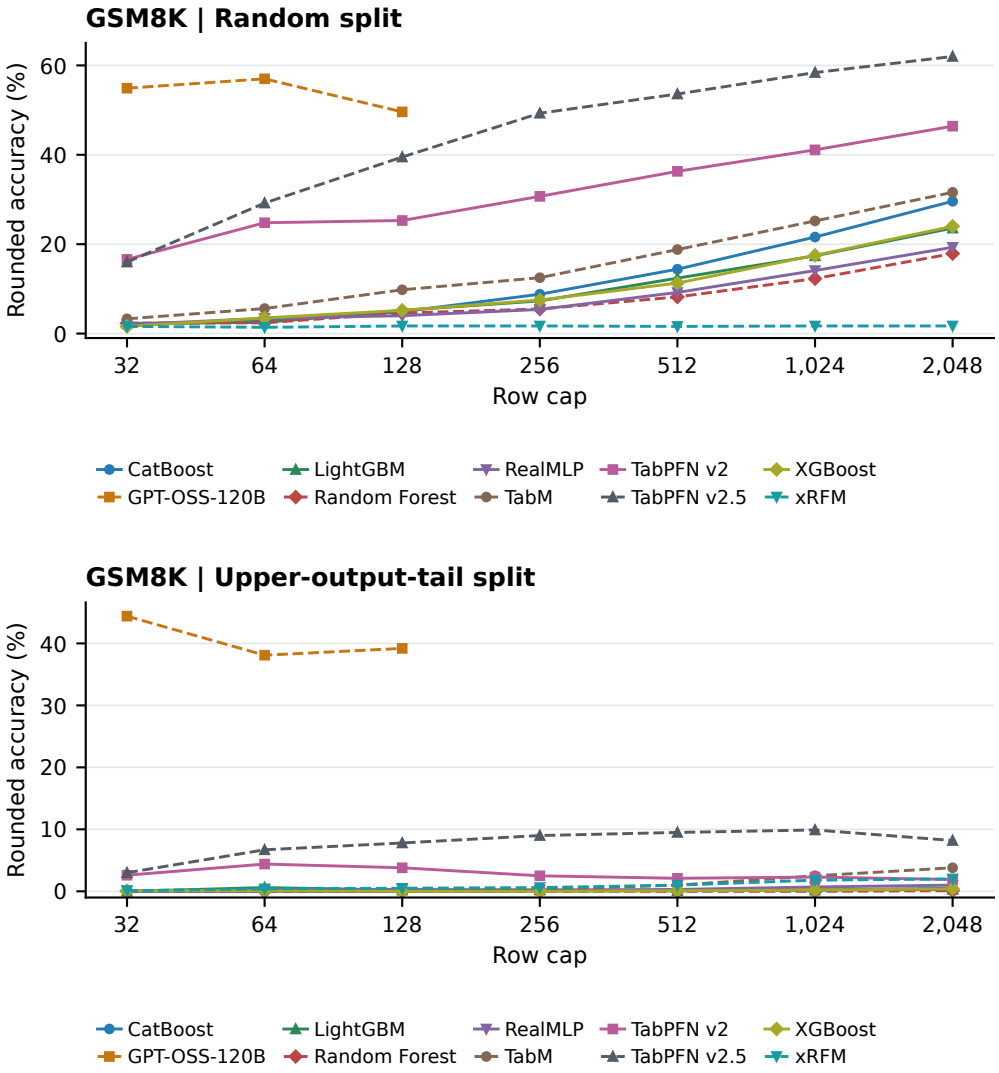


Figure 7: **GSM8K scaling across the complete model panel.** Top: random split. Bottom: upper-output-tail split. All methods use the 100-family task bank. Tabular methods extend to 2,048 rows, and GPT-OSS-120B is evaluated through 128 rows. Lines connect the evaluated budgets.

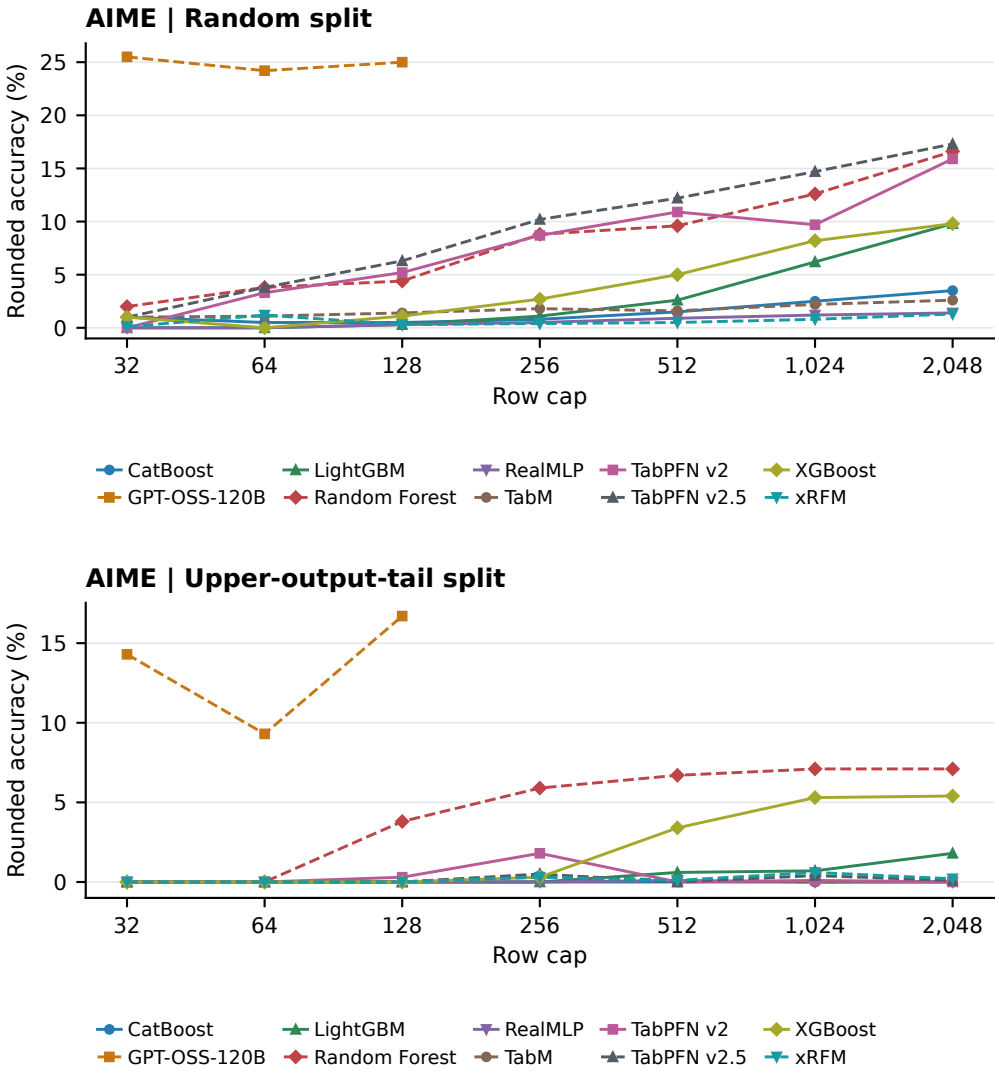


Figure 8: **AIME scaling over all 14 task families.** Top: random split. Bottom: upper-output-tail split. These panels extend Figure 5 to all ten methods and all evaluated row budgets. Competition-level computations present a difficult precision target under both regimes, especially in the upper output tail.

E Specification Construction and Evaluation Details

E.1 The Compiler's Contract

The synthesis prompt asks the teacher to generalize a seed while preserving its calculation. Its output contains five linked objects: text templates, typed slots, a generator, a verifier, and the seed's base assignment. The same slot names must appear in the templates, schema, generator output, and verifier input. A slot can represent a numerical quantity, an entity, a unit, or a categorical choice. This agreement is important because a program can execute correctly while a rendered question refers to a different quantity.

The teacher returns a single structured specification. The generator accepts a supplied random-number source and returns an assignment. The verifier accepts that assignment and returns a validity flag and a numeric answer. Templates use the assignment to express the question without exposing its answer. For tabular evaluation, the input columns are derived from the assignment and its generic features; the target is obtained by executing the verifier. Learners observe those rows, while the reference program remains part of the benchmark construction.

Table 8: Construction checks and the failure each detects. A candidate is revised when a check fails, within the 20-attempt synthesis budget.

Check	Failure detected
Schema agreement	A template requests a slot that the generator omits, or code uses an undeclared input.
Seed anchoring	The verifier fails to recover the published answer at the original assignment.
Valid generation	Sampled counts, denominators, or other quantities violate the task's constraints.
Deterministic execution	The same assignment receives different labels, or labeling fails to terminate within its budget.
Table integrity	Duplicate rows, nonnumeric targets, or malformed feature records enter the learning task.

Feedback identifies the concrete error, such as a missing slot, failed constraint, or seed-answer mismatch. The next proposal can revise the affected generator, verifier, or template. This local repair process makes the reference computation inspectable. The validation evidence consists of the seed and sampled assignments that pass these checks; expansion to a new parameter regime calls for checking that regime as well.

E.2 A Continuous-Quantity Example

The distance–rate–time family illustrates coherent sampling when quantities depend on one another. Let v be base speed, h a headwind reduction, and t elapsed time. The generator samples $v \in [6, 10]$, $h \in [1, 5]$, and $t \in [1, 10]$, then sets distance $d = (v - h)t$. A row exposes (d, h, t) and its label is

$$V(d, h, t) = d/t + h.$$

The intervals ensure positive effective speed and a nonzero denominator. The original example $(d, h, t) = (15, 3, 5)$ has answer 6. Sampling distance independently of the other quantities would require additional validity checks; constructing it from v, h, and t preserves the intended relation directly. This example complements the discrete flower-counting family in the main text: integer constraints and continuous arithmetic create different requirements for valid generation, while both support deterministic labels.

E.3 Representation and Split Construction

The representation combines primary numerical slots with generic arithmetic cues: log-magnitude $\log(1 + |x|)$, sign, parity, remainders for small moduli, and categorical encodings where applicable. These transformations expose common numerical structure without providing the family's answer program. An analysis of a particular family can inspect its column definitions alongside the verifier to identify which dependencies the learner must recover.

Each family starts with 2,048 deduplicated rows. The evaluated caps are 32, 64, 128, 256, 512, 1,024, and 2,048 total rows, with subsampling performed before the 80/20 split. The retained protocol uses sampling seed 2025. A random split assesses prediction within the sampled distribution. Sorting by the target before splitting puts the largest outputs in the query set; ties at the boundary remain possible. Both conditions retain the same underlying program, but they expose different regions of its output distribution.

Standardization uses training-split statistics. The evaluation code's feature-filling step precedes the split; all audited benchmark tables have complete features, so that step leaves these tables unchanged. TabPFN v2.5 uses the `v2.5_default` API configuration with eight estimators. The other tabular methods use their retained evaluation configurations. GPT-OSS-120B receives serialized context and query rows at caps 32, 64, and 128, with temperature zero and the parsing procedure described in Appendix A.

F Complete Budget Profiles and Regression Errors

The following tables retain the full seven-budget tabular comparison and complement rounded accuracy with regression errors. For a task's query set Q, let z and $\hat{z}$ be targets and predictions on the evaluation's standardized target scale. Then

$$\mathrm{RMSE} = \sqrt{|Q|^{-1}\sum_{j\in Q}(\hat{z}_j - z_j)^2}, \qquad \mathrm{MAE} = |Q|^{-1}\sum_{j\in Q}|\hat{z}_j - z_j|.$$

The tables use the median of each task-level regression metric and the mean of each task-level rounded accuracy. Standardized errors describe scale-relative prediction quality; rounded accuracy is computed after returning to the original answer scale. The complete profiles cover the 100 GSM8K and 14 AIME task families and the evaluated budgets for each method. GPT-OSS-120B has no reported standardized RMSE or MAE, so those cells are left blank. Its scale-invariant R^2 is reported alongside rounded accuracy.

The full budget profiles distinguish a persistent precision deficit from a small-data effect. Random-split accuracy typically benefits from more rows, whereas the upper-tail curves remain much flatter. Regression errors then show how much numerical structure survives when the integer-bin criterion is missed. The reference in R^2 is the query-set mean: a negative value means the model's squared error exceeds that constant reference on the query set.

F.1 GSM8K Budget and Error Profiles

Table 9: GSM8K rounded accuracy (%) at every evaluated tabular row cap. Random split. The benchmark contains 100 GSM8K families.

Model	32	64	128	256	512	1,024	2,048
GPT-OSS-120B	54.9	57.0	49.6	–	–	–	–
TabPFN v2.5	16.0	29.2	39.5	49.3	53.6	58.4	62.0
TabPFN v2	16.6	24.8	25.3	30.7	36.3	41.1	46.4
TabM	3.3	5.6	9.8	12.5	18.8	25.2	31.6
CatBoost	2.0	2.8	4.9	8.8	14.4	21.6	29.6
XGBoost	1.6	3.5	5.2	7.5	11.3	17.5	24.0
LightGBM	2.3	2.5	5.2	7.3	12.4	17.4	23.6
Random Forest	2.3	2.4	4.6	5.5	8.2	12.3	17.9
RealMLP-TD-S	2.0	3.4	4.0	5.4	9.2	14.1	19.3
xRFM	1.6	1.4	1.7	1.7	1.6	1.7	1.7

Table 10: GSM8K rounded accuracy (%) at every evaluated tabular row cap. Upper-output-tail split. The benchmark contains 100 GSM8K families.

Model	32	64	128	256	512	1,024	2,048
GPT-OSS-120B	44.4	38.1	39.2	–	–	–	–
TabPFN v2.5	3.0	6.7	7.8	9.0	9.5	9.9	8.2
TabPFN v2	2.6	4.4	3.8	2.5	2.1	2.3	1.9
TabM	0.0	0.2	0.2	0.3	1.0	2.5	3.8
CatBoost	0.0	0.0	0.0	0.1	0.1	0.2	0.5
XGBoost	0.1	0.1	0.0	0.0	0.1	0.2	0.4
LightGBM	0.0	0.6	0.2	0.2	0.1	0.2	0.3
Random Forest	0.0	0.0	0.0	0.0	0.0	0.0	0.1
RealMLP-TD-S	0.1	0.2	0.3	0.2	0.3	0.7	1.0
xRFM	0.1	0.4	0.5	0.6	1.0	1.8	2.0

Table 11: GSM8K regression errors across budgets. Each entry is the median of task-level scores. RMSE and MAE use training-standardized targets.

		Random			Upper tail		
Model	Cap	R^2	RMSE	MAE	R^2	RMSE	MAE
GPT-OSS-120B	128	0.84	–	–	-1.30	–	–
TabPFN v2.5	128	1.00	0.02	0.01	0.76	0.57	0.44
	256	1.00	0.01	0.01	0.76	0.46	0.37
	512	1.00	0.01	0.00	0.75	0.44	0.33
	1,024	1.00	0.01	0.00	0.59	0.53	0.33
	2,048	1.00	0.01	0.00	0.28	0.66	0.45
TabPFN v2	128	0.99	0.09	0.06	-1.30	1.67	1.33
	256	1.00	0.07	0.03	-1.30	1.55	1.22
	512	1.00	0.04	0.02	-1.40	1.61	1.16
	1,024	1.00	0.03	0.02	-1.40	1.56	1.19
	2,048	1.00	0.02	0.01	-1.40	1.50	1.09
TabM	128	0.96	0.20	0.14	-1.50	0.97	0.89
	256	0.98	0.12	0.08	-0.75	0.81	0.71
	512	0.99	0.07	0.05	-0.27	0.70	0.57
	1,024	1.00	0.04	0.03	0.03	0.58	0.48
	2,048	1.00	0.03	0.02	0.17	0.52	0.42
CatBoost	128	0.81	0.39	0.29	-5.20	1.55	1.38
	256	0.94	0.23	0.16	-4.00	1.37	1.22
	512	0.98	0.13	0.09	-2.80	1.21	1.07
	1,024	0.99	0.07	0.05	-2.00	1.05	0.88
	2,048	1.00	0.05	0.02	-1.50	0.96	0.76
XGBoost	128	0.82	0.41	0.30	-4.00	1.43	1.30
	256	0.89	0.31	0.21	-3.10	1.32	1.17
	512	0.96	0.19	0.13	-2.60	1.18	1.04
	1,024	0.98	0.13	0.09	-2.00	1.07	0.91
	2,048	0.99	0.09	0.06	-1.50	0.98	0.80
LightGBM	128	0.81	0.38	0.29	-3.40	1.24	1.12
	256	0.93	0.26	0.18	-2.40	1.13	1.00
	512	0.97	0.17	0.11	-1.90	1.04	0.90
	1,024	0.98	0.12	0.08	-1.70	1.00	0.85
	2,048	0.99	0.08	0.05	-1.60	0.96	0.81
Random Forest	128	0.79	0.43	0.32	-4.30	1.53	1.37
	256	0.88	0.34	0.23	-3.60	1.38	1.25
	512	0.93	0.25	0.18	-3.40	1.29	1.16
	1,024	0.96	0.20	0.14	-2.80	1.23	1.06
	2,048	0.98	0.15	0.10	-2.70	1.17	1.01
RealMLP-TD-S	128	0.77	0.46	0.32	-4.50	1.42	1.32
	256	0.88	0.32	0.23	-2.90	1.22	1.09
	512	0.95	0.23	0.15	-2.20	1.12	1.00
	1,024	0.98	0.14	0.09	-1.70	1.03	0.87
	2,048	0.99	0.08	0.05	-1.50	0.95	0.78
xRFM	128	0.90	0.32	0.20	-2.70	1.18	1.00
	256	0.96	0.19	0.13	-2.30	1.11	0.95
	512	0.99	0.12	0.07	-2.10	1.03	0.86
	1,024	0.99	0.07	0.04	-1.80	0.96	0.76
	2,048	1.00	0.04	0.03	-1.50	0.92	0.70

F.2 AIME Budget and Error Profiles

Table 12: AIME rounded accuracy (%) at every evaluated tabular row cap. Random split. The benchmark contains 14 AIME families.

Model	32	64	128	256	512	1,024	2,048
GPT-OSS-120B	25.5	24.2	25.0	–	–	–	–
TabPFN v2.5	1.0	3.8	6.3	10.2	12.2	14.7	17.3
TabPFN v2	0.0	3.3	5.2	8.7	10.9	9.7	15.9
TabM	1.0	1.1	1.4	1.8	1.6	2.2	2.6
CatBoost	1.0	0.5	0.5	0.8	1.5	2.5	3.5
XGBoost	1.0	0.0	1.1	2.7	5.0	8.2	9.8
LightGBM	0.0	0.0	0.3	1.1	2.6	6.2	9.8
Random Forest	2.0	3.8	4.4	8.8	9.6	12.6	16.6
RealMLP-TD-S	0.0	0.0	0.3	0.5	0.9	1.2	1.4
xRFM	0.1	1.2	0.3	0.4	0.5	0.8	1.3

Table 13: AIME rounded accuracy (%) at every evaluated tabular row cap. Upper-output-tail split. The benchmark contains 14 AIME families.

Model	32	64	128	256	512	1,024	2,048
GPT-OSS-120B	14.3	9.3	16.7	–	–	–	–
TabPFN v2.5	0.0	0.0	0.0	0.5	0.0	0.4	0.1
TabPFN v2	0.0	0.0	0.3	1.8	0.0	0.1	0.0
TabM	0.0	0.0	0.0	0.4	0.1	0.0	0.1
CatBoost	0.0	0.0	0.0	0.0	0.0	0.0	0.0
XGBoost	0.0	0.0	0.0	0.3	3.4	5.3	5.4
LightGBM	0.0	0.0	0.0	0.0	0.6	0.7	1.8
Random Forest	0.0	0.0	3.8	5.9	6.7	7.1	7.1
RealMLP-TD-S	0.0	0.0	0.0	0.0	0.1	0.0	0.0
xRFM	0.0	0.0	0.0	0.3	0.1	0.6	0.2

Table 14: AIME regression errors across budgets. Each entry is the median of task-level scores. RMSE and MAE use training-standardized targets.

		Random			Upper tail		
Model	Cap	R^2	RMSE	MAE	R^2	RMSE	MAE
GPT-OSS-120B	128	-0.31	–	–	-1.90	–	–
TabPFN v2.5	128	0.80	0.19	0.10	-1.30	5.60	4.80
	256	0.77	0.30	0.08	-0.69	4.80	3.90
	512	0.77	0.33	0.13	-0.77	4.50	3.80
	1,024	0.84	0.33	0.10	-0.53	4.80	4.00
	2,048	0.81	0.41	0.08	-1.20	6.70	5.40
TabPFN v2	128	0.75	0.30	0.21	-1.50	7.30	5.60
	256	0.70	0.35	0.12	-1.20	6.50	5.60
	512	0.73	0.44	0.19	-1.60	6.50	5.00
	1,024	0.79	0.45	0.20	-1.20	6.80	5.40
	2,048	0.83	0.35	0.12	-1.20	6.30	4.80
TabM	128	0.59	0.45	0.30	-1.00	1.99	1.31
	256	0.78	0.30	0.15	-0.90	1.93	1.16
	512	0.88	0.31	0.18	-0.55	1.70	1.07
	1,024	0.83	0.38	0.12	-0.61	1.64	1.04
	2,048	0.81	0.36	0.09	-0.46	1.64	0.97
CatBoost	128	0.56	0.54	0.33	-2.30	2.19	1.48
	256	0.60	0.42	0.27	-2.00	2.14	1.39
	512	0.75	0.45	0.22	-1.50	2.03	1.23
	1,024	0.77	0.44	0.15	-1.20	1.94	1.14
	2,048	0.79	0.38	0.13	-1.10	1.82	1.08
XGBoost	128	0.38	0.56	0.37	-2.10	2.08	1.40
	256	0.54	0.51	0.31	-2.50	2.01	1.27
	512	0.72	0.49	0.26	-1.40	2.00	1.17
	1,024	0.74	0.48	0.17	-1.70	1.88	1.12
	2,048	0.78	0.43	0.16	-1.50	1.79	1.12
LightGBM	128	0.47	0.64	0.48	-2.00	2.09	1.37
	256	0.53	0.52	0.37	-1.60	1.95	1.30
	512	0.75	0.45	0.27	-1.30	1.94	1.17
	1,024	0.74	0.48	0.21	-1.10	1.87	1.13
	2,048	0.78	0.42	0.20	-1.10	1.80	1.14
Random Forest	128	0.36	0.54	0.40	-2.20	2.12	1.41
	256	0.44	0.58	0.34	-2.00	2.06	1.39
	512	0.65	0.52	0.29	-1.50	2.01	1.28
	1,024	0.77	0.46	0.21	-1.50	1.97	1.17
	2,048	0.78	0.45	0.19	-1.50	1.95	1.13
RealMLP-TD-S	128	0.33	0.68	0.41	-2.00	2.10	1.46
	256	0.56	0.48	0.32	-2.30	2.05	1.38
	512	0.70	0.50	0.26	-1.30	2.03	1.27
	1,024	0.78	0.52	0.17	-1.30	1.98	1.16
	2,048	0.83	0.47	0.13	-1.20	1.90	1.16
xRFM	128	0.54	0.59	0.38	-2.00	2.07	1.31
	256	0.62	0.42	0.22	-2.50	2.09	1.33
	512	0.71	0.46	0.24	-1.30	1.90	1.15
	1,024	0.82	0.42	0.15	-1.30	1.87	1.14
	2,048	0.80	0.39	0.19	-1.40	1.90	1.13

G Numerical Generalization in Context

The tabular-learning landscape. Comparative studies relate the performance of neural models and tree ensembles to dataset properties, tuning, and computation (Shwartz-Ziv & Armon, 2021; Grinsztajn et al., 2022; McElfresh et al., 2023). NODE learns differentiable ensembles of oblivious decision trees (Popov et al., 2019); TabNet and TabTransformer use feature selection or contextual feature representations (Arik & Pfister, 2021; Huang et al., 2020). Non-Parametric Transformers attend across datapoints, enabling a prediction to depend explicitly on other rows (Kossen et al., 2021). These architectural choices provide different ways to exploit tabular structure. TABULAR-

MATH adds a computational target to this landscape: all rows in a family arise from the same executable relationship, so evaluation can separately measure statistical fit and the accuracy of the answers that relationship defines.

Task priors and task-specific fitting. Prior-fitted networks learn a prediction algorithm from synthetic tasks (Müller et al., 2022; Hollmann et al., 2022; 2025; Grinsztajn et al., 2025). Studies of in-context learning ask which functions and learning procedures a transformer can recover from examples (Garg et al., 2022; Akyürek et al., 2023); Bayesian and gradient-based accounts analyze this behavior in structured settings (Xie et al., 2022; von Oswald et al., 2023). Tabular architectures emphasize feature representations, attention, and efficient fitting on individual datasets (Gorishniy et al., 2021; 2022; Somepalli et al., 2021; Gorishniy et al., 2025), while pretraining objectives offer another way to transfer useful structure (Rubachev et al., 2022). TABULARMATH connects these approaches to the same deterministic task: observe rows from a fixed computation and return its outputs on new inputs. The row-cap curves measure how additional examples change that ability under random and upper-tail queries.

Numerical fit and explicit rule recovery. Neural arithmetic logic units and neural arithmetic units build addition and multiplication into learnable modules (Trask et al., 2018; Madsen & Johansen, 2020). Symbolic regression searches for an expression, as in PySR and AI Feynman, while SRBench evaluates both predictive performance and recovery of known equations (Cranmer, 2023; Udrescu & Tegmark, 2020; La Cava et al., 2021). Program synthesis extends the representation to executable procedures (Austin et al., 2021; Chen et al., 2021). These approaches differ in the rules they can express, but all can be evaluated through the benchmark's numerical interface. Its labels come from a known program, and the metric asks whether the returned answers preserve a declared precision as the queried range changes.

Searching for and learning symbolic structure. Deep symbolic regression trains a policy to search over expressions (Petersen et al., 2021). Neural Symbolic Regression and end-to-end transformer approaches learn to propose equations from observed values (Biggio et al., 2021; Kamienny et al., 2022). AI Feynman 2.0 exploits modular structure, while symbolic distillation extracts mathematical relationships from learned graph-network representations (Udrescu et al., 2020; Cranmer et al., 2020). These approaches make expression recovery an explicit modeling objective. TABULARMATH evaluates the complementary consequence: the numerical answers produced on held-out rows. This accommodates several representations of the same computation and makes precise output-range generalization the shared success criterion.

Arithmetic representation and extrapolation. Studies of transformer arithmetic show that number formatting, training examples, and positional representations materially affect generalization (Nogueira et al., 2021; Lee et al., 2023; Jelassi et al., 2023; McLeish et al., 2024). Theoretical and empirical work on neural-network extrapolation connects behavior outside the training distribution to architectural and feature-level structure (Xu et al., 2020). These findings motivate treating the representation and split as part of the evaluated task. TABULARMATH supplies declared numerical columns and generic arithmetic features, then queries a different output region. The resulting score asks whether a predictor preserves the computation's answers under that shift, complementing the digit-length shifts studied in sequence-based arithmetic.

Composition and numerical range are distinct generalization axes. SCAN, COGS, and CFQ construct splits that test systematic recombination of known elements (Lake & Baroni, 2018; Kim & Linzen, 2020; Keysers et al., 2020); broader analyses organize compositional generalization along several dimensions (Hupkes et al., 2019). Computation-graph analyses connect reasoning difficulty to the composition of intermediate subprocedures (Dziri et al., 2023). TABULARMATH retains a family's reference computation across training and test and varies which numerical region is queried. The invariant program provides a concrete reference for interpreting failure: the learned predictor tracks a relationship within the observed region but may lose answer precision as the query range expands.

What is held fixed under output shift. For a deterministic target $y = V(x)$, conditioning on the upper tail of y selects inputs through V. Some families reach that tail through one large input;

others require a particular combination, a discrete branch, or several interacting constraints. Thus the reference computation stays fixed while the joint distribution of input and output changes. This gives the split a concrete interpretation: learning from lower-output examples and predicting the computation's upper-output region. GSM-Ranges studies numerical-scale changes in word problems (Shrestha et al., 2025); TABULARMATH studies this challenge from tabular examples, exposing the relation between statistical fit and answer precision.

Different interfaces to a mathematical computation. MathQA associates word problems with operation-based representations, SVAMP probes robustness to variations in elementary problems, and DROP requires discrete reasoning over text (Amini et al., 2019; Patel et al., 2021; Dua et al., 2019). Minerva, Llemma, and DeepSeekMath develop mathematical capabilities through specialized training data and objectives (Lewkowycz et al., 2022; Azerbayev et al., 2023; Shao et al., 2024); PAL and Program of Thoughts connect language reasoning to executable solutions (Gao et al., 2023; Chen et al., 2023). TABULARMATH changes the learning interface from a verbal problem to numerical examples of its computation. This lets tabular predictors and learned task priors confront a mathematical target through the input representation they are designed to use.

Representation and answer resolution. The common input columns include generic arithmetic features, so the evaluated task is learning from that declared representation. Rounded accuracy fixes a common unit-width output resolution, while R^2 measures error relative to target variation. These choices define complementary properties of a predictor. A one-unit error can be negligible relative to a large target variance and still cross an answer-bin boundary. The task-level scatter establishes that this separation occurs within individual families, and the numerical-resolution audit checks that target storage preserves the bins used for scoring. Together, the representation, split, and output resolution make the computational objective explicit.

Reusable tasks and comparable evaluations. OpenML organizes datasets, tasks, and runs for shared analysis (Vanschoren et al., 2014), while scikit-learn provides common estimator and evaluation interfaces (Pedregosa et al., 2011). TABULARMATH contributes a reusable family-level object to this evaluation tradition: a valid-input generator, a deterministic answer program, declared splits, and a common answer resolution. The family structure enables additional rows and shifted queries to be generated from the same computation. Task-level metrics then preserve the differences among families that an overall regression average can obscure.