Do Tabular Foundation Models Beat Trees on Messy, Real-World Data?

A Zero-Shot Benchmark of TabICLv2 Against Gradient-Boosted Trees on CIC-IDS2017

Caleb Dame
Johns Hopkins University
EN.705.742: Advanced Applied Machine Learning
cdame1@jhu.edu

Summer 2026

Abstract

Gradient-boosted trees have long been the industry choice for tabular classification, and on the default CIC-IDS2017 intrusion-detection dataset they routinely report over 99% macro-F_1. Tabular foundation models in the TabPFN/TabICL line claim to consistently surpass tuned trees on standard benchmark prediction tasks by using in-context learning: one forward pass over a labeled context set to predict an unlabeled inference set, with no per-dataset training. This study tests whether one of these models, TabICLv2, holds up on a messy, heavily imbalanced real dataset. The study also looks past the headline performance metrics and at the residual structure: if the two model families make different structural mistakes, and whose probabilities are better calibrated. Since inference-set size is a bottleneck for in-context learning, in each seed the evaluation set is sampled from the held-out data and trimmed to 100k flows. The benign, high-frequency mega-classes are subsampled, with the final metrics reweighted to the natural prior to approximate global performance. Each training context is drawn without oversampling albeit with a floor that keeps at least two instances of every class. A naive prior-shift correction maps predicted probabilities back to the global prior before the estimated global calibration is measured. The TabICLv2 scores are repeatedly compared against tuned XGBoost and LightGBM models at multiple increasing context sizes. The design separates three questions: parity in performance, error diversity and potential benefits to ensembles, and calibration.

1 Problem Introduction

On tabular data, gradient-boosted decision trees such as XGBoost [8] and LightGBM [9] are still the models to beat [7]. That is what makes recent claims surrounding high-performing tabular foundation models so interesting. TabPFN [3, 4] and the TabICL line [5, 6] are transformers pretrained once on synthetic tabular tasks that perform *in-context learning* (ICL) for inference, predicting test labels using a context of labeled data in a single forward pass, with no gradient updates, no parameter tuning, and no per-task artifacts. Their authors report parity with, or wins over, tuned trees on standard classification benchmark suites.

However, curated benchmarks are clean and many realistic production datasets usually are not. CIC-IDS2017 [1] is a good stress case: about 2.8 million network flows in the original dataset release,

78 numeric features each, labeled benign or one of fourteen attack types. Tree ensembles report over 99% macro-F_1 on it, but that number deserves some suspicion. Part of it reflects dataset flaws: near-duplicate flows and time leakage (see [2]). An aggregate score near 1.0 also says very little about the rarest attacks. Heartbleed appears 11 times in the whole dataset, SQL injection only 18, and classes this small are often the ones where practitioners most want improvement. The imbalance is also what makes the foundation-model comparison non-trivial: a natural sample of 100k rows contains, in expectation, well under one Heartbleed flow, so the rare classes can vanish from the naive ICL context entirely, and any oversampling changes the class prior that predictions are made against.

The study is organized around three hypotheses, fixed before the main runs:

H1. Performance. Zero-shot TabICLv2 will consistently improve upon gradient boosting-based algorithms on macro-F_1 across context sizes.

H2. Error diversity. As the architectures are fundamentally different (greedy tree splits versus attention), their errors will be less correlated than the errors of similar tree-based algorithms leading to ensembling being more optimal than either alone.

H3. Calibration on rare attacks. TabICLv2 will be better calibrated than trees on rare classes, since it does not use impurity-based probability estimates that notoriously miscalibrate scores for classes with few examples.

H1 is the obvious raw numerical comparison. H2 and H3 are where a foundation model could be useful to the field even if it is not an outright winner, as a decorrelated and possibly better-calibrated alternative.

2 Related Work Motivating Design

2.1 Trees as the tabular default

Grinsztajn et al. [7] explain why tree ensembles keep winning on tables-based data: robustness to uninformative features, invariance to monotone feature transformations, and axis-aligned decision boundaries that match how many tabular targets behave. This study uses two gradient boosters, XGBoost and LightGBM; CatBoost [10], random forests, and extremely randomized trees [11] were initially considered as additional benchmark candidates but were abandoned: the extra compute was hard to justify given how much they overlapped, and adding them would only dilute the statistical power of the TabICLv2 comparison.

2.2 Tabular foundation models

TabPFN [3] reframed tabular classification as a single forward pass of a transformer pretrained on millions of synthetic datasets, with no tuning at deployment, though it was limited to about a thousand rows. Progress since has been a race to scale the context and chase the remaining marginal performance: TabPFN v2 [4] reached roughly ten thousand rows with alternating row- and column-wise attention, TabICL [5] about half a million through a two-stage column-embedding-then-row-interaction encoder, and TabICLv2 [6], the model tested here, a query-adaptive sparse attention (QASS) that stays sharp on contexts larger than most of its synthetic training tasks. The field is moving quickly, but TabICLv2 is a natural, performant, and open representative for a study whose limiting variable is how large a context the model can take in at once.

2.3 How these models are evaluated

These models are usually benchmarked on curated suites of small-to-medium, relatively clean datasets, where the headline is aggregate accuracy at parity with or above tuned tree ensembles. That leaves two things untested that this study targets: behavior on messy, severely imbalanced real datasets, and the rare-class and calibration behavior that aggregate metrics can hide. Intrusion detection, where the attacks that matter appear a handful of times in millions of flows, is a sharp case of exactly that gap.

2.4 CIC-IDS2017 and its pitfalls

Engelen et al. [2] audited CIC-IDS2017 and found mislabeled flows, feature-extraction artifacts, and redundancy that inflates random-split evaluation. They released a corrected re-derivation that also distinguishes successful from "attempted" attacks. Their corrected labels and reduced dataset are the primary source here, together with evaluation choices (Section 3) intended to avoid the leakage paths they identify.

2.5 Calibration and label shift

Calibration is measured by the expected calibration error (ECE) [16, 13], and it is fragile under class imbalance and under any reweighting of the training distribution. When a model is trained under one class prior and evaluated under another, the standard remedy is to adjust its predicted posteriors in closed form rather than retrain: Saerens et al. [19] and Elkan [20] give the correction for a known shift, and later work extends it to estimated shifts [21]. Here the shift is known exactly, because the context is constructed rather than sampled (Section 3.5). For error diversity, pairwise measures such as the error correlation and Yule's Q [18] quantify whether two classifiers disagree enough for an ensemble to help. Whether a combination actually improves is a separate question: McNemar's test [17] is the standard single-split comparison, but this study tests it with a paired test across independent replicate seeds instead, which also captures the split-to-split variability a single-set test ignores.

3 Method

Intrusion detection is treated here as multi-class tabular classification: each flow of 82 associated features is assigned a probability vector over C classes ($C = 16$ or 27 depending on the label set, Section 3.1). Benign flows are roughly 75% of the data, and the rarest attacks appear only a handful of times in the $\sim$2.1M cleaned flows, from 11 Heartbleed down to as few as 5 attempted SQL injections under the finer labels. Figure 1 shows the pipeline end to end.

3.1 Dataset

The original CIC-IDS2017 release [1] is eight capture files spanning the five weekdays (Monday through Friday), 2,830,743 labeled flows with 78 features each, and 15 classes (either benign or one of fourteen attack types). Benign is 80.3% of the flows, and the three rarest attacks (Heartbleed, SQL injection, Infiltration) together are 68 rows. The Engelen re-derivation [2] used here re-extracts the flows with corrected labels and separates successful attacks from "$-$Attempted" ones. After cleaning (drop identifier columns, map infinities to NaN, coerce features to numeric, drop rows with pathological missing values, remove exact duplicate flows, and drop the capture-day and time-of-day

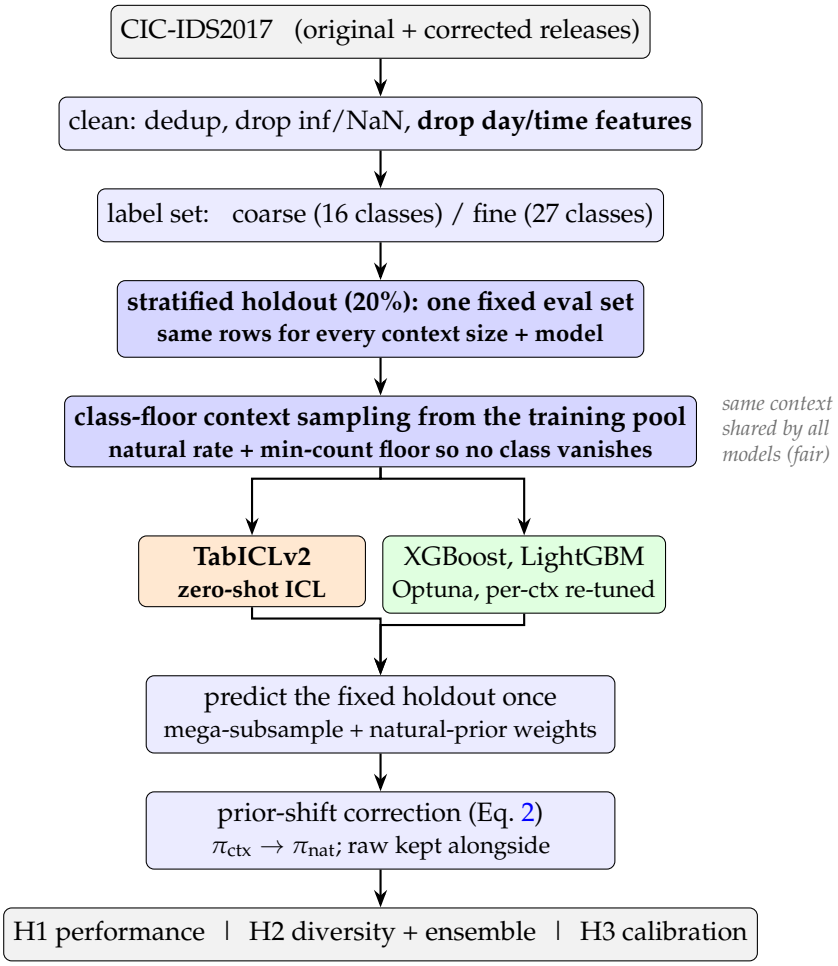

Figure 1: Pipeline. One stratified holdout is fixed across all context sizes and models; each run's natural-prior context (drawn from the complementary training pool) is shared by TabICLv2 and the tuned trees, the fixed holdout is predicted once, prior-corrected, and fed to the H1/H2/H3 analyses. The context-size sweep (Section 4) varies the size of that context.

features of Section 3.7), 2,099,967 flows and 82 numeric flow features remain, and benign falls to 75.4%. The feature count is four above the original 78 as the re-derivation creates five statistics the 2017 version did not have (2 different RST-flag counts, ICMP type and code, and total TCP flow time) while dropping a duplicate column.

Two label sets are built from the same cleaned flows (Section 3.1): the **coarse** set folds each "−Attempted" variant into its base attack for 16 classes (one more than the original fifteen as the re-derivation also distinguishes the initial stage of an Infiltration attack as its own class, Infiltration–Portscan), and the **fine** set keeps them separate for a total of 27 classes. Table 1 lists every class count. The imbalance now spans more than five orders of magnitude, from 1.58M benign flows down to 11 Heartbleed and, in the fine set, 5 attempted SQL injections, which is the regime the sampling, weighting, and prior-shift machinery below is built for.

Table 1: Class counts on the cleaned corrected release (2,099,967 flows, 82 features). The **coarse** 16-class label set folds "−Attempted" into the base attack; the **fine** set keeps the split (27 classes: the same 16 base classes plus the 11 attempted variants shown here). A dash marks classes with no variants.

	coarse (16)	fine (27)	
class	count	successful	attempted
BENIGN	1,582,559	1,582,559	–
Portscan	159,066	159,066	–
DoS Hulk	159,048	158,468	580
DDoS	95,144	95,144	–
Infiltration – Portscan	71,767	71,767	–
DoS GoldenEye	7,647	7,567	80
DoS Slowloris	5,705	3,859	1,846
DoS Slowhttptest	5,108	1,740	3,368
Botnet	4,803	736	4,067
FTP-Patator	3,984	3,972	12
SSH-Patator	2,988	2,961	27
Web Attack – Brute Force	1,365	73	1,292
Web Attack – XSS	673	18	655
Infiltration	81	36	45
Web Attack – SQL Injection	18	13	5
Heartbleed	11	11	–
total	2,099,967	2,099,967	

3.2 In-context learning with TabICLv2

TabICLv2 is used zero-shot via the open-weights `tabicl` package (`TabICLClassifier`, default v2 checkpoint). Given a labeled context set $\mathcal{D}_{\text{context}} = \{(x_i, y_i)\}_{i=1}^{n}$ and a test point $x_\star$, it outputs

$$\hat{p}(y_\star \mid x_\star, \mathcal{D}_{\text{context}}) = q_\theta(y_\star \mid x_\star, \mathcal{D}_{\text{context}}) \tag{1}$$

in one forward pass with the pretrained weights θ fixed. Internally it embeds each column value, compresses each row of combinations of column values to a single per-row vector with a row-wise transformer, and lets the test row attend over the embedded context rows [6]; Figure 2 traces this path. No fitting then occurs at or prior to deployment: the labeled context enters the same forward pass as the test row and takes the place of a trained parameter set, with the context set playing the role a tuning step plays for an ordinary learner and the context size n as the natural scaling axis (Section 4). The prediction is itself an average over an internal ensemble of members (feature- and class-order shuffles); the default eight-member average is reported throughout unless otherwise stated.

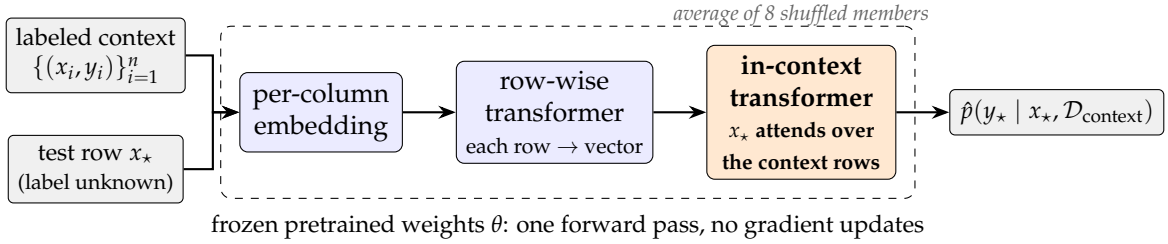

Figure 2: In-context learning in TabICLv2.

3.3 Tree baselines and tuning

The competitive baselines chosen for this study are XGBoost and LightGBM. Each is tuned with a Bayesian Search using Optuna [12] under stratified k-fold cross-validation on the context set, and re-tuned at every context size so hyperparameters can be allowed to adapt optimally to the amount of data in the training context. Three choices matter for fairness:

- **Objective.** Tuning targets macro one-vs-rest average precision rather than macro-F_1. It is more continuous and threshold-free, which gives the tuning regime a smoother signal under the heavy imbalance; macro-F_1 is still the reported headline metric.
- **No class re-weighting.** Class weights are not tuned. Balancing the loss shifts predicted probabilities off the natural prior (the same distortion as oversampling) and would undermine any calibration comparison. The trees see the exact same context as TabICLv2, with unweighted loss to fit the estimator.
- **Complexity caps.** Boost depth is capped at 10 with the number of estimators capped at 300 to keep the search in reasonable regions and the tuning tractable. This proved to be largely non-binding given the optimal parameters that were decided upon for most seeds and context sizes.

Each seeded run with its own context gets a single Optuna search of 30 trials, inner cross-validation capped at four folds (fewer when a class is too small to stratify). Each seed redraws the split and context and runs its own search, so the reported tree spread is cross-seed, not cross-search.

3.4 Context construction

At any context budget below the full dataset, a natural sample loses the rare classes entirely (a model cannot predict a class it never sees). Simply oversampling them back in is the obvious fix and an unusable one, since it changes the class prior that calibration is measured against. The context is instead drawn with a *natural-prior sampler with a per-class floor*: to a target budget, each class is sampled at its natural rate but with a minimum of $\kappa = 2$ rows so none can vanish completely, and never with replacement. Natural-rate sampling keeps the induced prior shift as small as the coverage constraint allows and preserves the challenge of rare class prediction. Every model fits the same context. The floor only lifts the very rarest classes above their natural rate enough to perform cross validation. For the residual shift that remains, it is corrected explicitly, as explained in Section 3.5.

3.5 Prior-shift correction

Since $\mathcal{D}_{\text{context}}$ is constructed, the original context prior π_{ctx} is known exactly, as is the natural prior π_{nat} from the full dataset. A classifier conditioned under π_{ctx} but evaluated under π_{nat} faces a pure label shift, and the standard closed-form adjustment [19, 20] is

$$p_{\text{corr}}(y = c \mid x) \;=\; \frac{\hat{p}(y = c \mid x)\,\dfrac{\pi_{\text{nat}}(c)}{\pi_{\text{ctx}}(c)}}{\sum_{c'} \hat{p}(y = c' \mid x)\,\dfrac{\pi_{\text{nat}}(c')}{\pi_{\text{ctx}}(c')}}, \tag{2}$$

applied with the run's recorded context counts as post-processing on model scores, costing next to nothing, and applied identically to every model. Calibration is reported both raw and corrected;

the gap measures how much a model leaned on the shifted context prior (Section 5). Because the natural-prior sampler keeps that shift small, the correction is minor here for all but the smallest context sizes.

3.6 Subsample evaluation

Stratified K-fold cross-fitting scores the whole dataset K times, and the foundation model's inference time grows with context times rows scored: at the 100k–300k contexts here that is days of GPU time per configuration, for little gain. The main question is *held-out generalization*, and the error bars come from multiple seeds of smaller evaluations, not fold structure (Section 3.8). Evaluation is therefore always on data from a single static 20% holdout and **fixed across every seed and method**, so the scaling curve is measured on identical rows and each context predicts it once. The context is always sampled from the complementary 80% training pool (Section 3.4) so the two never overlap.

Scoring the whole evaluation dataset through the foundation model is still dominated by the benign mega-classes, a slow, low-value bottleneck for TabICLv2. To enable more trials and more statistical power at fixed compute, the study subsamples them and *re-inflates* the resulting metrics using weights during metric evaluation. With n_c the count of class c in the holdout, every row of the rare and mid-frequency classes is kept and each very large class is subsampled to a cap m so each kept row when generating metrics then carries the weight

$$w_i = \frac{n_c}{\min(n_c, m)}, \tag{3}$$

so a metric computed on the subsample estimates its value on the full, un-subsampled holdout with all the mega-classes present at their natural prior. This re-inflation is applied to every count-dependent metric (overall ECE, and, through the confusion matrix, precision, macro-F_1, and macro average precision); per-class recall and balanced accuracy are within-class ratios and need none. **Every macro-F_1, ECE, and Brier figure reported below is this re-inflated, full-holdout estimate; for brevity it is written simply "macro-F_1", "ECE", and "Brier" from here on**, and the macro average across classes is uniform (each class counts equally).

3.7 Leakage controls

Following the audit in [2]: (i) the capture-day and time-of-day features are dropped, since each attack was run on a specific day and those features would let a model identify attacks by collection schedule rather than network behavior under a random split; (ii) exact duplicate flows are removed; (iii) the corrected re-derivation is the primary label source. A capture-day holdout would be more conservative but removes single-day attacks from training entirely, and doing so would limit the ability to answer the rare-class questions this study is about.

3.8 Metrics

Throughout, *rare* refers to the five attack classes with fewer than 2000 natural instances (Heartbleed, SQL injection, Infiltration, Web XSS, and Web brute force in Table 1); *common* refers to the remaining eleven. Metrics reported by frequency use this split. For H1: macro-F_1 over supported classes, balanced accuracy, per-class recall, and macro average precision. For H2: three diversity measures between model pairs, the error correlation, Yule's Q, and the normalized mutual information, each split into

rare and common classes. Since low correlation only shows an ensemble *could* help, several combiners (Section 5.2) are then scored on rare-class average precision, the metric that sees the tail, and on macro-F_1. The headline question, whether adding TabICLv2 to a tree beats adding a second tree, is tested by a paired Wilcoxon signed-rank across the replicate seeds. For H3: the multiclass Brier score on the rare-attack rows together with overall and per-class ECE, on both raw and prior-shift-corrected probabilities.

4 Experimental Protocol

The design is a full grid: zero-shot TabICLv2 and both freshly tuned trees are scored at every context size, on both label sets, each cell a ten-seed replicate on the same fixed holdout.

- **Context ladder.** 5k, 10k, 20k, 50k, 100k, 150k, 200k, 300k, 500k, 1M, and the full training partition (about 1.7M rows, no sampling, the *Max* point). The four smallest contexts additionally run TabICLv2 with sixteen ensemble members (Section 3.2).
- **Label granularity.** The coarse 16-class and fine 27-class label sets run the identical ladder; the fine per-class numbers are read with the rare-class caveat of Section 3.1.
- **Tree tuning.** A fresh Optuna search at *every* rung, so hyperparameters track the amount of data, with independent re-tuning all the way up to 1.7M rather than settings reused from a smaller context.
- **Hardware.** The main ladder through 300k ran on a 24 GB cloud GPU (an NVIDIA A10G); past 300k, TabICLv2 inference continued on a university-provided RTX PRO 6000 Blackwell (about 97 GB), with the trees re-tuned there.

The context limit is wall-clock, not memory: up to a few hundred thousand rows TabICLv2 stays inside the 24 GB card by shrinking its internal batch, and beyond that it offloads intermediate attention to host memory and per-row predict time climbs sharply. The Blackwell card's larger memory absorbs that super-linear cost so the foundation model reaches the whole partition; nothing about the model changes across the handoff, only the hardware, which is there to see whether the foundation model, given the compute, closes the gap the trees open with more data.

5 Results

The ladder of Section 3 was run: the trees are freshly tuned with an independent Optuna search at every context and compared to TabICLv2 from 5k of context to the complete ∼1.68M-row training partition (*Max*). Each configuration is run as **ten** independently seeded replicates (each a fresh evaluation-set sample, context sample, and set of model seeds), so every number below carries a cross-seed mean and standard error, the error bars in every figure.

5.1 H1: performance

H1 is read from macro-F_1 as context size grows, with TabICLv2 and both tuned trees scored on identical data at every rung (Figure 3). Parity inside any tested context window would support a weak form of H1: the zero-shot model keeps up with tuned boosters, outpacing them most at low context, where pre-training helps the foundation model most, and staying the plausible default out to Max.

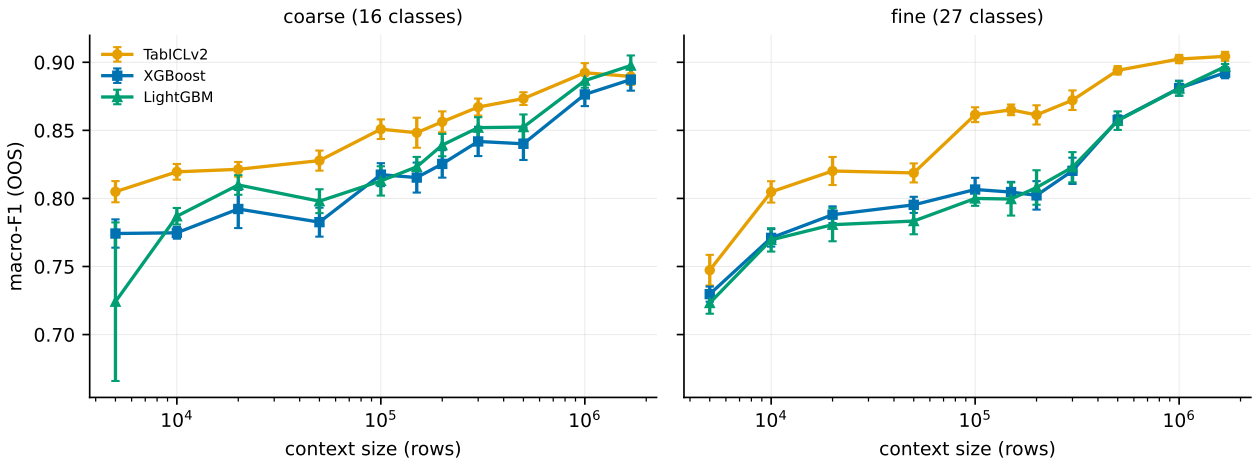

Figure 3: Macro-F_1 (fixed holdout) vs. context size, coarse 16-class (left) and fine 27-class (right) label sets on a shared axis. TabICLv2 and both trees run the full ladder on the same held-out rows, out to the complete ∼1.68M-row partition (*Max*). Markers are the mean over ten replicate seeds; error bars are the cross-seed standard error. TabICLv2 leads at every context on both label sets, except the coarse Max rung where the models fall well within two standard errors of each other.

With the ladder complete the pattern is clear (Figure 3, Table 2). Zero-shot TabICLv2 has the highest macro-F_1 at nearly every context: about 0.81 at 5k, where the boosters have almost nothing to tune on, rising to 0.89 by one million rows, ahead of the better tree at each rung by roughly 0.02–0.05. The trees draw level only at the very top of the ladder, and only on the coarse labels: at Max, LightGBM reaches 0.898 against TabICLv2's 0.890, a 0.008 gap of just over one standard error, so the two are effectively tied. On the fine 27-class labels (Table 2, right) TabICLv2 leads at *every* rung including Max (0.904 vs. 0.892–0.897) with much smaller relative standard errors. Splitting attempted from successful attacks makes the small contexts harder for every model but does not change the ordering, and even widens the separation. TabICLv2 is the stronger learner across the whole data-size axis, not just inside a small window.

Table 2: Macro-F_1 (mean over ten replicate seeds, cross-seed standard error in parentheses) at every context, coarse 16-class and fine 27-class side by side. TabICLv2 leads at every context on both label sets except the coarse Max rung, where LightGBM edges it by 0.008, just over a standard error. Trees are freshly tuned at every context, past 300k included. Best in each (context, label set) in bold.

	coarse (16-class)			fine (27-class)		
context	TabICLv2	LightGBM	XGBoost	TabICLv2	LightGBM	XGBoost
5k	**0.805** (.008)	0.724 (.058)	0.774 (.010)	**0.747** (.011)	0.723 (.008)	0.730 (.006)
10k	**0.820** (.006)	0.787 (.006)	0.775 (.004)	**0.805** (.008)	0.770 (.009)	0.771 (.006)
20k	**0.821** (.005)	0.810 (.007)	0.792 (.014)	**0.820** (.010)	0.781 (.012)	0.788 (.006)
50k	**0.828** (.007)	0.798 (.009)	0.783 (.011)	**0.819** (.007)	0.783 (.010)	0.795 (.006)
100k	**0.851** (.007)	0.813 (.011)	0.817 (.008)	**0.862** (.005)	0.800 (.005)	0.807 (.009)
150k	**0.848** (.011)	0.823 (.007)	0.815 (.011)	**0.865** (.004)	0.800 (.012)	0.805 (.007)
200k	**0.856** (.008)	0.839 (.008)	0.825 (.010)	**0.861** (.007)	0.808 (.013)	0.802 (.010)
300k	**0.867** (.006)	0.852 (.008)	0.842 (.011)	**0.872** (.007)	0.823 (.011)	0.820 (.010)
500k	**0.873** (.005)	0.852 (.009)	0.840 (.012)	**0.894** (.003)	0.857 (.007)	0.858 (.004)
1M	**0.892** (.007)	0.887 (.005)	0.876 (.009)	**0.902** (.003)	0.881 (.005)	0.881 (.005)
Max	0.890 (.006)	**0.898** (.007)	0.887 (.008)	**0.904** (.003)	0.897 (.002)	0.892 (.004)

The single place the trees pull ahead, LightGBM by 0.008 at Max on the coarse labels, need not be a plain data-scaling loss. One suspicion is that TabICLv2's QASS attention does not stay sharp as the context grows into the millions, even when the resulting decision is sharp [6]. A second, practical angle: increasing the number of shuffled member evaluations carries no downside beyond inference time and yields steady if small gains (Appendix B), so that knob alone could plausibly close a gap this noisy.

The aggregate hides where the gap actually is. Broken out by class on the rarest attacks (Table 3), TabICLv2's lead is concentrated almost entirely on the smallest classes. On Heartbleed (11 rows in the whole dataset) it reaches 0.93 F_1 against 0.73 and 0.63 for the trees, and on SQL injection (18 rows) 0.41 against 0.10–0.15, more than double. These are the classes a booster has almost nothing to split on, and where a zero-shot model pretrained on many small synthetic tasks has the most to offer. The pattern is not uniform: it underperforms the trees on Web XSS, a larger but still rare class, falling to 0.15 while the trees hold near 0.40, a lone failure whose cause is a specific sibling confusion rather than a benign miss (Section 5.3). On balance the foundation model's macro-F_1 edge is a rare-class edge, and mostly an ultra-rare-class one.

Table 3: Rare-class F_1 (ten-seed mean) on the coarse 16-class holdout at the 300k context. The macro-F_1 lead is concentrated on the smallest classes; Web XSS is the exception. Best in each row in bold.

rare class	nat. rows	TabICLv2	LightGBM	XGBoost
Heartbleed	11	**0.93**	0.73	0.63
SQL injection	18	**0.41**	0.10	0.15
Infiltration	81	**0.80**	**0.80**	0.69
Web XSS	673	0.15	**0.41**	0.39
Web brute force	1365	0.73	0.73	**0.74**

5.2 H2: error diversity and ensembling

H2 asks whether the two families are complementary and, if so, whether combining them pays. Complementarity is measured by the correlation and Yule's Q of the models' errors and the mutual information between their predictions, split by class frequency; whether it pays is tested with several combiners (a logistic stacker, an optimized weighted average, a confidence router, and an oracle ceiling) on rare-class average precision, with a paired Wilcoxon across seeds for the headline question: does adding TabICLv2 to a tree beat adding a second tree.

The diversity sits exactly where H2 expects it (Table 4). On the common classes every model is near-saturated and their errors are strongly correlated, so an ensemble has nothing to recover. On the rare attacks it inverts: TabICLv2's errors decorrelate from each tree's ($r \approx 0.4$, $Q \approx 0.7$) while the two boosters stay correlated with *each other* ($r \approx 0.6$, $Q \approx 0.9$). So the diversity is specific to the in-context learner, not a generic two-model effect, and the mutual information agrees, crediting TabICLv2 with a little rare-class predictive content neither tree carries. It narrows toward Max as all three converge on the now-easier tail, but does not close.

Whether that diversity pays depends entirely on the combiner, and the answer is a qualified yes. The complementarity is bidirectional (TabICLv2 rescues the smallest classes the trees miss, and the trees rescue the one class it collapses, Web XSS), so no single fixed blend follows the wins, and a cross-validated logistic stacker, the natural first choice, underperforms: while global F_1 may marginally gain, its rare-class F_1 collapses to 0.19 against TabICLv2's 0.60 because, without case weights for the loss, re-optimizing on the entire dataset prioritizes benign majority and only averages disagreements

Table 4: Error-indicator correlation between model pairs across the coarse context rungs (ten-seed means), split into common classes (natural count ≥ 2000) and the five rare classes. On the rare classes the TabICLv2–tree pairs decorrelate while the tree–tree pair stays correlated, so the in-context model errs on different rare rows than the boosters. The bottom three rows are across-context means of the correlation (r), Yule's Q [18], and normalized mutual information (NMI); all three agree the tree–tree pair is the least diverse on the rare tail.

	common classes			rare classes		
context	tab–xgb	tab–lgb	xgb–lgb	tab–xgb	tab–lgb	xgb–lgb
5k	0.67	0.61	0.76	0.59	0.54	0.65
10k	0.63	0.64	0.82	0.50	0.46	0.59
20k	0.66	0.66	0.80	0.58	0.55	0.65
50k	0.69	0.65	0.81	0.40	0.41	0.60
100k	0.64	0.61	0.77	0.39	0.39	0.63
150k	0.63	0.58	0.79	0.39	0.38	0.65
200k	0.75	0.68	0.79	0.38	0.37	0.61
300k	0.71	0.68	0.85	0.38	0.35	0.60
mean r	0.67	0.64	0.80	0.45	0.43	0.62
mean Q	1.00	0.99	1.00	0.72	0.71	0.89
mean NMI	0.96	0.96	0.97	0.41	0.39	0.46

on rare classes. A rare-aware combiner does better: either an optimized weighted average or a logistic stacker trained to favor the rare classes beats the *best single model* on rare-class average precision (Table 5), significantly over the tree–tree blend through 500k (paired Wilcoxon, $p \leq 0.05$) though the lead largely disappears when the trees are freshly tuned on the full partition. These rare-aware blends set their weight in-sample, so they mark what the diversity makes possible rather than a turnkey result, and a deployable combiner that captures it on the rare tail without in-sample tuning is still open.

Adding a TabICLv2 to a tree is worth more than a second tree exactly where context is the binding constraint, and a naive 50/50 ensemble average is the opposite, dragging TabICLv2 down with the weaker tree. However, the benefit is more difficult to distill the more the models converge to similar performance and become saturated with the context at their performance ceiling.

Table 5: Rare-class average precision (AUPR, ten-seed mean, coarse 16-class): the best single model, the optimized weighted TabICLv2+XGBoost blend, and the XGBoost+LightGBM blend. Both blend columns set their weight in-sample on the evaluation rows, so they are an achievable-in-principle ceiling; the comparison between them is still fair (identical tuning), and the deployable cross-validated stacker does not reach them on the rare tail. The last column is the paired Wilcoxon p-value for the TabICLv2+tree blend beating the tree+tree blend across the ten seeds. Higher is better; the cross-family blend wins significantly through 500k, and at 1M and Max the freshly retuned trees close the gap (the standalone retuned LightGBM has the best rare-AUPR at Max). Best in each row in bold.

context	best single	weighted tab+xgb	tree+tree	Wilcoxon p
5k	0.542	**0.570**	0.453	0.020
100k	0.692	**0.698**	0.568	0.002
300k	0.756	**0.761**	0.695	0.002
500k	0.747	**0.761**	0.698	0.049
1M	0.775	**0.791**	0.761	0.193
Max	**0.808**	0.806	0.790	0.695

5.3 H3: calibration on rare attacks

Calibration is read two ways a single number blurs, and both land on the rare attacks (Figure 4): the multiclass Brier score, a proper scoring rule that needs no binning, and the expected calibration error, taken on the rare-attack rows because the per-class one-vs-rest ECE collapses into a near-zero-confidence bin that understates rare miscalibration. The benign-dominated aggregate is a separate contest, noted below but not where H3 is decided.

On the rare attacks TabICLv2's probabilities are the most trustworthy at every context, on both metrics (Figure 4): its rare-row Brier and its rare-row ECE are lowest throughout, the gap widest where context is scarce and narrowing, without closing, as the trees are handed the whole partition; rare-class log-loss tells the same story. On the common rows the three coincide (left panel, dashed). Because the context is sampled at the natural prior the prior-shift correction is nearly a no-op here (rare Brier moves by ≤ 0.007 for any model), so the gap is real, not an artifact of the constructed prior.

The one place the ordering flips is the benign-heavy aggregate: from about 100k the trees overtake TabICLv2 on all-class ECE and pull several-fold ahead by Max, driven by the two large port-scan classes, which enough data lets them calibrate almost perfectly while TabICLv2 stays overconfident there. It is a calibration gap, not an accuracy one (all three classify those flows at $F_1 \approx 0.95$), and a common-class one, which is why H3 is stated on the tail rather than the aggregate.

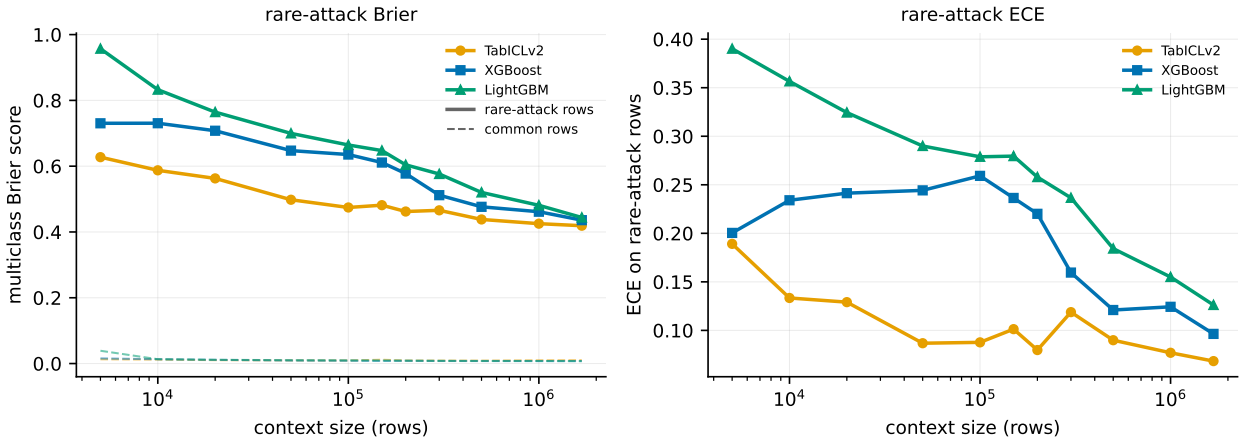

Figure 4: Rare-attack calibration vs. context (coarse 16-class, ten-seed mean, out to the full ∼1.68M-row Max partition). **Left:** multiclass Brier on rare-attack rows (solid) and common rows (dashed) per model. **Right:** ECE on the rare-attack rows. On both metrics TabICLv2 is best at every context, the gap largest at small and mid context and narrowing, without closing, by Max; on the common rows (left, dashed) the three coincide near zero.

The edge comes from how each family handles confusable classes (Appendix A). On the overlapping web attacks the trees turn overconfident, committing to predicted-XSS rows that are only about 40% correct, while TabICLv2 hedges. That caution is also why it loses Web XSS (Section 5.1): it folds the ambiguous rows into the larger sibling rather than overcommitting to them. The same hedging that costs it one class earns its calibration on the rest.

Post-hoc recalibration, whether temperature scaling [13], Platt scaling [14], or beta calibration [15], rescales a model's probabilities with a handful of shared parameters and cannot help a model that is overconfident on some classes and not others (Appendix A). As shown in H2, ensembling can: blending a TabICLv2 with a tree cancels the opposing biases and beats either alone on rare-class Brier from about 300k up. Averaging more of TabICLv2's internal members sharpens its calibration

monotonically at linear cost: all-class ECE keeps falling to sixteen members, about a quarter below the eight-member default and several times below a single member (Appendix B). The rare-class ranking saturates by eight, but the probabilities keep tightening, so more calibration is reachable through compute rather than more data.

5.4 Cost

TabICLv2's fit is essentially free (the context is stored, not trained on), but its inference time for the marginal row grows super-linearly with context size, which is what the fixed mega-capped holdout (Section 3.6) exists to bound. That super-linear growth, not memory, is why the ladder through 300k is scored on the A10G and the 500k, 1M, and Max points on the larger Blackwell card, which both fits the largest contexts in memory and keeps runtime realistic. The tuned boosters fit in seconds; their cost sits in the tuning budget instead (a single Optuna search of thirty trials at four-fold cross-validation per configuration, per seed). That search is cheap at small context but grows on the extended rungs, where LightGBM's CPU tuning dominates: a fresh search with cross validation at the full partition runs a few hours per cell.

Fundamentally, having hyperparameters to tune at all increases the risk of non-optimal fitting (over- or under-fitting), adds variance to model performance through how well each parameter search happens to land, and adds the operational burden of deciding when model artifacts should be updated on new data. A foundation model like TabICLv2 sidesteps much of this, both in terms of performance and for the use case where models must be updated frequently as new data arrives.

6 Conclusions

What is a tabular foundation model good for on real-world data? Does its competitive edge on curated benchmarks survive heavy imbalance, dozens of classes, and the need for trustworthy calibration? This study concludes that, for one such task along these lines, the answer is *yes*.

On performance, the zero-shot model is the stronger contender across almost the whole ladder, and in every case where the gap exceeded the seed-to-seed error bars. Given enough compute to scale the context to all available data, the foundation model does not degrade with attention softening and continues to earn its lead.

On error diversity, the two families make consistently different mistakes where it matters, and with the right combiner that can make a meaningful difference. On the common classes all three models are near-perfect and their errors correlated, so an ensemble has next to nothing to recover, but on the rare classes the correlation between different architectures drops well below the boosters. Further research outside of this study's scope will be required to determine optimal model stacking techniques to meet different analytical goals, but tabular foundation models appear to have genuinely different error modes from tree-based models.

On calibration, the foundation model's rare-attack probabilities are the most trustworthy at every context, on both a proper scoring rule and expected calibration error: it hedges on the confusable classes where the trees turn overconfident, an edge no post-hoc rescaling recovers and only ensembling matches. Reading this fairly required constructing the context and correcting the induced prior shift, and the raw-versus-corrected gap that correction exposes doubles as a check on whether a model's rare-class calibration is earned or an artifact of the sampling, which belongs in any such foundation-model-versus-tree comparison.

Next steps. The above conclusions rest on one dataset and one model, which sets the agenda for further exploration at the margins:

- More datasets, and a leakage-hardened temporal (capture-day) split, to test how far the three conclusions carry
- Newer and larger tabular foundation models such as TabPFN-2.5 and ContextTab, to see whether the rare-class and calibration edges hold or grow, learning what behavior can be credited to that tabular foundation model family and what might be unique to TabICLv2
- A rigorous analysis of ensembling a tabular foundation model with tree-based models across datasets, to find reliable ways to combine their complementary strengths
- the inference cost, and whether any edge captured can be distilled or cached into something fast enough for high volume real-time detection on par with boosted trees.

References

[1] Sharafaldin, I., Lashkari, A. H., & Ghorbani, A. A. (2018). Toward Generating a New Intrusion Detection Dataset and Intrusion Traffic Characterization. *ICISSP*.

[2] Engelen, G., Rimmer, V., & Joosen, W. (2021). Troubleshooting an Intrusion Detection Dataset: the CICIDS2017 Case Study. *IEEE Security and Privacy Workshops (SPW)*.

[3] Hollmann, N., Müller, S., Eggensperger, K., & Hutter, F. (2023). TabPFN: A Transformer That Solves Small Tabular Classification Problems in a Second. *ICLR*.

[4] Hollmann, N., Müller, S., et al. (2025). Accurate Predictions on Small Data with a Tabular Foundation Model. *Nature*, 637.

[5] Qu, J., Holzmüller, D., Varoquaux, G., & Le Morvan, M. (2025). TabICL: A Tabular Foundation Model for In-Context Learning on Large Data. *ICML*.

[6] Qu, J., Holzmüller, D., Varoquaux, G., & Le Morvan, M. (2026). TabICLv2: A Better, Faster, Scalable, and Open Tabular Foundation Model. *arXiv:2602.11139*.

[7] Grinsztajn, L., Oyallon, E., & Varoquaux, G. (2022). Why Do Tree-Based Models Still Outperform Deep Learning on Typical Tabular Data? *NeurIPS Datasets and Benchmarks*.

[8] Chen, T., & Guestrin, C. (2016). XGBoost: A Scalable Tree Boosting System. *KDD*.

[9] Ke, G., et al. (2017). LightGBM: A Highly Efficient Gradient Boosting Decision Tree. *NeurIPS*.

[10] Prokhorenkova, L., et al. (2018). CatBoost: Unbiased Boosting with Categorical Features. *NeurIPS*.

[11] Breiman, L. (2001). Random Forests. *Machine Learning*, 45(1).

[12] Akiba, T., et al. (2019). Optuna: A Next-Generation Hyperparameter Optimization Framework. *KDD*.

[13] Guo, C., Pleiss, G., Sun, Y., & Weinberger, K. Q. (2017). On Calibration of Modern Neural Networks. *ICML*.

[14] Platt, J. C. (1999). Probabilistic Outputs for Support Vector Machines and Comparisons to Regularized Likelihood Methods. *Advances in Large Margin Classifiers*.

[15] Kull, M., Silva Filho, T. M., & Flach, P. (2017). Beta Calibration: a Well-Founded and Easily Implemented Improvement on Logistic Calibration for Binary Classifiers. *AISTATS*.

[16] Naeini, M. P., Cooper, G. F., & Hauskrecht, M. (2015). Obtaining Well Calibrated Probabilities Using Bayesian Binning. *AAAI*.

[17] Dietterich, T. G. (1998). Approximate Statistical Tests for Comparing Supervised Classification Learning Algorithms. *Neural Computation*, 10(7).

[18] Kuncheva, L. I., & Whitaker, C. J. (2003). Measures of Diversity in Classifier Ensembles and Their Relationship with the Ensemble Accuracy. *Machine Learning*, 51(2).

[19] Saerens, M., Latinne, P., & Decaestecker, C. (2002). Adjusting the Outputs of a Classifier to New a Priori Probabilities: A Simple Procedure. *Neural Computation*, 14(1).

[20] Elkan, C. (2001). The Foundations of Cost-Sensitive Learning. *IJCAI*.

[21] Lipton, Z. C., Wang, Y.-X., & Smola, A. (2018). Detecting and Correcting for Label Shift with Black Box Predictors. *ICML*.

A Calibration direction (H3)

Because per-class ECE is near zero on the rare classes (Section 5.3), the direction of the miscalibration is of particular interest. Table 6 gives mean confidence minus precision on the rows each model commits to a rare class: positive is overconfident. The ultra-rare classes pull every model under-confident (the safe direction), while the confusable web attacks pull the trees sharply overconfident and leave TabICLv2 near zero, which is the reason for its durable rare-class ECE lead.

Table 6: Calibration *direction* on the rows each model commits to a rare class (coarse 16-class, 50k context, ten-seed mean, $\times 100$): mean confidence minus precision, so positive is overconfident and negative under-confident.

rare class	nat. rows	TabICLv2	XGBoost	LightGBM
Heartbleed	11	-15	-28	-16
SQL injection	18	-7	$+14$	$+43$
Infiltration	81	-0	-35	-18
Web XSS	673	$+7$	$+35$	$+39$
Web brute force	1365	-0	$+14$	$+18$

B TabICLv2 member count

TabICLv2 averages over an internal ensemble of members; the four smallest contexts store all sixteen. Table 7 sweeps the count from one to sixteen, bootstrapping the average over random member subsets at each count (so the values do not depend on which particular members the ordering happens to put first). Aggregate macro-F_1 and the rare-class ranking (AUPR) saturate by eight, so eight is a sound headline default. What keeps improving all the way to sixteen is calibration: all-class ECE falls monotonically, about a quarter below the eight-member value and several times below a single member, because averaging more members reduces the variance of the probability estimate. The per-seed spread is wide, but the decline is monotone at every step, so the extra members buy sharper probabilities at linear cost well beyond the package default of 8.

Table 7: Bootstrapped effect of the number of internal TabICLv2 members (coarse 16-class, ten-seed mean over ctx 5k–50k, averaged over random member subsets at each count). Aggregate macro-F_1 saturates by eight; all-class ECE keeps falling to sixteen while rare-class ranking is flat past eight.

members	macro-F_1	all-ECE ($\times 10^{-3}$)	rare-F_1	rare-AUPR
1	0.809	4.4	0.463	0.542
2	0.815	3.1	0.476	0.555
4	0.818	2.1	0.482	0.562
8	0.820	1.6	0.486	0.567
12	0.821	1.4	0.486	0.568
16	0.822	1.3	0.490	0.569

C Timing

TabICLv2's cost is inference and it dominates the wall-clock; the trees' cost is the Optuna search and it is minutes. Table 8 gives per-cell wall-clock on the coarse 16-class ladder, keeping the two GPUs as separate categories: the initial 5 seeds from the first sweep ran on a 24 GB cloud A10G, and the extension and second sweep of 5 additional seed ran on a larger Blackwell card that was a shared cluster node. Because that node was occasionally contended, the TabICLv2 figures are the per-group minimum (the floor is near-constant across seeds and contention only adds outliers above it); the tree figures are seed means, which were tight. Inference is reported per held-out row (about 99.9k rows) so it reads independent of the evaluation-set size.

TabICLv2 inference is about four times faster on the RTX PRO 6000 Blackwell card (about 97 GB) and grows super-linearly with context. The 5k–50k rows (marked *) were run at sixteen ensemble members to bank the member study, so their inference is roughly twice what an eight-member baseline would be; the $\geq$100k rows and above were ran using only eight members. For a fixed member size, per-row predict time rises steadily with context, reaching 0.8 ms at 100k, 3.6 ms at 300k, and 65 ms at the full partition, which is the wall-clock ceiling the main-body ladder is built around. The trees invert the cost: inference is single-digit microseconds per row, orders of magnitude below TabICLv2, while the expense sits in tuning, minutes at small context but growing to hours for LightGBM's CPU search at the extended rungs. XGBoost tunes on the GPU and LightGBM on the CPU, and that CPU search at 500k and up ran capped at 32 threads on the Blackwell box, whereas the $\leq$300k figures come from a different machine at 64 threads, so the LightGBM tune column is not directly comparable across the 300k boundary.

Table 8: Per-cell wall-clock on the coarse 16-class ladder (about 99.9k held-out rows). TabICLv2 inference is the per-group minimum on each GPU; through 300k, tree tune+fit is the A10G seed mean and inference the seed-mean per-row cost. A10G did not run 500k, 1M, or Max; the extended-rung tree figures (500k and up) are the Blackwell-box's CPU seed means as all tree training changes machine after 300k; the LightGBM CPU tuning is the few-hours-per-cell cost of Section 5.4. *TabICLv2 inference at 5k–50k was run at sixteen ensemble members to bank the member study (Appendix B); the eight-member headline cost would be roughly half.

	TabICLv2 inference			XGBoost		LightGBM	
context	A10G (s)	BW (s)	BW (ms/row)	tune+fit (s)	μs/row	tune+fit (s)	μs/row
5k*	86	24	0.24	131	1.3	129	3.5
10k*	106	29	0.29	155	1.4	162	3.5
20k*	145	40	0.40	160	1.2	224	4.7
50k*	296	78	0.79	197	1.4	378	5.7
100k	309	78	0.78	242	1.3	504	5.0
150k	536	130	1.31	294	1.4	667	6.5
200k	815	186	1.86	317	1.4	825	6.6
300k	1539	364	3.64	391	1.3	1059	7.0
500k	–	800	8.01	518	0.9	3702	4.6
1M	–	2548	25.51	1154	1.0	7983	6.0
Max	–	6513	65.22	1492	1.0	9848	5.9