arXiv:2610.17065v1 [cs.LG] 12 Oct 2026

TILE: BRIDGING LLMS AND TABLE FOUNDATION MODELS FOR TABULAR LEARNING WITH STRINGS

Félix Lefebvre[1] **David Holzmüller**[1] **Gioia Blayer**[1] **Myung Jun Kim**[2,*]
Marine Le Morvan[1] **Gaël Varoquaux**[1,3]
[1]SODA Team, Inria Saclay [2]SAP SE [3]Probabl [*]Work done while at Inria SODA

ABSTRACT

Real-world tables are often heterogeneous, combining numerical features with free-form string columns. Table foundation models (TFMs) excel on numerical tables but require strings to be encoded numerically, while large language models (LLMs) hold semantic knowledge but are not designed for tabular predictions. Most existing bridges use LLMs as plug-in encoders for cell strings, yet how to extract representations effectively for tabular learning remains underexplored. We show that the *interface* between LLMs and tables matter as much as the LLM itself. Applied cell-wise, larger LLMs do not give better features. With our interface, the same LLMs improve steadily with size. TILE (Table-Interfaced LLM Embeddings) combines four ingredients: *row serialization* to provide cross-column context, *intermediate-layer extraction* to avoid layers specialized for next-token prediction, *columnar pooling* to remove mean pooling's length bias, and *by-pass features* to route numbers and categories past the LLM. Paired with TFMs, TILE sets a new state of the art for tables with strings. Across 100 datasets, it beats the strongest baseline on 89% of splits, with gains across 7 LLMs and 10 tabular learners, and ranks first on the 108-dataset STRABLE benchmark.

1 INTRODUCTION: STRINGS, A NEW FRONTIER FOR TABULAR LEARNING

From businesses to hospitals, tables underpin critical decisions, making tabular learning—row-wise classification or regression—a cornerstone of applied machine learning (Clements et al., 2020; Pham

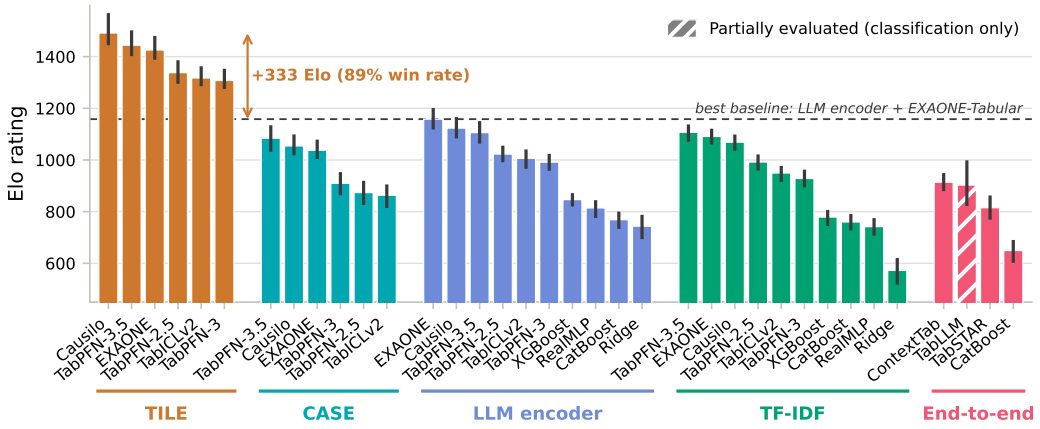


Figure 1: **TILE outperforms 30 baselines, with every table foundation model.** Elo ratings (Appendix B) over 100 regression and classification tasks, at 1,024 train size. TILE uses a Llama-3.1-8B backbone. *LLM encoder* applies the same LLM to each cell in isolation, then a PCA. TabLLM, an LLM fine-tuned to generate tabular predictions, is rated on the 34 datasets where it ran. CASE uses Gemma-3-4B, as its table-tuned backbone is not public (Appendix E).

et al., 2026; van Breugel & van der Schaar, 2024). Yet, modeling real-world tables is challenging because of the heterogeneity of their features. Numerical columns representing quantities sit beside string columns carrying semantics and world knowledge, two modalities historically handled by different learning algorithms (Grinsztajn et al., 2022; Shwartz-Ziv & Armon, 2022).

Table foundation models, particularly Prior-data Fitted Networks (PFNs) (Hollmann et al., 2023; 2025; Qu et al., 2025; 2026), now displace the industry-standard tree-based algorithms (Chen & Guestrin, 2016; Prokhorenkova et al., 2018) on numerical tables (Erickson et al., 2025). But they are built to model numbers. Standard pipelines therefore prepend a string encoder, such as TF-IDF, that reduces text to surface statistics, discarding both the world knowledge carried by the strings, and the relational context—column headers and neighbouring entries that give cell values their full meaning.

Conversely, LLMs internalize extensive knowledge about the entities mentioned in string cells (Petroni et al., 2019; Roberts et al., 2020; Bordt et al., 2024), but bridging them with tables is not straightforward. Naively applying LLMs to tables forces them to do numerical reasoning, at which they struggle (Mirzadeh et al., 2025; Baeumel et al., 2025), and requires flattening the structure that tabular algorithms exploit (Hegselmann et al., 2023; Gardner et al., 2024). Such end-to-end LLMs are in fact outperformed by character n-grams (Figure 1), showing that LLMs alone are not enough.

Existing bridges between LLMs and tables force a choice between context and structure. Row- and table-serialization methods (Hegselmann et al., 2023; Gardner et al., 2024; Schindler et al., 2025) provide context but flatten the table and typically return verbalized answers rather than representations for tabular learners. Cell-wise methods (Spinaci et al., 2025; Arazi et al., 2025; Kim et al., 2024; 2025) embed each string in isolation, which preserves the column structure but gives the LLM short, uncontextualized inputs. The cost of cell isolation becomes clear at scale. Applied cell by cell, larger LLMs do not give better features: across Llama3 models from 1B to 8B parameters the cell-wise encoder barely improves, and across Qwen3 the 0.6B model beats the 8B one (Figure 4). A lone cell such as "E.T." remains ambiguous, so the knowledge that grows with model size stays locked in. We show that better interfaced to tabular learning, the same LLMs improve steadily with size.

Contributions We call the *interface* between an LLM and a table the way the LLM fits into tabular learning: what the LLM is shown, which layer is read, how token embeddings are pooled, and which features are passed to the TFM. Prior work leaves it at its default: isolated cells in, mean-pooled last-layer vectors out. We argue that the interface matters as much as the LLM itself, and turn it into a design space. TILE makes four key choices, each addressing a limitation of current practices. **Row serialization** gives every cell its row-level context, addressing the *cell isolation* that leaves cell-wise encoders with strings too short to disambiguate (Section 3.1). **Intermediate-layer extraction** reads the LLM before the layers specialized for next-token prediction, which do not carry the most transferable semantics (Section 5.2). **Columnar pooling** restores the column structure that serialization flattens, and removes the *length bias* by which global mean pooling lets verbose columns drown out short informative ones (Section 3.1). Finally, **by-pass features** route numbers and exact cell identity around the LLM, correcting its numerical imprecision and the *representation drift* that gives one string different embeddings across rows (Section 3.2). Together, these choices turn LLM scale into predicitive gains: TILE's advantage over the same LLM applied cell-wise grows from 4.2 points with Qwen3-0.6B to 16.7 with Qwen3-8B (Section 5.1). Across 100 datasets, TILE beats 30 baselines, including prior LLM-based methods and the newest TFMs (Figure 1), and ranks first on the external STRABLE benchmark (Blayer et al., 2026), where TILE with Causilo (Cho et al., 2026) even beats an oracle that selects the best of 51 baselines on each dataset.

2 Prior work: existing bridges trade context against structure

From tree-based models to foundation models Long dominated by gradient-boosted decision trees (GBDTs) (Chen & Guestrin, 2016; Prokhorenkova et al., 2018) and their inductive biases for heterogeneous features (Grinsztajn et al., 2022), tabular learning has seen a resurgence of deep approaches (Ye et al., 2025; Holzmüller et al., 2024; Gorishniy et al., 2025) and, with them, table foundation models (Hollmann et al., 2025; Ma et al., 2025; Qu et al., 2025; Grinsztajn et al., 2025; Qu et al., 2026; Grinsztajn et al., 2026; Jäger et al., 2026; Eo et al., 2026; Cho et al., 2026; Zhang et al., 2026a; Tao et al., 2026) that beat GBDTs out-of-the-box (Erickson et al., 2025). Rather than fitting

one model per dataset, PFNs are pretrained on synthetic or real data and learn *in-context*, processing labelled training rows and unlabelled queries in one forward pass.

Traditional string handling in tabular pipelines Categorical variables are usually one-hot or ordinal encoded, and unconstrained strings vectorized by TF-IDF (Deerwester et al., 1990) or character n-grams (Cerda & Varoquaux, 2022), which remains purely surface-level. Studies explore the open problem of language model encodings for text in tables (Grinsztajn et al., 2023; Mráz et al., 2025; Lefebvre et al., 2025; Koloski et al., 2025; Kolomenko et al., 2025; Qiang et al., 2026).

Bridging language models and tabular learners Recent work passes either isolated cells or serialized rows to a language model. Cell-level approaches such as ContextTab (Spinaci et al., 2025), TabSTAR (Arazi et al., 2025), CARTE (Kim et al., 2024) and TARTE (Kim et al., 2025) rely on relatively small language models (Wang et al., 2020; 2022; Bojanowski et al., 2017), which lack the world knowledge of much larger models. Yet cell-wise content is short and uncontextualized, and we show that simply scaling the language model does not help it (Section 5.1). Row- and table-serialization methods like LIFT (Dinh et al., 2022), TabLLM (Hegselmann et al., 2023), TabuLa (Gardner et al., 2024), and others (Wen et al., 2024; Schindler et al., 2025) employ larger models but pay for them with per-dataset fine-tuning or the LLM context limit, and verbalize predictions rather than use a tabular learner. That cost buys little (Figure 1). TaRL (Kang et al., 2026) and Latte (Shi et al., 2025) use LLM representations for few-shot row classification, or latent knowledge transfer during training. The very recent CASE (Schindler et al., 2026) serializes full tables, primes the KV cache of a Tabular Language Model (a fine-tuned Gemma 3) on a sample of training rows, and appends its PCA-reduced last-layer embeddings to the original features. TILE similarly combines LLM representations with a tabular learner, but serializes rows, uses an off-the-shelf LLM, extracts intermediate-layer representations, and pools them column-wise.

3 TILE: INTEGRATING LLMS' REPRESENTATIONS WITH TFMS

TILE lets the LLM understand and contextualize the table, and the TFM learn from it (Figure 2; step by step in Appendix C). Its ingredients follow from two ways in which a table is not a text document.

Columns are heterogeneous, so one path cannot serve them all The split between numbers and strings is only the coarsest part of the heterogeneity, and string columns themselves come in kinds. Blayer et al. (2026) report that across 108 real-world tables about half of the string columns are plain categoricals, a quarter are entity names, a sixth are structured codes, and under a tenth is free text.

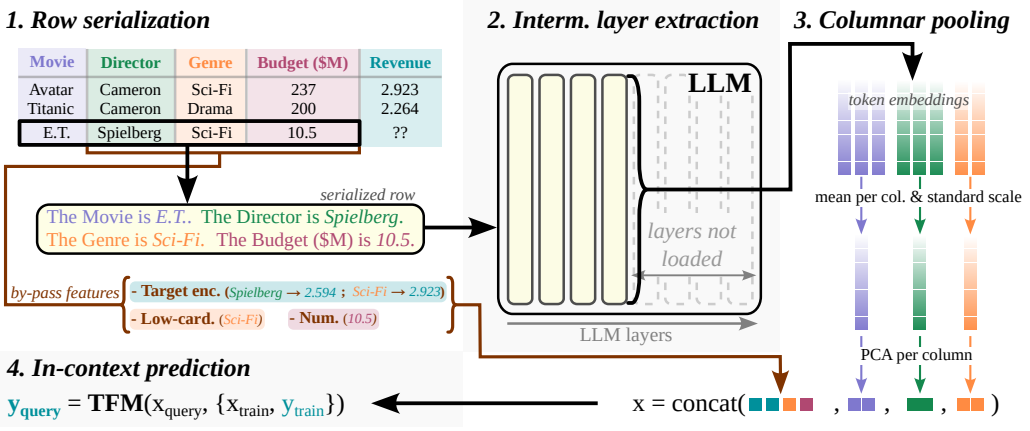


Figure 2: **The TILE architecture.** Each row is serialized into a sentence and fed to an LLM. Intermediate-layer token embeddings are pooled within string column spans; then standardized and reduced by one PCA per column, both fitted on training rows. The resulting contextualized representation is concatenated with low-cardinality categorical, numerical, and target-encoded categorical features. A TFM makes the final prediction in a single forward pass over the training and query rows.

These make opposite demands. A name such as "Spielberg" needs world knowledge, which only a language model supplies; a plain categorical needs exact identity and its association with the target, which an embedding blurs but a target encoding captures exactly; a number needs its value, which an LLM carries imprecisely. A single path for all columns would help some and hurt others; *by-pass features* (Section 3.2) and a cardinality threshold (Section F.3) route each column accordingly.

Structure repeats columnwise, so the representation must too A text sequence stands alone, whereas a table repeats the same structure across every row: column j carries the same meaning in row 1 and row 1000, and tabular learners exploit exactly that regularity (Grinsztajn et al., 2022). Serializing a row into a sentence discards this alignment, and pooling that sentence into a single vector discards it again. *Columnar pooling* (Section 3.1) restores it.

3.1 The LLM–table interface: context, depth, and pooling

Row serialization cures cell isolation Strings in tables are often very short, such as the few words of a movie title, and short strings are ambiguous: an LLM can resolve them only given context. "E.T." alone could be a time zone; next to "Spielberg" in a table about movies, it is the 1982 film (Figure 2). TILE therefore contextualizes each string within its row by row serialization (Hegselmann et al., 2023): each row becomes a sequence of blocks "The <column name> is <cell value>," so that the LLM's own attention conditions each cell on the column headers and on the rest of the row.

Intermediate-layer extraction avoids the output layer's specialization Contrary to prior work (Hegselmann et al., 2023; Gardner et al., 2024), we do not ask the LLM to verbalize an answer, but extract a representation of its input for a tabular learner. The last layer of a decoder LLM is optimized for next-token prediction, and the first layers still encode surface form; intermediate layers carry more transferable semantics than either end (Rogers et al., 2020; Skean et al., 2025). TILE reads one layer per LLM, chosen once (14 of 32 for Llama-3.1-8B; Section 5.2).

Columnar pooling removes the length bias of mean pooling Formally, let $\mathbf{x} = [x_1; \ldots; x_M]$ be a table row with M feature columns, serialized into L tokens $\mathbf{t} = (t_1, \ldots, t_L)$, and let $H \in \mathbb{R}^{L \times d}$ be the token embeddings h_i at the extraction layer. Standard mean pooling (Agarwal, 2026) computes $\frac{1}{L}\sum_{i=1}^{L} h_i$, which is biased toward columns with longer text and dilutes the signal of short but informative ones. Columnar pooling instead gives each string column its own vector. Writing $\mathcal{S}$ for the set of string columns and $\mathcal{I}_j$ for the indices of the tokens of column $j \in \mathcal{S}$ (header and value), the cell embedding is

$$c_j = \frac{1}{|\mathcal{I}_j|} \sum_{i \in \mathcal{I}_j} h_i. \tag{1}$$

Numerical columns are serialized, to contextualize the strings, but no embedding is read from their spans, since they reach the learner through the by-pass of Section 3.2. Each of the d dimensions is then standardized on the training rows, so that none dominates the PCA by scale alone, giving $\tilde{c}_j$.

One PCA per column keeps columns apart and fits the learner A cell embedding has far more dimensions than the feature counts TFMs are designed for (Tajjar et al., 2026). TILE fits one PCA P_j per column on the training rows, preserving the alignment that columnar pooling restored:

$$\mathbf{u} = \left[P_j(\tilde{c}_j)\right]_{j \in \mathcal{S}} \in \mathbb{R}^{k \cdot |\mathcal{S}|}. \tag{2}$$

TILE uses $k = 30$ components per column by default (Grinsztajn et al., 2023), moved into the window $[a, b]$ of feature counts the learner handles well—$[100, 300]$ for TabPFN-2.5, $[50, 300]$ for TabICLv2 and $[50, 250]$ for TabPFN-3 (Section F.2): $k = \max\left(1, \min(\lfloor b/|\mathcal{S}| \rfloor, \max(30, \lceil a/|\mathcal{S}| \rceil))\right)$. A narrow table thus gets more components per column, and a wide one fewer. Beyond accuracy improvements, this adaptive window also enables better memory scalability with respect to wide tables.

3.2 By-pass features address the blind spots of LLM embeddings

The projected representation $\mathbf{u}$ summarizes the row as the LLM reads it, with two blind spots: numbers are carried imprecisely, and *representation drift* makes a string's embedding depend on its row. These columns stay in the sentence, which is what contextualizes the strings, but TILE also routes them around the LLM as *by-pass features*, giving $\tilde{\mathbf{x}} = [\mathbf{u}; \mathbf{x}_{\text{num}}; \mathbf{x}_{\text{cat}}; \mathbf{x}_{\text{te}}]$.

The numerical by-pass preserves exact access to numbers Table foundation models excel at processing numerical features (Erickson et al., 2025), whereas language models handle numbers unreliably (Mirzadeh et al., 2025; Baeumel et al., 2025). Routing numerical columns only through the LLM would give the TFM a degraded version of the features it handles best. TILE instead also passes them directly to the TFM: the LLM contributes its understanding of the strings, informed by the numbers, while the TFM keeps exact access to the numbers themselves.

The categorical and target-encoding by-passes expose cell identity Row-level context makes TILE's representations informative, but it also causes representation drift: the same string receives different embeddings in different rows, which hides exact-value repetitions and their association with training targets. TILE therefore gives the TFM two more views of the string columns. String columns with sufficient repetition are target-encoded out of fold (Appendix D), and low-cardinality ones (below 50 distinct values; Section F.3) also enter as native categorical features.

4 Experimental study: TILE against 30 baselines on 100 tables

4.1 Evaluation protocol: 100 tables with strings

Experimental setup We evaluate TILE on 100 tables containing strings, drawn from TextTabBench (17), CARTE (51) and WikiDBs (32), with 65 regression and 35 classification tasks (Appendix A). For each dataset, we sample n_{train} training and 1,024 test samples (or all remaining ones), over 5 random seeds, and report standard errors of the mean, or 95% bootstrap intervals for Elo (Appendix B).

Metrics and score normalization We use R^2 (clipped at 0) for regression and ROC-AUC for classification. To aggregate across datasets of varying difficulty, we min-max normalize the scores per dataset and seed (Grinsztajn et al., 2022), over a fixed pool of 17 baselines (Appendix E) and the three default TILE variants, so that scores remain comparable across figures. Elo ratings (Figure 1) and raw scores (Appendix H) do not depend on this pool. TILE's few settings—the extraction layer, the feature-capacity window and the cardinality threshold—were set once on this benchmark and kept fixed; each sits on a broad plateau (Figure 5, Figure F.1, Figure F.2), and STRABLE (Section 4.3) validates them on separately assembled tables.

Baselines We compare TILE with table foundation models—TabPFN-2.5 (Grinsztajn et al., 2025), TabPFN-3 (Grinsztajn et al., 2026), TabPFN-3.5 (Jäger et al., 2026), TabICLv2 (Qu et al., 2026), EXAONE-Tabular (Eo et al., 2026) and Causilo (Cho et al., 2026)—and with tuned RealMLP (Holzmüller et al., 2024), tuned XGBoost (Chen & Guestrin, 2016), CatBoost (Prokhorenkova et al., 2018) and Ridge, each fed with one of two Skrub string encoders. *TF-IDF* (`StringEncoder`) represents each column with character n-gram TF-IDF features followed by SVD. The *LLM encoder* embeds each cell independently, mean-pools its last-layer representations, and reduces each column to 30 PCA components (Grinsztajn et al., 2023); for a fair comparison, it uses Llama-3.1-8B, TILE's default backbone. We also compare with methods that handle strings themselves: CatBoost, ContextTab (Spinaci et al., 2025), TabSTAR (Arazi et al., 2025), TabLLM (Hegselmann et al., 2023) and CASE (Schindler et al., 2026). TabPFN-3.5, EXAONE-Tabular and Causilo are not part of the normalization pool. Appendix E details all implementations.

4.2 TILE establishes a new state-of-the-art for tables with strings

TILE is the best performer across all training sizes Table 1 reports results at three training-set sizes, and Figure 1 summarizes the comparison at $n_{\text{train}} = 1{,}024$ with Elo ratings. Every TILE variant ranks above every baseline in all six columns, ahead of tuned gradient-boosted trees, CASE, and every cell-wise pipeline built on the same learners. At 1,024 training samples, the best baseline, LLM encoder + Causilo, scores 77.8 in regression and 81.0 in classification, against 91.6 and 90.4 for TILE (Causilo). In Elo ratings, TILE with Causilo leads the strongest baseline by 333 points, and beats it on 89% of splits (Appendix B). Raw differences are smaller in classification (89.7 against 88.6 AUC), where ROC-AUC saturates near 1 on many tasks (Appendix H). The regression gap shrinks with training-set size, from 13.8 points at 1,024 samples to 9.1 at 10,000 and 7.3 at 50,000 (over fewer datasets), likely because with more data learners rely less on external knowledge.

Table 1: **TILE leads at every train size, on both tasks.** Normalized scores $\times 100$, aggregated over the datasets large enough for each size; counts in parentheses. Because the set of datasets varies, compare scores within columns only. LLM enc. is Llama-3.1-8B applied to each cell independently; CASE uses Gemma-3-4B (Appendix E). Best result in bold, second best underlined. Table H.2 adds RealMLP, XGBoost, CatBoost and Ridge. Table H.1 gives raw scores.

	Regression (normalized R^2)			**Classification (normalized AUC)**		
Method	**1,024 (65)**	**10k (38)**	**50k (7)**	**1,024 (35)**	**10k (10)**	**50k (2)**
TILE (TabPFN-2.5)	90.21	93.19	92.58	88.00	87.62	79.64
TILE (TabICLv2)	88.70	92.32	92.00	88.94	85.93	80.17
TILE (TabPFN-3)	88.66	92.32	92.14	87.72	87.67	80.68
TILE (TabPFN-3.5)	<u>90.67</u>	<u>94.11</u>	<u>94.66</u>	<u>90.35</u>	**88.31**	**84.58**
TILE (Causilo)	**91.63**	**95.20**	**96.05**	**90.38**	<u>88.02</u>	<u>83.67</u>
CASE + TabPFN-2.5	61.97	64.96	49.11	67.50	77.13	35.14
CASE + TabICLv2	59.43	63.19	48.94	70.45	76.86	42.45
CASE + TabPFN-3	63.35	66.28	53.68	67.51	77.80	44.33
CASE + TabPFN-3.5	70.74	74.16	57.43	78.45	84.90	63.31
CASE + Causilo	71.88	77.00	68.16	76.52	80.83	59.23
ContextTab	70.78	78.83	66.68	69.96	80.46	49.36
TabSTAR	66.96	79.52	81.26	57.40	75.75	46.36
CatBoost	52.33	64.40	51.89	41.02	75.67	62.07
LLM enc. + TabPFN-2.5	72.97	83.20	85.26	79.48	83.24	52.16
LLM enc. + TabICLv2	68.82	83.61	85.67	80.59	83.46	66.50
LLM enc. + TabPFN-3	74.09	79.25	83.21	74.12	76.97	49.29
LLM enc. + TabPFN-3.5	76.36	86.03	85.81	80.65	83.51	60.38
LLM enc. + Causilo	77.84	85.10	88.73	80.99	83.34	63.75
TF-IDF + TabPFN-2.5	72.91	80.41	79.55	72.59	78.37	50.79
TF-IDF + TabICLv2	71.03	79.28	79.86	71.80	77.83	57.42
TF-IDF + TabPFN-3	71.61	78.73	81.86	67.36	75.14	49.37
TF-IDF + TabPFN-3.5	76.57	86.12	86.65	76.47	82.54	69.50
TF-IDF + Causilo	75.11	84.13	84.50	74.00	80.05	63.16

Without TILE, LLMs barely improve on TF-IDF Naive string encodings like TF-IDF, which ignore semantics, are a surprisingly strong baseline: character n-grams roughly match the large Llama-3.1-8B applied cell-wise, and beat TabLLM, which fine-tunes an LLM to predict by generating text, for 2.0 GB of memory against 17.1 (Section G.3). A TFM on plain TF-IDF likewise beats end-to-end ContextTab and TabSTAR. An LLM alone is not enough: its interface matters.

4.3 External validation: TILE ranks first on STRABLE

For a further evaluation varying datasets and protocol, we use STRABLE (Blayer et al., 2026), a separately assembled benchmark of 108 string-carrying tables and 445 string-tabular pipelines. Only 15 of its tables share a source dataset with our benchmark, and those are preprocessed differently (Section A.6). The four TILE variants take the first four places (Figure 3; raw scores in Table H.5), ahead of TF-IDF with Causilo and a cell-wise Qwen3-Embedding-8B fed to TabPFN-2.5. TILE with Causilo even beats picking the best baseline on each dataset. The ranking reproduces what we observe on our own datasets: with a TFM, TF-IDF and cell-wise LLMs perform comparably, and end-to-end models trail both. TILE also leads string representations absent from our benchmark (E5, Jasper, All-MiniLM, FastText), providing largely independent evidence that its gains generalize.

5 Ablations: the interface decides what an LLM contributes

5.1 Row context unlocks the benefits of larger LLMs

Scale pays off only through the right interface Figure 4 compares TILE with the cell-wise LLM encoder across four Llama (Grattafiori et al., 2024) and three Qwen3 models (Yang et al., 2025), spanning two orders of magnitude in size. The cell-wise encoder gains little with scale across Llama, and even declines across Qwen3, where the 0.6B model beats the 8B one. TILE improves

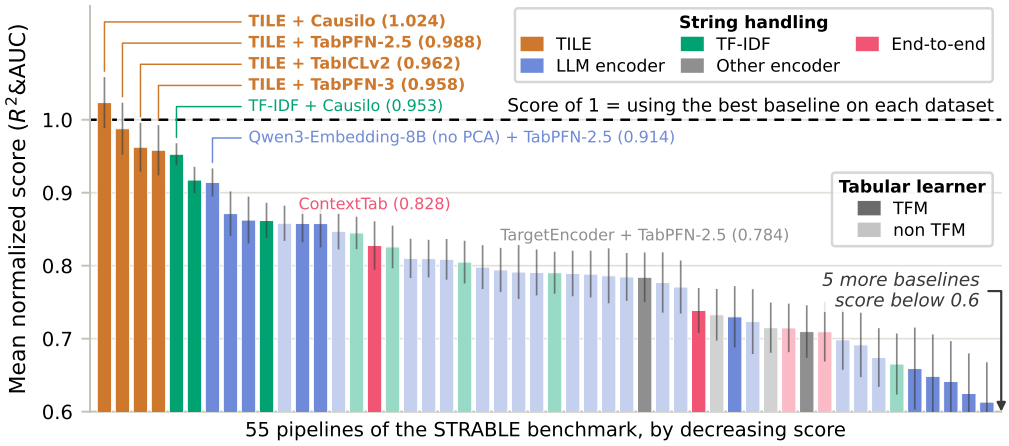


Figure 3: **TILE ranks first on STRABLE.** Mean normalized score over STRABLE's 108 datasets, under its evaluation protocol. Scores are normalized per dataset over 51 STRABLE baselines, TILE excluded, so a score of 1 amounts to picking the best baseline on each dataset. Colors indicate the string handling; transparent bars mark learners that are not table foundation models, trained from scratch. The four TILE variants take the first four places, and TILE with Causilo scores above 1.

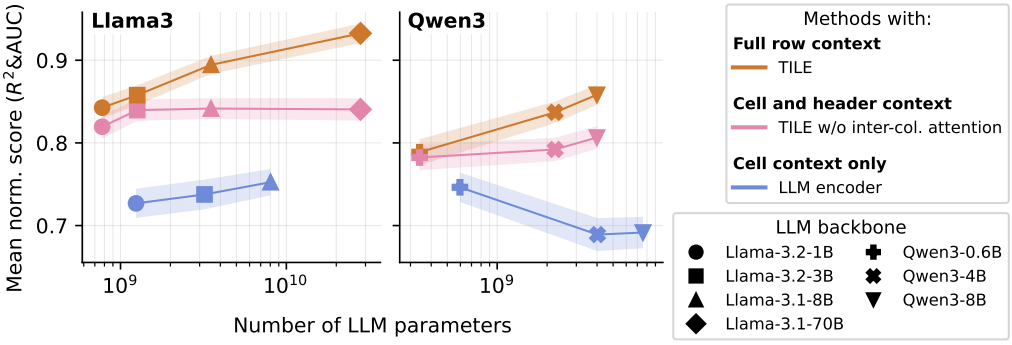


Figure 4: **Read cell by cell, larger LLMs do not help; with row-level context, they do.** TILE (brown), an ablation masking inter-column attention (pink), and the cell-wise LLM encoder (blue), across four Llama-3 and three Qwen3 backbones. The LLM encoder processes isolated cells, mean-pools last-layer representations, and applies per-column PCA. TILE extracts intermediate representations and thus loads fewer parameters. Table H.3 gives raw scores.

steadily with size in both families, so its advantage grows from 4.2 points for Qwen3-0.6B to 16.7 for Qwen3-8B. Both pipelines use the same LLMs, pretrained on the same data, and thus the gap reflects only the quality of the interface. Masking inter-column attention (pink) isolates the role of context, leaving each cell its header but blind to the rest of the row. That ablation still gains from scale, but less: larger models need the full row-level context to reach their potential.

TILE is versatile across LLMs TILE leads on all seven backbones. For Llama-3.1-70B, the cell-wise encoder does not fit in GPU memory, while truncated TILE does. TILE with Llama-3.1-8B loads about as many parameters as the full Llama-3.2-3B, yet scores 89.4 against 73.8 for the latter applied cell-wise (Section G.2, Section G.3 give runtime and memory).

5.2 Intermediate layers and columnar pooling make better representations

We select the extraction layer of each LLM once, on 27 probing datasets. To do so, we extract columnar-pooled representations at every layer, score them with Ridge, without PCA or by-pass

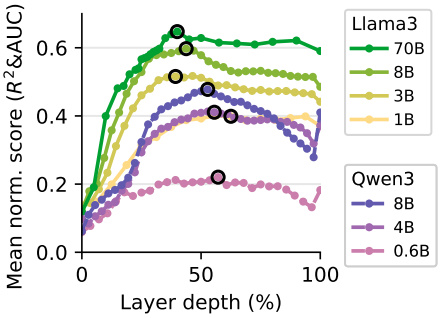


Figure 5: **Intermediate layers give the best representations, for all 7 backbones.** Columnar-pooled embeddings scored by Ridge (no PCA or by-pass features) across LLM depth, on 27 datasets (Section A.5). Circles mark TILE's extraction layer.

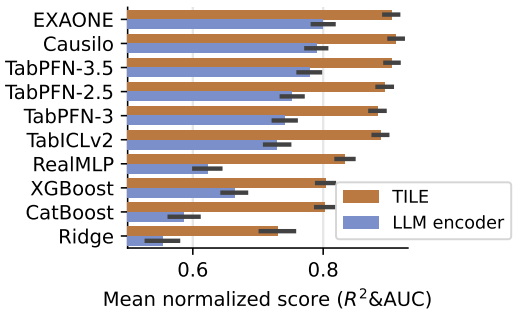


Figure 6: **TILE boosts every tabular learner.** With TILE, TFMs use their feature-capacity window (Section F.2); RealMLP, XGBoost and CatBoost use 30 components per column, as the LLM encoder does; Ridge takes the standard-scaled cell embeddings without PCA.

features, as a proxy for representation quality, and keep the best layer fixed across all datasets. A single forward pass gives all layers, so this sweep is inexpensive. For all 7 backbones, representation quality climbs steeply through the early layers, plateaus around the middle, and degrades towards the output (Figure 5). Every selected layer lies in the middle third of its network, consistent with Zhang et al. (2026b), and none is the last; for Llama-3.1-8B, it is layer 14 of 32. On the same probe, columnar pooling beats mean and last-token pooling at every layer (Section F.4).

5.3 TILE GENERALIZES ACROSS TABULAR LEARNERS

Replacing the table foundation models with alternative learners like RealMLP or Ridge lowers absolute performance (Figure 6), but with every learner, TILE's representations beat the LLM encoder's, by 10 to 22 points. The most recent TFMs, Causilo, TabPFN-3.5 and EXAONE-Tabular, score highest with TILE, showing that its gains add to those of each new generation of learners.

5.4 ROW CONTEXT AND THE NUMERICAL BY-PASS CARRY MOST OF THE GAIN

Table 2 measures the contribution of each component of TILE, and evaluates variations that were not retained. The rows above the default remove a component; those below add to or modify it; the bottom rows give the cell-wise LLM encoder. The three learners agree; we quote TabPFN-2.5.

The semantic and the by-pass paths compound Removing the numerical by-pass is the most damaging ablation, even though the LLM still reads the numbers. Conversely, omitting numerical values from the serialization also hurts, even though those columns reach the learner through the by-pass. Neither path substitutes for the other: the LLM needs the numbers to interpret the strings, and the learner needs them exactly for its predictions. Pooling embeddings of the numerical cells as well, however, does not help: within the TFM's feature-capacity window, they come at the cost of compressing the string embeddings further. On the semantic path, masking inter-column attention causes the second-largest drop, and reading the last layer instead of layer 14 costs 0.5 points in classification and 3.6 in regression, confirming the probing results of Section 5.2 on the full pipeline. Standardization, the feature-capacity window and the target-encoding by-pass each bring less than a point. None of these components alone explains the gap to the cell-wise LLM encoder, which shares TILE's numerical by-pass: masking inter-column attention, the largest, costs 3.0 (classification) and 6.5 (regression) points, against a gap of 8.5 and 17.3. Giving the LLM encoder TILE's other by-passes narrows the gap only to 7.7 and 15.0, and sizing its PCA with the feature-capacity window leaves it at 8.9 and 16.9 (both widen it with TabPFN-3).

TILE uses table-level context Replacing informative column headers with "Col 1", "Col 2", etc., or reversing the column order substantially degrades performance, showing that TILE exploits both

Table 2: **Ablation study.** Normalized score (×100) of the default TILE architecture, with Llama-3.1-8B and each of the three default learners, and the difference obtained when removing one of its components or adding a variation to it. The bottom rows give the cell-wise LLM encoder, alone, with the target-encoding and categorical by-passes, or with TILE's feature-capacity window. Normalization puts the regression and classification on a comparable scale; Table H.4 gives raw scores. Cell shading grows with the size of the effect (red: drop, blue: gain). The largest drops result from removing the numerical by-pass or inter-column attention. Columnar pooling cannot be removed without changing the shape of the representation; Section F.4 compares pooling strategies.

	TabPFN-2.5		TabICLv2		TabPFN-3	
Model	**clf**	**reg**	**clf**	**reg**	**clf**	**reg**
− numerical by-pass	-11.16	-14.19	-11.67	-13.58	-11.65	-14.59
− inter-column attention	-2.99	-6.52	-3.13	-8.92	-7.17	-9.05
− intermediate layer (last layer)	-0.52	-3.55	-1.33	-3.43	-0.63	-3.07
− numbers in the serialization	-1.43	-1.23	-1.70	-1.63	-1.52	-1.76
− feature-capacity window	-0.22	-0.72	-0.10	-0.22	+0.12	-0.41
− target-encoding by-pass	-0.22	-0.41	-0.67	-0.07	-0.94	-0.04
− standard scaling	-0.20	-0.17	-0.47	-0.14	-0.30	-0.37
− categorical by-pass	+0.24	-0.12	+0.03	-0.07	-0.03	-0.12
TILE (default)	**88.00**	**90.21**	**88.94**	**88.70**	**87.72**	**88.66**
+ numerical-cell embeddings	-0.18	-0.15	-0.14	+0.01	-0.30	-0.09
+ one-hot instead of native cat.	-0.43	-0.03	+0.13	-0.03	+0.13	-0.05
+ target name	-1.45	-0.62	-1.42	-0.63	-1.30	-0.92
+ repeated serialization	-1.26	-0.87	-1.46	-0.94	-1.57	-1.16
+ mirrored column order	-1.79	-1.38	-2.18	-1.86	-2.54	-2.05
+ uninformative column names	-2.05	-2.52	-2.41	-2.87	-2.55	-2.46
LLM encoder (`skrub`)	-8.52	-17.25	-8.35	-19.88	-13.60	-14.57
LLM encoder + by-pass features	-7.74	-14.95	-8.13	-18.22	-14.74	-16.18
LLM encoder + feature-capacity window	-8.91	-16.93	-8.44	-18.76	-16.87	-18.75

the column names and their arrangement. The effect of ordering is likely due to the causal mask of decoder LLMs, under which each cell attends only to preceding columns. As real tables often put identifying columns first, reversing them may deprive entries of informative context.

Richer prompts do not pay off Prepending the target column name ("What is the value of <target column name> given that:") is meant to inform the model of the task, but degrades performance. Repeating the serialized row twice and reading the embeddings from the second copy, inspired by prompt repetition in LLMs (Springer et al., 2025; Leviathan et al., 2025), lets every column attend to the whole row despite the causal mask; it hurts as well, and doubles the input length.

6 Discussion: Unlocking LLM World Knowledge for Tables

Building the right interface between LLMs and tables Pairing an LLM with a strong tabular learner, TILE turns strings into predictive signal, with sizable gains over the state of the art (Figure 1, Table 1). Our results suggest that what has held back LLMs on tables is their interface to the table. With row-level context and a well-chosen layer, TILE makes LLMs' gains grow with their size (Figure 4). TILE is versatile and improves every LLM backbone (Figure 4) and tabular learner (Figure 6) we evaluate, and gains from advances in both, as the newest TFMs show.

Limitations and future work TILE targets tables whose strings carry predictive signal; on purely numerical tables, a standalone TFM suffices. Its LLM embedding stage needs a GPU, though truncation nearly halves its memory (Section G.3). Row serialization is bounded by the LLM's context window; this was never a constraint in our benchmark, whose longest serialized row is 3,425 tokens against the 128k-token context of Llama-3.1-8B, but very wide tables with long texts would need chunking. Finally, the causal mask of decoder LLMs makes representations depend on column order, which bidirectional encoders (BehnamGhader et al., 2024) would avoid: a natural next step.

AI USE STATEMENT

In this work, we used generative AI tools as general-purpose assistants for writing and coding. For writing, they were used to polish prose and improve the clarity of text drafted by the authors; all scientific claims, experimental results and their interpretation were written and verified by the authors. For coding, they were used to assist with implementation and with the scripts that produce the figures and tables; all such code was read, tested and validated by the authors against the experimental protocol described in Section 4 and Appendix E. We did not use generative AI tools for research ideation, for the design of the method, or for the literature review, and they are not part of the TILE pipeline beyond the publicly released language models that it reads representations from (Appendix I). We have reviewed all AI-assisted work and take responsibility for the final content of this work, including its text, claims and artifacts.

ETHICS STATEMENT

By relying on LLMs, TILE may internalize and propagate the societal biases present in their pretraining data (Bender et al., 2021; Hu et al., 2025). In sensitive applications such as hiring, lending, or criminal justice, these encoded biases could produce unfair predictions or exacerbate existing demographic disparities, making pre-deployment audits crucial. Conversely, by improving predictive performance on heterogeneous tables, TILE could positively impact critical tabular-driven domains, such as healthcare, where patient records frequently intermix clinical measurements with free-text notes or diagnoses (Johnson et al., 2016; Miotto et al., 2016).

ACKNOWLEDGMENTS

This work was granted access to the HPC resources of IDRIS under the allocation 2025-AD011016808 made by GENCI. GV acknowledges support from ANR via grant TaFoMo (ANR-25-CE23-1822). This work is partly supported by Hi! PARIS and ANR/France 2030 program (ANR-23-IACL-0005).

REFERENCES

Ayushi Agarwal. PoolBench: A benchmark for pooling strategies in concept representation evaluation for decoder-only LLMs. *arXiv preprint arXiv:2608.05162*, 2026.

Alan Arazi, Eilam Shapira, and Roi Reichart. TabSTAR: A tabular foundation model for tabular data with text fields. In *Advances in Neural Information Processing Systems*, volume 38, pp. 190695–190748, 2025.

Tanja Baeumel, Josef Van Genabith, and Simon Ostermann. The lookahead limitation: Why multi-operand addition is hard for LLMs. In *Proceedings of the 8th BlackboxNLP Workshop: Analyzing and Interpreting Neural Networks for NLP*, pp. 250–262, 2025. doi: 10.18653/v1/2025.blackboxnlp-1.15.

Parishad BehnamGhader, Vaibhav Adlakha, Marius Mosbach, Dzmitry Bahdanau, Nicolas Chapados, and Siva Reddy. LLM2Vec: Large language models are secretly powerful text encoders. In *Conference on Language Modeling (COLM)*, 2024. URL https://openreview.net/forum?id=IW1PR7vEBf.

Emily M. Bender, Timnit Gebru, Angelina McMillan-Major, and Shmargaret Shmitchell. On the dangers of stochastic parrots: Can language models be too big? In *Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency*, pp. 610–623, 2021. doi: 10.1145/3442188.3445922.

Gioia Blayer, Myung Jun Kim, Félix Lefebvre, Lennart Purucker, Alan Arazi, Eilam Shapira, Roi Reichart, Frank Hutter, Marine Le Morvan, David Holzmüller, and Gaël Varoquaux. STRABLE: Benchmarking tabular machine learning with strings. *arXiv preprint arXiv:2605.12292*, 2026.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146, 2017. doi: 10.1162/tacl_a_00051.

Sebastian Bordt, Harsha Nori, Vanessa Rodrigues, Besmira Nushi, and Rich Caruana. Elephants never forget: Memorization and learning of tabular data in large language models. In *Conference on Language Modeling (COLM)*, 2024. URL https://openreview.net/forum?id=HLoWN6m4fS.

Ralph Allan Bradley and Milton E. Terry. Rank analysis of incomplete block designs: I. The method of paired comparisons. *Biometrika*, 39(3/4):324–345, 1952. doi: 10.2307/2334029.

Patricio Cerda and Gaël Varoquaux. Encoding high-cardinality string categorical variables. *IEEE Transactions on Knowledge and Data Engineering*, 34(3):1164–1176, 2022.

Tianqi Chen and Carlos Guestrin. XGBoost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 785–794, 2016. doi: 10.1145/2939672.2939785.

Wei-Lin Chiang, Lianmin Zheng, Ying Sheng, Anastasios Nikolas Angelopoulos, Tianle Li, Dacheng Li, Banghua Zhu, Hao Zhang, Michael Jordan, Joseph E. Gonzalez, and Ion Stoica. Chatbot arena: An open platform for evaluating LLMs by human preference. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 8359–8388. PMLR, 2024. URL https://openreview.net/forum?id=3MW8GKNyzI.

Minyong Cho, Minho Jeong, Dooho Lee, Jinmo Lee, and Jaemin Yoo. Causilo technical report. *arXiv preprint arXiv:2609.22866*, 2026.

Jillian M Clements, Di Xu, Nooshin Yousefi, and Dmitry Efimov. Sequential deep learning for credit risk monitoring with tabular financial data. *arXiv preprint arXiv:2012.15330*, 2020.

Scott Deerwester, Susan T Dumais, George W Furnas, Thomas K Landauer, and Richard Harshman. Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, 41 (6):391–407, 1990.

Tuan Dinh, Yuchen Zeng, Ruisu Zhang, Ziqian Lin, Michael Gira, Shashank Rajput, Jy-yong Sohn, Dimitris Papailiopoulos, and Kangwook Lee. Lift: Language-interfaced fine-tuning for non-language machine learning tasks. *Advances in Neural Information Processing Systems*, 35: 11763–11784, 2022.

Moonjung Eo, Min-Kook Suh, Hye-Seung Cho, Jiwon Kim, Seoyoon Kim, Sangjun Nam, and Soonyoung Lee. EXAONE tabular 1.0: Technical report. *arXiv preprint arXiv:2608.25774*, 2026.

Nick Erickson, Lennart Purucker, Andrej Tschalzev, David Holzmüller, Prateek Mutalik Desai, David Salinas, and Frank Hutter. TabArena: A living benchmark for machine learning on tabular data. In *Advances in Neural Information Processing Systems (NeurIPS), Datasets and Benchmarks Track*, 2025. URL https://arxiv.org/abs/2506.16791.

Josh Gardner, Juan C Perdomo, and Ludwig Schmidt. Large scale transfer learning for tabular data via language modeling. *Advances in Neural Information Processing Systems*, 37:45155–45205, 2024.

Yury Gorishniy, Akim Kotelnikov, and Artem Babenko. TabM: Advancing tabular deep learning with parameter-efficient ensembling. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=Sd4wYYOhmY.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. The Llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Léo Grinsztajn, Edouard Oyallon, and Gaël Varoquaux. Why do tree-based models still outperform deep learning on typical tabular data? In *Advances in Neural Information Processing Systems (NeurIPS), Datasets and Benchmarks Track*, volume 35, pp. 507–520, 2022.

Léo Grinsztajn, Edouard Oyallon, Myung Jun Kim, and Gaël Varoquaux. Vectorizing string entries for data processing on tables: when are larger language models better? *arXiv preprint arXiv:2312.09634*, 2023.

Léo Grinsztajn, Klemens Flöge, Oscar Key, Felix Birkel, Philipp Jund, Brendan Roof, Benjamin Jäger, Dominik Safaric, Simone Alessi, Adrian Hayler, et al. TabPFN-2.5: Advancing the state of the art in tabular foundation models. *arXiv preprint arXiv:2511.08667*, 2025.

Léo Grinsztajn, Klemens Flöge, Oscar Key, Felix Birkel, Philipp Jund, Brendan Roof, Mihir Manium, Shi Bin Hoo, Magnus Bühler, Anurag Garg, et al. TabPFN-3: Technical report. *arXiv preprint arXiv:2605.13986*, 2026.

Stefan Hegselmann, Alejandro Buendia, Hunter Lang, Monica Agrawal, Xiaoyi Jiang, and David Sontag. TabLLM: Few-shot classification of tabular data with large language models. In *Proceedings of the 26th International Conference on Artificial Intelligence and Statistics (AISTATS)*, volume 206 of *Proceedings of Machine Learning Research*, pp. 5549–5581. PMLR, 2023.

Noah Hollmann, Samuel Müller, Katharina Eggensperger, and Frank Hutter. TabPFN: A transformer that solves small tabular classification problems in a second. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=cp5PvcI6w8_.

Noah Hollmann, Samuel Müller, Lennart Purucker, Arjun Krishnakumar, Max Körfer, Shi Bin Hoo, Robin Tibor Schirrmeister, and Frank Hutter. Accurate predictions on small data with a tabular foundation model. *Nature*, 637(8045):319–326, 2025.

David Holzmüller, Léo Grinsztajn, and Ingo Steinwart. Better by default: Strong pre-tuned MLPs and boosted trees on tabular data. *Advances in Neural Information Processing Systems*, 37: 26577–26658, 2024.

Tiancheng Hu, Yara Kyrychenko, Steve Rathje, Nigel Collier, Sander van der Linden, and Jon Roozenbeek. Generative language models exhibit social identity biases. *Nature Computational Science*, 5(1):65–75, 2025. doi: 10.1038/s43588-024-00741-1.

Benjamin Jäger, Nick Erickson, Léo Grinsztajn, Felix Birkel, Klemens Flöge, Oscar Key, Kürşat Kaya, Jonas Kübler, Adèle Frankel, Tobias Schröder, et al. TabPFN-3.5: Technical report. *arXiv preprint arXiv:2609.17895*, 2026.

Alistair EW Johnson, Tom J Pollard, Lu Shen, Li-wei H Lehman, Mengling Feng, Mohammad Ghassemi, Benjamin Moody, Peter Szolovits, Leo Anthony Celi, and Roger G Mark. MIMIC-III, a freely accessible critical care database. *Scientific Data*, 3(1):160035, 2016. doi: 10.1038/sdata.2016.35.

Inwon Kang, Parikshit Ram, Yi Zhou, Horst Samulowitz, and Oshani Seneviratne. Language model representations for efficient few-shot tabular classification. In *Proceedings of the ACM Web Conference 2026*, pp. 4069–4080, 2026.

Myung Jun Kim, Leo Grinsztajn, and Gael Varoquaux. CARTE: Pretraining and transfer for tabular learning. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 23843–23866. PMLR, 2024.

Myung Jun Kim, Félix Lefebvre, Gaëtan Brison, Alexandre Perez-Lebel, and Gaël Varoquaux. Table foundation models: On knowledge pre-training for tabular learning. *Transactions on Machine Learning Research*, 2025. URL https://openreview.net/forum?id=QV4P8Csw17.

Oksana Kolomenko, Ricardo Knauer, and Erik Rodner. Embedding world knowledge into tabular models: Towards best practices for embedding pipeline design. In Horst Schulte, Frank Hoffmann, and Ralf Mikut (eds.), *Proceedings – 35. Workshop Computational Intelligence, Berlin, 20.–21. November 2025*, pp. 103–120. Karlsruher Institut für Technologie (KIT), 2025. doi: 10.5445/IR/1000186052. arXiv:2603.17737.

Boshko Koloski, Andrei Margeloiu, Xiangjian Jiang, Blaž Škrlj, Nikola Simidjievski, and Mateja Jamnik. Llm embeddings for deep learning on tabular data. *arXiv preprint arXiv:2502.11596*, 2025.

Félix Lefebvre, Myung Jun Kim, and Gaël Varoquaux. Knowledge-rich embeddings for tabular learning. In *AITD 2025-Workshop on AI for Tabular Data*, 2025. URL https://openreview.net/forum?id=sNli26jLOy.

Yaniv Leviathan, Matan Kalman, and Yossi Matias. Prompt repetition improves non-reasoning LLMs. *arXiv preprint arXiv:2512.14982*, 2025.

Haokun Liu, Derek Tam, Mohammed Muqeeth, Jay Mohta, Tenghao Huang, Mohit Bansal, and Colin Raffel. Few-shot parameter-efficient fine-tuning is better and cheaper than in-context learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 35, pp. 1950–1965, 2022.

Nelson F Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. Lost in the middle: How language models use long contexts. *Transactions of the Association for Computational Linguistics*, 12:157–173, 2024.

Junwei Ma, Valentin Thomas, Rasa Hosseinzadeh, Alex Labach, Jesse C Cresswell, Keyvan Golestan, Guangwei Yu, Anthony L Caterini, and Maksims Volkovs. TabDPT: Scaling tabular foundation models on real data. In *Advances in Neural Information Processing Systems*, volume 38, pp. 191387–191417, 2025. doi: 10.52202/085713-5748.

Daniele Micci-Barreca. A preprocessing scheme for high-cardinality categorical attributes in classification and prediction problems. *ACM SIGKDD Explorations Newsletter*, 3(1):27–32, 2001. doi: 10.1145/507533.507538.

Riccardo Miotto, Li Li, Brian A. Kidd, and Joel T. Dudley. Deep patient: An unsupervised representation to predict the future of patients from the electronic health records. *Scientific Reports*, 6(1): 26094, 2016. doi: 10.1038/srep26094.

Seyed Iman Mirzadeh, Keivan Alizadeh, Hooman Shahrokhi, Oncel Tuzel, Samy Bengio, and Mehrdad Farajtabar. GSM-symbolic: Understanding the limitations of mathematical reasoning in large language models. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=AjXkRZIvjB.

Martin Mráz, Breenda Das, Anshul Gupta, Lennart Purucker, and Frank Hutter. Towards benchmarking foundation models for tabular data with text. In *1st ICML Workshop on Foundation Models for Structured Data*, 2025. arXiv:2507.07829.

Fabio Petroni, Tim Rocktäschel, Sebastian Riedel, Patrick Lewis, Anton Bakhtin, Yuxiang Wu, and Alexander Miller. Language models as knowledge bases? In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pp. 2463–2473, 2019. doi: 10.18653/v1/D19-1250.

Minh-Khoi Pham, Thang-Long Nguyen Ho, Thao Thi Phuong Dao, Tai Tan Mai, Minh-Triet Tran, Marie E. Ward, Una Geary, Rob Brennan, Nick McDonald, Martin Crane, and Marija Bezbradica. Retrieval-aligned tabular foundation models enable robust clinical risk prediction in electronic health records under real-world constraints. *arXiv preprint arXiv:2604.01841*, 2026.

Liudmila Prokhorenkova, Gleb Gusev, Aleksandr Vorobev, Anna Veronika Dorogush, and Andrey Gulin. CatBoost: unbiased boosting with categorical features. *Advances in neural information processing systems*, 31, 2018.

Minjie Qiang, Mingming Zhang, Xiaoyi Bao, Xing Fu, Yu Cheng, Weiqiang Wang, Zhongqing Wang, and Ningtao Wang. TabEmbed: Benchmarking and learning generalist embeddings for tabular understanding. *arXiv preprint arXiv:2605.04962*, 2026.

Jingang Qu, David Holzmüller, Gaël Varoquaux, and Marine Le Morvan. TabICL: A tabular foundation model for in-context learning on large data. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 50817–50847. PMLR, 2025.

Jingang Qu, David Holzmüller, Gaël Varoquaux, and Marine Le Morvan. TabICLv2: A better, faster, scalable, and open tabular foundation model. In *Proceedings of the 43rd International Conference on Machine Learning (ICML)*, 2026. URL https://arxiv.org/abs/2602.11139.

Adam Roberts, Colin Raffel, and Noam Shazeer. How much knowledge can you pack into the parameters of a language model? In *Proceedings of the 2020 conference on empirical methods in natural language processing (EMNLP)*, pp. 5418–5426, 2020.

Anna Rogers, Olga Kovaleva, and Anna Rumshisky. A primer in BERTology: What we know about how BERT works. *Transactions of the Association for Computational Linguistics*, 8:842–866, 2020. doi: 10.1162/tacl_a_00349.

Victor Sanh, Albert Webson, Colin Raffel, Stephen Bach, Lintang Sutawika, Zaid Alyafeai, Antoine Chaffin, Arnaud Stiegler, Arun Raja, Manan Dey, M Saiful Bari, Canwen Xu, Urmish Thakker, Shanya Sharma Sharma, Eliza Szczechla, Taewoon Kim, Gunjan Chhablani, Nihal Nayak, Debajyoti Datta, Jonathan Chang, Mike Tian-Jian Jiang, Han Wang, Matteo Manica, Sheng Shen, Zheng Xin Yong, Harshit Pandey, Rachel Bawden, Thomas Wang, Trishala Neeraj, Jos Rozen, Abheesht Sharma, Andrea Santilli, Thibault Fevry, Jason Alan Fries, Ryan Teehan, Teven Le Scao, Stella Biderman, Leo Gao, Thomas Wolf, and Alexander M Rush. Multitask prompted training enables zero-shot task generalization. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=9Vrb9D0WI4.

Günther Schindler, Maximilian Schambach, Michael Medek, and Sam Thelin. TabGemma: Text-based tabular ICL via LLM using continued pretraining and retrieval. *arXiv preprint arXiv:2511.03570*, 2025.

Günther Schindler, Maximilian Schambach, and Johannes Höhne. Enhancing tabular learners with context-aware semantic embeddings. In *2nd ICML Workshop on Foundation Models for Structured Data*, 2026. URL https://openreview.net/forum?id=QArxQg4U71. arXiv:2608.03565.

Ruxue Shi, Hengrui Gu, Hangting Ye, Yiwei Dai, Xu Shen, and Xin Wang. Latte: Transfering llms' latent-level knowledge for few-shot tabular learning. In James Kwok (ed.), *Proceedings of the Thirty-Fourth International Joint Conference on Artificial Intelligence, IJCAI-25*, pp. 6173–6181. International Joint Conferences on Artificial Intelligence Organization, 8 2025. doi: 10.24963/ijcai.2025/687. URL https://doi.org/10.24963/ijcai.2025/687. Main Track.

Ravid Shwartz-Ziv and Amitai Armon. Tabular data: Deep learning is not all you need. *Information Fusion*, 81:84–90, 2022.

Oscar Skean, Md Rifat Arefin, Dan Zhao, Niket Nikul Patel, Jalal Naghiyev, Yann LeCun, and Ravid Shwartz-Ziv. Layer by layer: Uncovering hidden representations in language models. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 55854–55875. PMLR, 2025.

Marco Spinaci, Marek Polewczyk, Maximilian Schambach, and Sam Thelin. ConTextTab: A semantics-aware tabular in-context learner. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025.

Jacob Mitchell Springer, Suhas Kotha, Daniel Fried, Graham Neubig, and Aditi Raghunathan. Repetition improves language model embeddings. In *International Conference on Learning Representations*, 2025.

Mustafa Tajjar, Alexander Pfefferle, Lennart Purucker, and Frank Hutter. Towards pretraining text encoders for TabPFN. *arXiv preprint arXiv:2606.04876*, 2026.

Yefan Tao, Xiyuan Zhang, Xinyi Liu, et al. Mitra-v2 technical report. *arXiv preprint arXiv:2609.04540*, 2026.

Anton Frederik Thielmann, Manish Kumar, Christoph Weisser, Arik Reuter, Benjamin Säfken, and Soheila Samiee. Mambular: A sequential model for tabular deep learning. *arXiv preprint arXiv:2408.06291*, 2024.

Boris van Breugel and Mihaela van der Schaar. Position: Why tabular foundation models should be a research priority. In *Proceedings of the 41st International Conference on Machine Learning (ICML)*, volume 235 of *Proceedings of Machine Learning Research*, pp. 48976–48993. PMLR, 2024.

Liane Vogel, Jan-Micha Bodensohn, and Carsten Binnig. WikiDBs: A large-scale corpus of relational databases from Wikidata. *Advances in Neural Information Processing Systems*, 37:41186–41201, 2024.

Liang Wang, Nan Yang, Xiaolong Huang, Binxing Jiao, Linjun Yang, Daxin Jiang, Rangan Majumder, and Furu Wei. Text embeddings by weakly-supervised contrastive pre-training. *arXiv preprint arXiv:2212.03533*, 2022.

Wenhui Wang, Furu Wei, Li Dong, Hangbo Bao, Nan Yang, and Ming Zhou. MiniLM: Deep self-attention distillation for task-agnostic compression of pre-trained transformers. In *Advances in Neural Information Processing Systems*, volume 33, pp. 5776–5788, 2020.

Yuanrui Wang, Xingxuan Zhang, Han Yu, Mingchao Hao, Gang Ren, Hao Yuan, Li Mao, Yunjia Zhang, Chun Yuan, and Peng Cui. LimiX-2M: Mitigating low-rank collapse and attention bottlenecks in tabular foundation models. In *Proceedings of the 43rd International Conference on Machine Learning (ICML)*, 2026. URL https://openreview.net/forum?id=YjYPYg112A.

Xumeng Wen, Han Zhang, Shun Zheng, Wei Xu, and Jiang Bian. From supervised to generative: A novel paradigm for tabular deep learning with large language models. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 3323–3333, 2024.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

Han-Jia Ye, Huai-Hong Yin, De-Chuan Zhan, and Wei-Lun Chao. Revisiting nearest neighbor for tabular data: A deep tabular baseline two decades later. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=JytL2MrlLT.

Dun Zhang, Ziyang Zeng, Yudong Zhou, and Shuyang Lu. Jasper-token-compression-600m technical report. *arXiv preprint arXiv:2511.14405*, 2025.

Xingxuan Zhang, Gang Ren, Hao Yuan, et al. LimiX-2: A contextual mechanism network towards general structured-data intelligence. *arXiv preprint arXiv:2609.17488*, 2026a.

Yeqin Zhang, Yunfei Wang, Jiaxuan Chen, Ke Qin, Yizheng Zhao, and Cam-Tu Nguyen. LLM-based embeddings: Attention values encode sentence semantics better than hidden states. In *Proceedings of the 43rd International Conference on Machine Learning (ICML)*, 2026b. URL https://arxiv.org/abs/2602.01572.

Appendices

Contents

A Evaluation datasets 17
A.1 Data sources . . . 17
A.2 Data preprocessing . . . 17
A.3 Data characteristics . . . 17
A.4 Excluded dataset . . . 17
A.5 Datasets used for LLM probing . . . 17
A.6 Overlap with STRABLE . . . 18
B Elo ratings 23
C The TILE pipeline, end to end 24
D Target encoding of the categorical string columns 25
E Baselines and their implementation 26
F Additional experimental results 28
F.1 The improvability metric . . . 28
F.2 The feature-capacity window of each learner . . . 28
F.3 Cardinality threshold for categorical columns . . . 28
F.4 Choosing the pooling strategy . . . 28
G Compute and memory costs 31
G.1 Hardware and total compute . . . 31
G.2 Runtime by task . . . 31
G.3 Peak GPU memory . . . 31
H Raw results 33
I Models, software and licenses 45

A Evaluation datasets

A.1 Data sources

To evaluate tabular learning with strings, we assemble a diverse benchmark of 100 tabular datasets from three sources with distinct characteristics and inclusion biases: TextTabBench (Mráz et al., 2025), CARTE (Kim et al., 2024), and WikiDBs (Vogel et al., 2024).

TextTabBench and CARTE are established benchmarks for tabular learning, providing real-world tables with varied text features, from short entity names to longer descriptions. Each table is associated with a predefined prediction task (regression, binary, or multi-class classification). WikiDBs is a large corpus of over 1.6 million semi-synthetic tables generated from Wikidata. To create meaningful tasks from this source, we first filtered for tables with at least 1,200 rows, then manually curated a subset of 32 tables for which we could define a relevant prediction problem. Table A.1 summarizes the final distribution of tasks across the three sources. Table A.2, Table A.3 and Table A.4 provide further details on each individual dataset.

Table A.1: Task distribution across sources.

Source	b-clf	m-clf	reg	Total
TextTabBench	5	2	10	**17**
CARTE	11	-	40	**51**
WikiDBs	1	16	15	**32**
Total	**17**	**18**	**65**	**100**

A.2 Data preprocessing

We adopt the original preprocessing from TextTabBench and CARTE. For WikiDBs, we apply a procedure similar to TextTabBench. We also ensure that multi-class classification tasks have at most 10 classes, each with at least 105 samples. For all 100 datasets, we then apply the following preprocessing pipeline: (1) we log-transform regression targets with wide-ranging distributions; (2) we downsample majority classes in multi-class problems to create balanced datasets; and (3) we discard any table with fewer than 1,050 rows post-processing to ensure sufficient data for evaluation. We also exclude one dataset from TextTabBench with excessively long text entries, as detailed in Section A.4.

A.3 Data characteristics

Table A.5 reports aggregated features across datasets, broken down by data source. In particular, it shows that TextTabBench tables tend to contain longer text fields while CARTE and WikiDBs tables gather shorter string entries.

A.4 Excluded dataset

The dataset excluded from our benchmark is the OkCupid table of TextTabBench, a multi-class classification task with 59,946 rows, 28 string columns and very long text entries. The exclusion is a constraint of the baselines, not of our method: the cell-wise LLM encoder runs out of memory on it, so no LLM-encoder pipeline ran on this dataset, and including it would leave holes in the comparison. TILE processes it without difficulty, as do the TF-IDF pipelines.

For completeness, Table A.6 reports the methods that do run on all 101 tables. Adding this dataset leaves the ranking unchanged. In absolute terms it is a hard task for every method, with raw ROC AUC scores ranging from 65.0 to 73.2, the best being TILE with Causilo.

A.5 Datasets used for LLM probing

The layer sweeps of Figure 5 and the pooling comparison of Figure F.3 use a subset of 27 datasets. The datasets used are the following:

- **CARTE datasets:** NBA Draft, Prescription Drugs, Coffee Ratings, Ramen Ratings, Chocolate Bar Ratings, Roger Ebert, UsedCars.com, Used Cars 24, Michelin, Beer Ratings, Company Employees, Journal JCR, Videogame Sales, Filmtv Movies, Wine Vivino Price, Zomato, Yelp.
- **TextTabBench datasets:** Beer, Mercari.
- **WikiDBs datasets:** Magic Narrative, Geopolitical Regions, Sculptures, Research Articles, Ukrainian villages, Business Locations, Territorial Entities, Drawings Catalog.

A.6 Overlap with STRABLE

STRABLE (Blayer et al., 2026) and our benchmark were assembled separately, but both draw on the small pool of public tables with meaningful text, so some sources appear in both. Of STRABLE's 108 tables, 15 (about 14%) share a source dataset with CARTE or TextTabBench; the remaining 93 are unique to STRABLE.

Shared sources Ten of the shared tables come from CARTE: AI/ML salaries, beer ratings, chocolate bar ratings, CLEAR corpus, SCImago journal ranking, Michelin, ramen ratings, whisky, Wine Enthusiast, and Yelp. A sixteenth match is possible if CARTE's Kaggle museum table, a repackaging of the IMLS Museum Universe file, is counted. Six come from TextTabBench: beer ratings (also in CARTE), CFPB complaints, Kickstarter, Mercari, OSHA accidents, and San Francisco building permits. Two of the TextTabBench-shared sources, OSHA and San Francisco permits, are paired with a different prediction target in STRABLE: injury severity and construction cost, rather than task regularity and approval time. Several nominal matches are distinct datasets: STRABLE's California Housing is the 1990 census table, not TextTabBench's text-rich listings, and TextTabBench's wine and Spotify tables have no STRABLE counterpart.

Shared tables are not identical copies STRABLE applies uniform minimal preprocessing, with no dataset-specific feature engineering. It keeps the most recent snapshot of each source and applies a skewness-minimizing target transform. CARTE, by contrast, applies hand-crafted per-dataset transformations, such as recoding whisky price brackets and binarizing its target.

The two benchmarks are largely, though not fully, independent On STRABLE's side, the overlap is small relative to its size and concentrated in its Food, Commerce, and Education domains. On ours, the shared sources account for 17 of the 100 tables, or 18 if the museum table is counted: 11 of the 51 CARTE tables, as Wine Enthusiast gives both a price and a rating table, and 6 of the 17 TextTabBench tables. This overlap reflects the scarcity of public tables with meaningful text rather than a derivative design. Results on STRABLE can thus be read as largely complementary evidence to those on our benchmark, with the caveat that a model's gains on the ~ 15 shared sources should not be counted as independent confirmation across benchmarks.

Table A.2: Overview of TextTabBench datasets used in our benchmark. Table statistics after preprocessing.

Dataset	**Task**	**# rows**	**# columns**		**# classes**
			cat.	*num.*	
Diabetes	b-clf	17,000	4	12	2
Job Frauds	b-clf	1,732	11	3	2
Kickstarter	b-clf	18,720	9	8	2
Lending Club	b-clf	11,254	12	15	2
Osha Accidents	b-clf	3,598	15	3	2
Customer Complaints	m-clf	1,384	8	2	4
Spotify	m-clf	10,000	3	15	10
Airbnb	reg	3,818	32	23	-
Beer	reg	2,914	5	15	-
California Houses	reg	11,349	13	16	-
Covid Trials	reg	1,165	13	2	-
Insurance Complaints	reg	37,484	8	2	-
IT Salary	reg	1,253	16	2	-
Mercari	reg	12,000	4	2	-
San Francisco Permits	reg	183,794	12	16	-
Stack Overflow	reg	19,427	89	13	-
Wine	reg	1,281	12	3	-

Table A.3: Overview of CARTE datasets used in our benchmark. Table statistics after preprocessing.

Dataset	**Task**	**# rows**	**# columns**		**# classes**
			cat.	*num.*	
Chocolate Bar Ratings	b-clf	2,218	6	5	2
Coffee Ratings	b-clf	1,670	8	-	2
Michelin	b-clf	6,774	5	2	2
NBA Draft	b-clf	1,550	4	5	2
Ramen Ratings	b-clf	3,726	4	-	2
Roger Ebert	b-clf	2,668	5	5	2
Spotify	b-clf	41,096	7	11	2
US Accidents Severity	b-clf	20,930	9	4	2
Whisky	b-clf	1,788	6	-	2
Yelp	b-clf	60,088	8	4	2
Zomato	b-clf	60,302	7	1	2
Movies	reg	7,224	7	9	-
US Accidents Counts	reg	22,623	6	-	-
US Presidential	reg	19,857	6	-	-
Anime Planet	reg	14,391	6	10	-
Babies R Us	reg	5,085	4	-	-
Beer Ratings	reg	3,197	5	14	-
Bikedekho	reg	4,786	5	5	-
Bikewale	reg	8,992	5	5	-
Buy Buy Baby	reg	10,718	4	-	-
Cardekho	reg	37,813	13	6	-
Clear Corpus	reg	4,724	10	19	-
Company Employees	reg	10,941	7	-	-
Employee Remuneration	reg	35,396	2	5	-
Employee Salaries	reg	9,211	6	8	-
Fifa22 Players	reg	18,085	9	18	-
Filmtv Movies	reg	41,205	6	6	-
Journal JCR	reg	9,615	4	5	-
Journal SJR	reg	27,931	9	-	-
Japanese Anime	reg	15,535	11	5	-
K-Drama	reg	1,239	8	4	-
ML/DS Salaries	reg	10,456	7	4	-
Museums	reg	11,467	14	2	-
Mydramalist	reg	3,400	10	3	-
Prescription Drugs	reg	1,714	5	5	-
Rotten Tomatoes	reg	7,158	10	6	-
Used Cars 24	reg	5,918	6	5	-
Used Cars Benz Italy	reg	16,391	5	2	-
UsedCars.com	reg	4,009	8	5	-
Used Cars Pakistan	reg	72,655	4	6	-
Used Cars Saudi Arabia	reg	5,507	7	6	-
Videogame Sales	reg	16,410	4	4	-
Wikiliq Beer	reg	13,461	7	2	-
Wikiliq Spirit	reg	12,275	5	2	-
Wina Poland	reg	2,247	12	6	-
Wine.com Prices	reg	15,254	6	3	-
Wine.com Ratings	reg	4,095	6	3	-
WineEnthusiasts Prices	reg	120,975	8	1	-
WineEnthusiasts Ratings	reg	129,971	8	1	-
WineVivino Price	reg	13,834	5	2	-
WineVivino Rating	reg	13,834	6	2	-

Table A.4: Overview of WikiDBs datasets used in our benchmark. Table statistics after preprocessing.

Dataset	**Task**	**# rows**	**# columns**		**# classes**
			cat.	*num.*	
CC Authors	b-clf	16,224	7	1	2
Defenders	m-clf	18,610	9	-	10
Philosophers	m-clf	4,230	7	-	10
US Music Albums	m-clf	3,270	10	1	10
Artist Copyrights	m-clf	2,000	9	1	10
Artworks Catalog	m-clf	1,210	7	2	10
Forward Players	m-clf	1,400	10	-	10
Geographers	m-clf	1,130	8	-	10
Island	m-clf	19,650	2	2	10
Magic Narratives	m-clf	1,062	4	-	9
Museums	m-clf	9,550	4	2	10
Noble Individuals	m-clf	1,400	9	-	10
Notable Trees	m-clf	1,408	4	2	8
Sculptures	m-clf	3,720	6	-	10
Spring Locations	m-clf	5,930	1	2	10
State Schools	m-clf	2,800	2	2	10
Transport Stations	m-clf	4,640	7	2	10
Business Locations	reg	16,821	4	4	-
Dissolved Municipalities	reg	13,462	6	2	-
Geopolitical Regions	reg	1,114	6	3	-
Historical Figures	reg	11,260	11	-	-
Municipal District Capitals	reg	1,658	5	3	-
Poets	reg	60,240	10	-	-
Territorial Entities	reg	36,717	7	4	-
WWI Personnel	reg	30,675	11	-	-
Artworks Inventory	reg	10,635	5	1	-
Drawings Catalog	reg	63,130	8	1	-
Eclipsing Binary Stars	reg	297,934	6	2	-
Registered Ships	reg	4,644	6	3	-
Research Articles	reg	6,962	6	2	-
Research Article Citations	reg	4,115	9	-	-
Ukrainian Villages	reg	21,355	3	3	-

Table A.5: **Aggregated features of tabular datasets across sources.** The cardinality is computed over 1,024 rows. “String similarity" represents cosine similarity of TF-IDF across rows.

	TextTabBench	**CARTE**	**WikiDBs**
# columns	15.7	6.8	6.7
cardinality	286.4	371.4	463.7
string length	975.3	298.8	203.6
string similarity	0.16	0.10	0.08

Table A.6: **Adding the excluded dataset.** Normalized scores over the excluded table alone, over the benchmark, and over their union. Normalization is over the 15 methods that ran on all 101 tables, a smaller pool than in the rest of the paper, so the ranking is what should be read.

Method	OkCupid (1)	Benchmark (100)	Both (101)
TILE (Causilo)	**83.2**	**87.6**	**87.6**
TILE (TabPFN-3.5)	82.6	86.5	86.5
TILE (TabPFN-2.5)	62.5	85.0	84.8
TILE (TabICLv2)	82.3	84.2	84.2
TILE (TabPFN-3)	68.7	83.6	83.5
TF-IDF + TabPFN-3.5	77.5	69.6	69.7
TF-IDF + EXAONE-Tabular	41.2	68.8	68.5
TF-IDF + Causilo	67.4	67.8	67.8
TF-IDF + TabPFN-2.5	64.5	64.7	64.7
TF-IDF + TabICLv2	74.6	63.2	63.3
ContextTab	75.0	61.0	61.2
TabSTAR	25.9	54.1	53.8
TF-IDF + XGBoost	46.3	53.3	53.2
TF-IDF + RealMLP	33.6	51.2	51.0
TF-IDF + Ridge	58.6	25.5	25.8

B ELO RATINGS

Figure 1 summarizes the benchmark with Elo ratings, computed with the implementation of TabArena (Erickson et al., 2025), which follows Chatbot Arena (Chiang et al., 2024).

Pairwise comparisons Each method m is evaluated on dataset d with 5 random train/test splits (seeds) s, the same for every method, giving an error $e_{m,d,s} = 1 - R^2$ for regression and $1 -$ ROC AUC for classification. The unit of comparison is a *battle*: for every dataset d, seed s and pair of methods (a, b) that were both run on (d, s), method a wins if $e_{a,d,s} < e_{b,d,s}$, loses if $e_{a,d,s} > e_{b,d,s}$, and a tie counts as half a win for each (ties occur in fewer than 0.01% of battles). Each battle has weight $1/5$, so that every dataset has the same total weight. Because a battle only compares two methods on the same split, regression and classification datasets can be pooled into a single rating without ever comparing R^2 with ROC AUC. Over the 36 methods and 100 datasets this gives 303,345 battles.

Rating model The ratings r_m are the maximum-likelihood estimate of a Bradley–Terry model (Bradley & Terry, 1952) on the Elo scale,

$$P(a \text{ beats } b) = \frac{1}{1 + 10^{-(r_a - r_b)/400}}, \tag{3}$$

fit by weighted logistic regression over all battles at once (with negligible regularization, $C = 10^6$). Unlike the online Elo update, this estimate uses no K-factor and does not depend on the order of the battles: it is the fixed point that online Elo approaches with a vanishing step size. Ratings are only defined up to a common shift, which we fix so that the mean rating over methods is 1000.

Interpretation By Equation 3, only differences between ratings matter. A gap of Δ points translates into an expected head-to-head win probability: $\Delta = 100$ gives 64%, $\Delta = 200$ gives 76% and $\Delta = 400$ gives 91%. The $+333$ Elo gap between TILE (Causilo) and the strongest baseline (LLM encoder + EXAONE-Tabular) thus means that TILE is expected to beat that baseline on 87.2% of splits. On the 500 splits where both were run, it actually wins on 88.6% (443 wins, no ties), and 88.6% as well when averaged per dataset. This strongest baseline does not appear in Table 1: because of compute constraints, we ran LLM encoder + EXAONE-Tabular only at $n_{\text{train}} = 1{,}024$, the training size of the Elo ratings and of the per-dataset results of Appendix H, and not yet at $10{,}000$ and $50{,}000$.

Confidence intervals Error bars are 95% percentile bootstrap intervals. We resample the 100 datasets with replacement (all seeds of a drawn dataset are kept together), refit the ratings on the battles of the resample, and repeat this 200 times (bootstrap random seed 0). The bars go from the 2.5% to the 97.5% quantile of the bootstrap ratings and are drawn around the point estimate fitted on the full benchmark, which is why they can be asymmetric. Resampling datasets rather than individual battles accounts for the correlation between battles on the same dataset, so the intervals reflect uncertainty over the choice of datasets. Per-method intervals do not directly give the uncertainty of a difference, so we bootstrap the gap itself with the same resamples: the gap between TILE (Causilo) and LLM encoder + EXAONE-Tabular has a 95% interval of $[272, 397]$ and is positive in all 200 resamples.

Partially evaluated methods TabLLM was run on 34 classification datasets only. Battles are formed only between methods run on the same split, so TabLLM is rated from its 5,845 battles on those datasets. Its wider interval reflects the smaller number of battles.

C THE TILE PIPELINE, END TO END

Putting the above together, TILE maps a row $\mathbf{x}$ to a prediction as follows; every statistic (standardization, PCA, target encoding) is fitted on the training rows only.

1. **Serialize.** $s(\mathbf{x})$ concatenates "The <column name> is <cell value>" over the M columns; tokenizing gives $\mathbf{t} \in \mathcal{V}^L$ and the token span of each column.
2. **Encode.** A forward pass through the LLM truncated at layer ℓ gives $H = \mathrm{LLM}_{1:\ell}(\mathbf{t}) \in \mathbb{R}^{L \times d}$.
3. **Pool by column.** Equation 1, then standardization, gives $\tilde{c}_j$ for every $j \in \mathcal{S}$.
4. **Project by column.** Equation 2 gives $\mathbf{u}$, with k set by the learner's feature-capacity window.
5. **Append the by-pass features.** $\tilde{\mathbf{x}} = [\mathbf{u}; \mathbf{x}_{\mathrm{num}}; \mathbf{x}_{\mathrm{cat}}; \mathbf{x}_{\mathrm{te}}]$, none of which passes through the PCA.
6. **Predict.** $\hat{y} = \mathrm{PFN}\big(\tilde{\mathbf{x}}_{\mathrm{query}} \mid \{(\tilde{\mathbf{x}}_i, y_i)\}_{i=1}^{n}\big)$, in one forward pass over the training and test rows.

D TARGET ENCODING OF THE CATEGORICAL STRING COLUMNS

Alongside its LLM embeddings, TILE passes every suitable string column to the tabular learner as a target encoding (Micci-Barreca, 2001). This section specifies that step; Section 3.2 motivates it.

Encoding For a string column taking the value c, classification with K target classes encodes c by the class proportions of the training rows carrying it, $\hat{p}_k(c) = n_{c,k}/n_c$: one column per target class, or the single $\hat{p}_1(c)$ for a binary target, since the proportions sum to one. Similarly, regression uses the category mean $n_c^{-1} \sum_{i:c_i=c} y_i$, and adds a single column. Missing entries form their own category.

Out-of-fold estimation Training rows are encoded from five folds so that a row never contributes to its own value (the rows of one fold use the statistics of the other four). Without that, a high-cardinality column would encode every singleton category to its own label, and the learner would read the target off the encoding. Test rows are encoded from the whole training split. Categories seen at test time but absent from the training rows fall back to the prior: the global target class proportions for classification or the global target mean $\bar{y}$ for regression.

Which columns are encoded A string column is target-encoded when more than 10% of its non-missing entries hold a value that occurs at least twice. Below that the column is almost entirely singletons and would encode to a near-constant.

Integration in the pipeline The encoded columns are appended as numerical features to the PCA-reduced embedding and to the table's own numerical and low-cardinality categorical columns, and are not themselves passed through the PCA. This path is independent of the categorical by-pass: a column with fewer than 50 distinct values also reaches the learner as a native categorical feature, whereas a high-cardinality column reaches it only through its embedding and its target encoding.

E BASELINES AND THEIR IMPLEMENTATION

We compare against three families of string handling. (i) Learners with internal string handling: CatBoost, ContextTab and TabSTAR. (ii) TF-IDF followed by an SVD, and (iii) the LLM encoder (Llama-3.1-8B applied to the cell content), each combined with seven tabular learners: Ridge, XGBoost, CatBoost, RealMLP, TabPFN-2.5, TabPFN-3 and TabICLv2. This gives 17 baselines, all of which ran on the full benchmark (65 regression and 35 classification datasets) with 5 seeds.

These 17 baselines, together with the three default TILE variants (using Llama-3.1-8B as LLM backbone), form the reference pool used to normalize the scores reported throughout the paper (Section 4.1). We additionally report TabLLM (Hegselmann et al., 2023), a fine-tuned LLM that predicts by generating text. Since it is a classifier and runs out of memory on one classification dataset, it covers only part of the benchmark and does not enter the reference pool. The concurrent CASE (Schindler et al., 2026) is reported outside the pool as well, and so are three recent table foundation models: TabPFN-3.5 (Jäger et al., 2026), EXAONE-Tabular (Eo et al., 2026) and Causilo (Cho et al., 2026).

Every baseline sees exactly the splits TILE sees: the same five seeds and the same training and test rows (Section 4.1), stratified for classification, and scored with the same metric code, ROC AUC (one-vs-rest, weighted) for classification and R^2 for regression.

Learners with internal string handling **CatBoost** (v1.2.10) is run on the raw table with the string columns declared as `cat_features`: columns are typed with a `skrub TableVectorizer` (`cardinality_threshold=0`, `high_cardinality="passthrough"`) so that no column is one-hot encoded and every non-numeric column is passed to CatBoost as a categorical feature, with missing entries mapped to the string `"nan"`. **TabSTAR** (Arazi et al., 2025) is the official `tabstar` package (v1.1.15) with its default pre-trained checkpoint; numeric columns are kept numeric, datetimes are decomposed with `skrub`'s `DatetimeEncoder`, and all remaining columns are cast to text so the model sees them through its text encoder. **ContextTab** (Spinaci et al., 2025) is the official SAP release `sap-rpt-oss` (v1.0.1) with default settings, on tables passed through `skrub`'s `Cleaner`.

TF-IDF string representations Every non-numeric column is embedded independently with `skrub`'s `StringEncoder` (v0.7.0), a character n-gram TF-IDF followed by a truncated SVD. The SVD width is 30 components per column, reduced to $\lfloor 500/n_{\text{str}} \rfloor$ when the table has more than 17 string columns, to keep the representation of wide tables tractable. The resulting matrix is concatenated with the numeric columns and handed to each of the seven tabular learners listed below.

LLM encoder This baseline is `skrub`'s `LLMEncoder` (formerly `TextEncoder`). Each string cell is embedded on its own by Llama-3.1-8B (`meta-llama/Llama-3.1-8B`) loaded through `sentence-transformers` (v5.2.0); embeddings are computed once per unique cell value and broadcast back to the table. Each column's 4096-dimensional block is then reduced to 30 dimensions by a column-wise PCA fitted on the training rows, concatenated across columns and with the numeric columns, and fed to the same seven learners. This baseline is the natural ablation of TILE: the same language model, but applied at full depth to isolated cells without inter-column attention.

Downstream tabular learners Both representation families are paired with **Ridge** (`scikit-learn` v1.7.2, `RidgeCV`/`RidgeClassifierCV` with $\alpha \in \{10^{-4}, \ldots, 10^{4}\}$, numeric columns mean-imputed and power-transformed), **XGBoost** (Chen & Guestrin, 2016) (v3.2.0), **CatBoost** (Prokhorenkova et al., 2018) (v1.2.10), **RealMLP** (Holzmüller et al., 2024) (`pytabkit` v1.7.3), **TabPFN-2.5** and **TabPFN-3** (Grinsztajn et al., 2025; 2026) (`tabpfn` v8.0.6, default checkpoint of each version, `ignore_pretraining_limits` enabled), and **TabICLv2** (Qu et al., 2026) (`tabicl` v2.1.1, default checkpoint). Unless stated otherwise, learners are run with their default hyper-parameters, which is the recommended usage of the table foundation models. Two exceptions: XGBoost is tuned by a 100-iteration random search over the grid of Table E.1, evaluated by 5-fold cross-validation repeated 5 times on the training rows; and RealMLP uses its own built-in hyper-parameter optimization (`RealMLP_HPO`, 50 search steps, internal 5-fold cross-validation, validation metric `1-auc_ovr` for classification).

Additional table foundation models We use **TabPFN-3.5** (Jäger et al., 2026) through `tabpfn` v9.0.0 with the `v3.5-20260909` checkpoint, **EXAONE-Tabular** (Eo et al., 2026) through `exaonetabular` v1.0.0 with the v1 default checkpoints (Hugging Face revision `093ad1c`), and **Causilo** (Cho et al., 2026) through `causilo` v1.0.2 with its release checkpoints (revision `94f2bd9`). All three run with their default inference settings (8 ensemble members for EXAONE-Tabular and Causilo) and no hyper-parameter tuning.

Table E.1: **Search space of the tuned XGBoost baseline.** Sampled by a 100-iteration random search, scored by 5-fold cross-validation repeated 5 times on the training rows.

Hyperparameter	Distribution	Range
`n_estimators`	Integer	[50, 1000]
`max_depth`	Integer	[2, 6]
`min_child_weight`	Log-uniform	[1, 100]
`subsample`	Uniform	[0.5, 1.0]
`learning_rate`	Log-uniform	$[10^{-5}, 1]$
`colsample_bylevel`	Uniform	[0.5, 1.0]
`colsample_bytree`	Uniform	[0.5, 1.0]
`gamma`	Log-uniform	$[10^{-8}, 7]$
`reg_lambda`	Log-uniform	[1, 4]
`alpha`	Log-uniform	$[10^{-8}, 100]$

TabLLM TabLLM (Hegselmann et al., 2023) serializes each row as a sentence (*Text* template: `"The <column> is <value>."`) and fine-tunes a T0 model (Sanh et al., 2022) with $(IA)^3$ following the t-few recipe (Liu et al., 2022). We run the original TabLLM and t-few code at pinned commits, with the configuration behind the paper's main results—*Text* serialization, $(IA)^3$ initialized from the 100k-step multitask pre-trained checkpoint, 30 epochs, batch size 4, learning rate $3 \cdot 10^{-3}$, length-normalized answer-choice scoring, inputs truncated at 1024 tokens—with one substitution: we use T0-3B rather than T0-11B, the variant TabLLM reports in its appendix and that reaches similar performance, because T0-11B runs out of memory at $n_{\text{train}} = 1{,}024$ samples (and otherwise would cost on the order of 10^3 GPU-hours). Two further deviations are required for comparability: the model is trained on the benchmark's fixed stratified splits (`num_shot=all` on exactly the n_{train} rows every other baseline sees) instead of TabLLM's own $80/20$ re-split and class-balanced sampling with replacement, and the predictions are scored downstream with the same metric code as every other baseline. Since TabLLM is a classifier, it is evaluated on the classification part of the benchmark only.

CASE We run CASE (Schindler et al., 2026) with the authors' official implementation (commit `64a0a79`) and its default configuration. As the table-tuned TabGemma checkpoints of the paper have not been released, we use the fallback backbone recommended in the official repository, `google/gemma-3-4b-pt`, in `bfloat16`. All other settings are the library defaults: the KV cache is prefilled with 128 training rows and their targets under a 16k-token budget, with at most 50 tokens per cell; each row is embedded as the last-layer hidden state at its final token; and the row embeddings are reduced to 32 components by PCA and appended to the raw features, which the downstream learner preprocesses itself. Our only deviation is the attention implementation: PyTorch's SDPA rather than FlashAttention-2, which computes the same attention without a compiled dependency. CASE is fit per split, with the context rows and the PCA taken from the training rows only. The resulting features are shared by TabICLv2, TabPFN-2.5 and TabPFN-3, run at their default settings on the same splits as every other baseline. Reported preprocessing time covers embedding the training and test rows and excludes loading the backbone, which is done once per job.

F Additional experimental results

F.1 The improvability metric

The cost analyses report the *improvability* of Erickson et al. (2025), the fraction of a method's error ($1 - R^2$ for regression, $1 - \text{AUC}$ for classification) that would be removed by switching to the best available method on each dataset, averaged over datasets and seeds. A method that is best on every dataset has an improvability of 0%.

F.2 The feature-capacity window of each learner

How many features can a table foundation model actually digest? The number of features used to pre-train PFNs is usually much lower than the number of dimensions of an LLM embedding (Tajjar et al., 2026; Wang et al., 2026), and being transformers they likely inherit the shortcomings of attention with large contexts (Liu et al., 2024). To measure this, we sweep the number of components of a single PCA applied to a row's concatenated cell embeddings, so that the sweep sets the learner's total feature count directly, independently of the table's width.

Figure F.1 shows a clear trade-off: too few components and the projection discards information, too many and the learner is handed more features than it handles comfortably. Between the two, each curve has a broad plateau rather than a sharp optimum. We define a learner's *feature-capacity window* as the range of PCA dimensions over which its mean normalized score over all tasks remains flat, that is, within about one point (0.01) of the learner's optimum. This gives $[100, 300]$ for TabPFN-2.5, $[50, 300]$ for TabICLv2, $[50, 250]$ for TabPFN-3, $[75, 300]$ for TabPFN-3.5, $[75, 250]$ for EXAONE-Tabular and $[75, 400]$ for Causilo. These windows are what TILE sizes its per-column PCA against (Section 3.1); the number of components per column is left at its default unless the resulting feature count would fall outside the window of the learner at hand.

F.3 Cardinality threshold for categorical columns

TILE passes low-cardinality string columns to the tabular learner as native categorical features in addition to target encoding and serializing them. The cardinality threshold therefore controls how much information is represented by exact category identity.

Figure F.2 sweeps the threshold from 5 to 100 over the 100 datasets, for TabPFN-2.5, TabICLv2 and TabPFN-3. The curves are mostly flat for all three learners and both problem types, so the threshold matters little. We use 50 by default, but other values would serve as well.

F.4 Choosing the pooling strategy

Figure F.3 compares three ways of pooling the token embeddings of Llama-3.1-8B into a row representation, at every layer: mean pooling over the whole sequence, the embedding of the last token, and columnar pooling. As in Section 5.2, the representations are scored with Ridge on the embeddings alone, without the by-pass features, on the 27 probing datasets of Section A.5. Columnar pooling is ahead at every layer, and at layer 14 it reaches 0.85, against 0.72 for mean pooling of the last layer, the default of the sentence-transformers library; we therefore use it in TILE. This comparison favors columnar pooling in one respect: Ridge is applied without PCA, so columnar pooling hands it one embedding per column, an input M times wider than the single vector of mean or last-token pooling, and part of its advantage may come from that extra dimensionality.

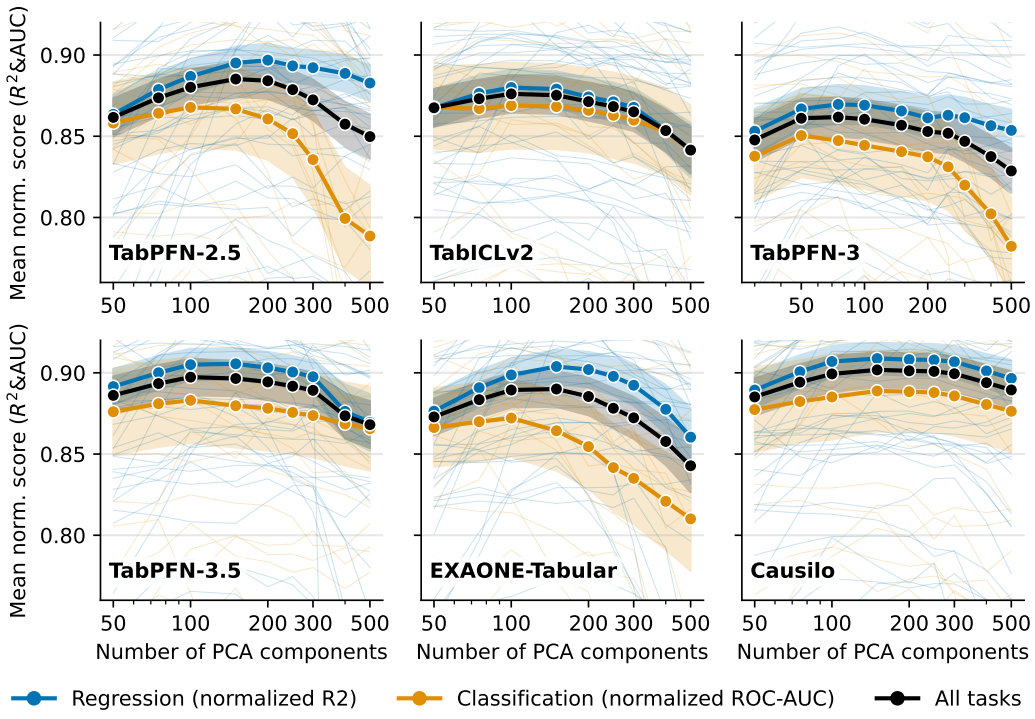


Figure F.1: **Each learner handles a broad range of feature counts.** Averaged score as a function of the number of components of a single PCA over the concatenated cell embeddings, which here sets the number of features the learner receives. Every estimator benefits from the projection, and each stays close to its best over a wide plateau, from which we read the feature-capacity windows used in Section 3.1. Thick lines represent aggregated performance across the 100 datasets, while thin lines represent individual datasets. Error intervals display the standard error of the mean across datasets.

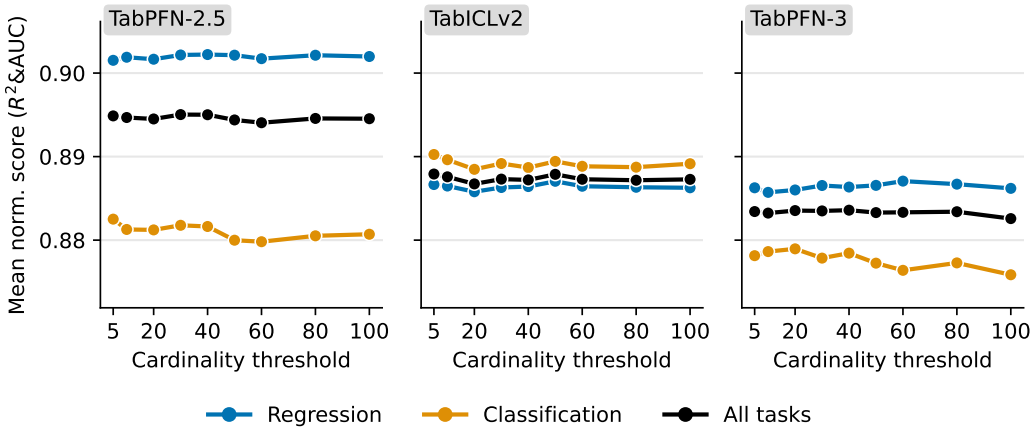


Figure F.2: **The categorical cardinality threshold has little effect.** Averaged normalized score of TILE as a function of the cardinality threshold below which a string column is also passed directly to the tabular learner, for TabPFN-2.5, TabICLv2 and TabPFN-3. The curves are mostly flat. We use 50 in all other experiments.

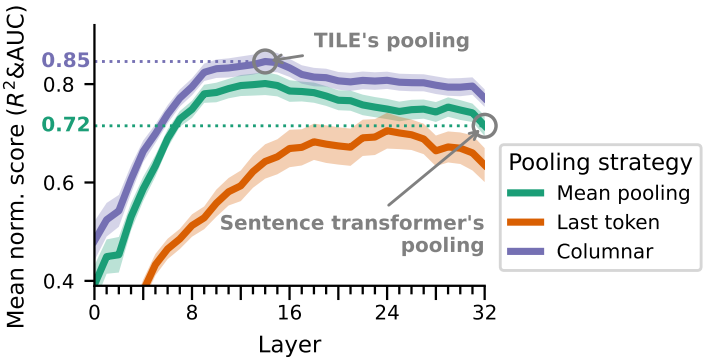


Figure F.3: **Columnar pooling gives the best representations.** Performance of three pooling strategies across the layers of Llama-3.1-8B, scored by Ridge on the embeddings alone (no by-pass features), on the 27 probing datasets of Section A.5. Ridge is used without PCA, so columnar pooling gives it a higher-dimensional input than the other two strategies. Scores are normalized over the strategies shown here, so they are not comparable with those of the other figures.

G Compute and memory costs

G.1 Hardware and total compute

All experiments were conducted on a high-performance computing cluster. Model training and evaluation, including TILE and all baseline methods, were executed using different hardware. The TILE model can typically be run with a single NVIDIA V100 GPU (32GB VRAM) and 15 CPU cores.

In total, this project consumed approximately 15,200 GPU compute hours. This total consists of an estimated 6,500 GPU hours dedicated to developing our method, running ablation studies, and conducting exploratory research, alongside 8,700 GPU hours for the evaluation and hyperparameter tuning of the baseline models.

G.2 Runtime by task

Figure G.1 plots improvability against median runtime per dataset, separately for each problem type. TILE holds the accurate end of the Pareto front in both: with Causilo, TabPFN-2.5, TabICLv2 and TabPFN-3 in regression, and with Causilo, TabICLv2 and TabPFN-3 in classification. The TILE variants take 26 to 45 s per dataset (median). TF-IDF pipelines hold the cheap end of the front, with Ridge, then Causilo and TabPFN-3.5. In classification, CASE with EXAONE-Tabular and several LLM-encoder pipelines also reach the front. The best baseline sits at 18% improvability in regression and 21% in classification, against 2% and 5% for TILE with Causilo. Of the error that separates the best existing pipeline from the per-dataset best, TILE thus removes nearly nine tenths in regression and about four fifths in classification.

Two caveats apply to these runtimes. First, our LLM encoder embeds each unique cell value only once and broadcasts the embedding back to the table (Appendix E), an optimization that `skrub` does not implement; without it, the LLM encoder is several times slower than TILE. Second, TILE and the LLM encoder ran on V100 GPUs, whereas CASE, limited by memory, ran on A100 and H100 GPUs, so its lower runtimes are largely an artifact of the faster hardware.

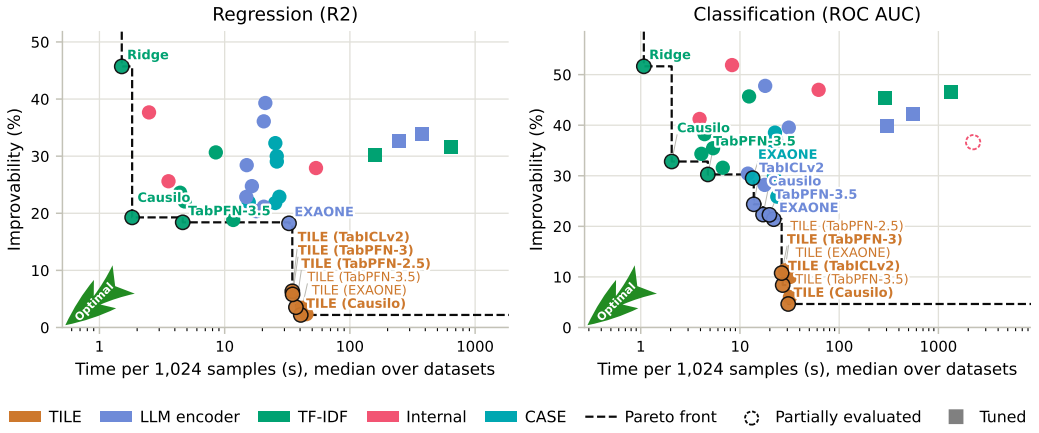


Figure G.1: **TILE holds the accurate end of the Pareto front.** Improvability against median time per dataset at $n_{\text{train}} = 1{,}024$, split into regression (left) and classification (right). The dashed staircase is the Pareto front; Pareto-optimal pipelines are labeled in bold, and the other TILE variants in light type. Colors indicate the string encoding and labels the learner; squares mark tuned baselines. CASE ran on faster GPUs than TILE and the LLM encoder (Section G.2). TabLLM supports classification only, and is evaluated on the 34 datasets it completes.

G.3 Peak GPU memory

Figure G.2 reports the memory counterpart of Figure G.1, measured on a V100-32GB with batches of 16 rows, on the `chocolate_bar_ratings` dataset of CARTE (Kim et al., 2024). TILE peaks

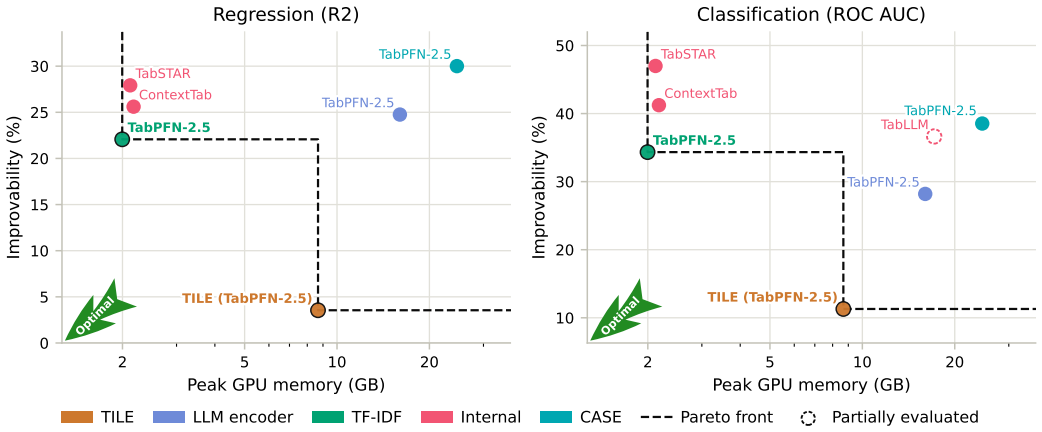


Figure G.2: **TILE nearly halves the memory of the same LLM applied cell-wise.** Improvability (Section F.1, lower is better) against the peak GPU memory a method needs on 1,024 train rows of the `chocolate_bar_ratings` dataset of CARTE, taken as the larger of its two stages: building the representation, then fitting the learner and predicting. As in Figure G.1, both axes are costs and the dashed staircase is the Pareto front. We report it for TILE, for TF-IDF, the LLM encoder and CASE combined with the same learner (TabPFN-2.5), and for the three end-to-end models, which read the raw table with a single network. TabLLM supports classification only, and runs out of memory on one of the 35 classification datasets, hence the open "partially evaluated" marker: its improvability is computed on the 34 datasets it completes.

at 8.7 GB against 16.0 GB for the cell-wise LLM encoder built on the same Llama-3.1-8B. Loading only the layers up to the extraction point, about 45% of the model, nearly halves the footprint of the LLM stage, and TILE is by far the more accurate of the two. The learner that follows is light in comparison, needing about 2 GB on the projected representation, as much as on TF-IDF features, so the additional memory TILE requires is entirely the language model and is released once the representation is built.

The front has only two points on this axis. TF-IDF with TabPFN-2.5 holds the frugal end at 2.0 GB, and ContextTab and TabSTAR sit just beside it at 2.1–2.2 GB while being markedly less accurate. TabLLM needs 17.1 GB—twice TILE's footprint—to generate its predictions as text, for a much higher improvability. CASE is the heaviest method, peaking at 24.6 GB, about three times TILE: it keeps its Gemma-3-4B backbone loaded while TabPFN-2.5 runs, so the backbone and the learner count together. Its accuracy does not make up for it: at 30% improvability in regression and 38% in classification, against 3.6% and 11.3% for TILE, it sits in the dominated top-right corner of both panels. Relative to the frugal end, TILE trades about four times the memory for an improvability divided by six in regression and by about three in classification. The LLM stage of TILE, not the tabular learner, is the binding constraint on the hardware it requires.

Table H.1: **Raw scores at every training-set size.** Unnormalized counterpart of Table 1: R^2 for regression and ROC-AUC for classification, multiplied by 100 and aggregated over the datasets large enough for each training size; their number in parentheses. Because the set of datasets changes with the training size, compare scores within a column rather than across. LLM enc. is Llama-3.1-8B applied to each cell content in isolation, the same LLM as TILE; TF-IDF is followed by an SVD; CASE uses Gemma-3-4B (Appendix E). Best result in bold, second best underlined.

	Regression (R^2)			Classification (AUC)		
Method	**1k (65)**	**10k (38)**	**50k (7)**	**1k (35)**	**10k (10)**	**50k (2)**
TILE (TabPFN-2.5)	70.8	76.7	82.6	89.4	90.5	82.6
TILE (TabICLv2)	70.1	76.4	82.4	89.6	90.3	82.6
TILE (TabPFN-3)	70.2	76.4	82.3	89.4	90.5	82.7
TILE (TabPFN-3.5)	<u>70.9</u>	<u>77.1</u>	<u>83.4</u>	<u>89.7</u>	<u>90.6</u>	**83.1**
TILE (Causilo)	**71.3**	**77.8**	**84.1**	**89.7**	**90.6**	<u>83.0</u>
CASE + TabPFN-2.5	58.4	61.8	53.9	86.8	89.3	78.4
CASE + TabICLv2	57.1	62.1	55.9	87.3	89.4	79.2
CASE + TabPFN-3	58.3	62.5	55.3	86.8	89.4	79.4
CASE + TabPFN-3.5	61.8	67.2	59.5	88.2	90.2	81.2
CASE + Causilo	63.2	70.9	76.3	88.0	89.8	80.8
ContextTab	63.1	71.0	74.5	87.1	89.4	79.7
TabSTAR	61.9	71.0	79.4	86.1	89.0	79.5
CatBoost	55.7	65.7	70.9	83.2	89.0	80.8
LLM enc. + TabPFN-2.5	62.9	72.4	79.9	88.3	90.0	80.0
LLM enc. + TabICLv2	60.6	72.7	80.0	88.5	90.0	81.3
LLM enc. + TabPFN-3	63.9	70.7	79.3	87.7	89.0	79.8
LLM enc. + TabPFN-3.5	64.0	73.6	80.0	88.5	89.9	80.7
LLM enc. + Causilo	64.7	73.5	81.1	88.6	90.0	81.0
LLM enc. + RealMLP	58.1	70.5	80.3	86.4	89.2	80.0
LLM enc. + XGBoost	58.9	70.8	79.6	87.3	89.6	80.7
LLM enc. + CatBoost	55.0	71.3	79.2	87.2	89.5	80.6
LLM enc. + Ridge	56.9	63.9	74.0	86.1	87.5	78.1
TF-IDF + TabPFN-2.5	64.0	71.7	78.7	87.3	89.3	79.8
TF-IDF + TabICLv2	63.0	71.1	79.2	87.3	89.3	80.5
TF-IDF + TabPFN-3	63.3	71.0	79.7	86.7	89.0	79.7
TF-IDF + TabPFN-3.5	65.4	74.0	81.3	87.9	89.9	81.6
TF-IDF + Causilo	64.6	73.2	80.6	87.6	89.5	81.0
TF-IDF + RealMLP	60.1	70.3	78.9	85.6	88.5	79.8
TF-IDF + XGBoost	60.8	70.0	78.4	86.0	88.7	80.0
TF-IDF + CatBoost	60.5	70.1	77.9	85.9	88.7	80.1
TF-IDF + Ridge	48.8	32.4	44.5	84.3	75.8	75.7

H RAW RESULTS

Table H.1 gives the raw scores behind Table 1, for every method and every training-set size. Table H.2 gives the normalized scores of every method, including the learners omitted from Table 1.

Table H.3 reports how TILE's performance changes with the base LLM choice and with the ablations presented in Figure 4 (removing inter-column attention or using the base LLM directly on cell content, the LLM encoder).

Raw scores on STRABLE Table H.5 gives the unnormalized scores behind Figure 3: the mean ROC-AUC over STRABLE's 32 classification datasets and the mean R^2 over its 76 regression datasets, for the four TILE variants and the STRABLE baselines. The baselines are run with STRABLE's protocol and implementations (Blayer et al., 2026), not those of Appendix E. Most pair a string encoder with a tabular learner:

- **String encoders:** TF-IDF; FastText (Bojanowski et al., 2017); the sentence encoders E5-small-v2 (Wang et al., 2022), All-MiniLM-L6-v2 (Wang et al., 2020), Jasper-0.6B (Zhang et al., 2025) and Qwen3-Embedding-8B (Yang et al., 2025); Llama-3.1-8B (Grattafiori et al., 2024); Skrub's

Table H.2: **Normalized scores at every training-set size, all learners.** Full version of Table 1. Normalized scores $\times 100$, aggregated over the datasets large enough for each size; counts in parentheses. Because the set of datasets varies, compare scores within columns only. LLM enc. is Llama-3.1-8B applied to each cell independently; TF-IDF is followed by SVD; CASE uses Gemma-3-4B (Appendix E). Best result in bold, second best underlined. Table H.1 gives raw scores.

	Regression (normalized R^2)			Classification (normalized AUC)		
Method	**1,024 (65)**	**10k (38)**	**50k (7)**	**1,024 (35)**	**10k (10)**	**50k (2)**
TILE (TabPFN-2.5)	90.21	93.19	92.58	88.00	87.62	79.64
TILE (TabICLv2)	88.70	92.32	92.00	88.94	85.93	80.17
TILE (TabPFN-3)	88.66	92.32	92.14	87.72	87.67	80.68
TILE (TabPFN-3.5)	<u>90.67</u>	<u>94.11</u>	<u>94.66</u>	<u>90.35</u>	**88.31**	**84.58**
TILE (Causilo)	**91.63**	**95.20**	**96.05**	**90.38**	<u>88.02</u>	<u>83.67</u>
CASE + TabPFN-2.5	61.97	64.96	49.11	67.50	77.13	35.14
CASE + TabICLv2	59.43	63.19	48.94	70.45	76.86	42.45
CASE + TabPFN-3	63.35	66.28	53.68	67.51	77.80	44.33
CASE + TabPFN-3.5	70.74	74.16	57.43	78.45	84.90	63.31
CASE + Causilo	71.88	77.00	68.16	76.52	80.83	59.23
ContextTab	70.78	78.83	66.68	69.96	80.46	49.36
TabSTAR	66.96	79.52	81.26	57.40	75.75	46.36
CatBoost	52.33	64.40	51.89	41.02	75.67	62.07
LLM enc. + TabPFN-2.5	72.97	83.20	85.26	79.48	83.24	52.16
LLM enc. + TabICLv2	68.82	83.61	85.67	80.59	83.46	66.50
LLM enc. + TabPFN-3	74.09	79.25	83.21	74.12	76.97	49.29
LLM enc. + TabPFN-3.5	76.36	86.03	85.81	80.65	83.51	60.38
LLM enc. + Causilo	77.84	85.10	88.73	80.99	83.34	63.75
LLM enc. + RealMLP	61.65	79.02	85.37	63.28	77.64	52.22
LLM enc. + XGBoost	64.29	79.10	83.11	70.21	81.11	58.73
LLM enc. + CatBoost	53.18	79.59	82.05	68.86	79.97	58.24
LLM enc. + Ridge	56.28	62.99	66.20	53.58	60.17	32.36
TF-IDF + TabPFN-2.5	72.91	80.41	79.55	72.59	78.37	50.79
TF-IDF + TabICLv2	71.03	79.28	79.86	71.80	77.83	57.42
TF-IDF + TabPFN-3	71.61	78.73	81.86	67.36	75.14	49.37
TF-IDF + TabPFN-3.5	76.57	86.12	86.65	76.47	82.54	69.50
TF-IDF + Causilo	75.11	84.13	84.50	74.00	80.05	63.16
TF-IDF + RealMLP	62.95	77.37	80.18	57.62	71.65	50.36
TF-IDF + XGBoost	64.98	76.38	77.80	60.93	74.56	52.51
TF-IDF + CatBoost	64.28	76.78	76.50	59.75	73.04	54.19
TF-IDF + Ridge	39.72	17.24	12.40	38.62	15.04	12.18

TargetEncoder (Micci-Barreca, 2001); and TARTE (Kim et al., 2025). The language models embed each cell on its own.

- **Tabular learners:** the table foundation models TabPFN-2.5 (Grinsztajn et al., 2025), TabICLv2 (Qu et al., 2026) and Causilo (Cho et al., 2026), and the learners trained from scratch TabM (Gorishniy et al., 2025), RealMLP (Holzmüller et al., 2024), XGBoost (Chen & Guestrin, 2016), ExtraTrees and Ridge. The suffix "-tuned" marks a hyper-parameter-tuned learner.
- **End-to-end models**, which handle strings internally: CatBoost (Prokhorenkova et al., 2018), ContextTab (Spinaci et al., 2025), TabSTAR (Arazi et al., 2025) and Mambular (Thielmann et al., 2024).

Tables H.6–H.23 report the raw R^2 and ROC-AUC scores of all methods on each individual dataset, averaged over 5 random seeds.

Table H.3: **Serialization ablation results aggregated over 100 datasets.** Scores are raw R^2/AUC multiplied by 100; best is in bold and second-best is underlined. These results correspond to the ablations presented in Figure 4. The cell-wise LLM encoder could not be run with Llama-3.1-70B: the full-depth model weights exceed the memory of our GPUs. TILE runs with that backbone, as it only loads the layers up to the extraction point.

Method	Base LLM	Regression (R^2)	Classification (AUC)
TILE	Llama-3.1-70B	**72.16**	**89.82**
	Llama-3.1-8B	<u>70.76</u>	<u>89.44</u>
	Llama-3.2-3B	69.27	89.03
	Llama-3.2-1B	68.33	89.02
	Qwen3-8B-Base	69.35	89.02
	Qwen3-4B-Base	68.51	88.80
	Qwen3-0.6B-Base	66.56	88.23
TILE w/o inter-col. attention	Llama-3.1-70B	67.12	89.22
	Llama-3.1-8B	68.06	89.07
	Llama-3.2-3B	68.37	88.91
	Llama-3.2-1B	67.28	88.80
	Qwen3-8B-Base	65.98	88.92
	Qwen3-4B-Base	65.46	88.72
	Qwen3-0.6B-Base	65.83	88.27
Base LLM on cell content (LLM encoder)	Llama-3.1-70B	*Out of memory*	*Out of memory*
	Llama-3.1-8B	62.94	88.33
	Llama-3.2-3B	61.53	88.33
	Llama-3.2-1B	61.36	88.11
	Qwen3-8B-Base	59.67	87.78
	Qwen3-4B-Base	59.84	87.61
	Qwen3-0.6B-Base	64.27	87.72

Table H.4: **Ablation study, raw scores.** The ablations of Table 2, with the raw R^2/AUC ($\times 100$) of the default architecture for each learner, and the raw difference obtained for each variation. Differences on classification are compressed here because ROC-AUC saturates near the top of its range; the normalized scores of Table 2 place the two problem types on a comparable scale.

	TabPFN-2.5		TabICLv2		TabPFN-3	
Model	**AUC**	R^2	**AUC**	R^2	**AUC**	R^2
− numerical by-pass	-0.47	-4.41	-0.50	-4.02	-0.52	-4.39
− inter-column attention	-0.37	-2.70	-0.34	-3.65	-0.83	-3.87
− intermediate layer (last layer)	-0.01	-1.51	-0.09	-1.50	-0.01	-1.38
− numbers in the serialization	-0.14	-0.49	-0.16	-0.65	-0.15	-0.68
− feature-capacity window	-0.02	-0.74	-0.01	-0.18	+0.01	-0.47
− target-encoding by-pass	-0.05	-0.13	-0.12	-0.00	-0.13	-0.03
− standard scaling	-0.03	-0.12	-0.04	-0.10	-0.07	-0.17
− categorical by-pass	+0.02	-0.06	+0.01	-0.01	+0.01	-0.04
TILE (default)	**89.44**	**70.76**	**89.57**	**70.14**	**89.37**	**70.19**
+ numerical-cell embeddings	-0.03	-0.03	-0.03	-0.01	-0.05	-0.01
+ one-hot instead of native cat.	-0.05	-0.07	+0.02	-0.02	+0.02	-0.06
+ target name	-0.20	-0.27	-0.14	-0.26	-0.16	-0.33
+ repeated serialization	-0.16	-0.40	-0.19	-0.42	-0.23	-0.51
+ mirrored column order	-0.20	-0.66	-0.24	-0.85	-0.28	-0.99
+ uninformative column names	-0.26	-1.13	-0.31	-1.26	-0.31	-1.09
LLM encoder (`skrub`)	-1.11	-7.82	-1.03	-9.50	-1.71	-6.32
LLM encoder + by-pass features	-1.00	-6.73	-0.98	-8.46	-1.86	-7.00
LLM encoder + feature-capacity window	-1.15	-7.43	-1.05	-8.76	-2.12	-8.40

Table H.5: **Raw scores on STRABLE.** Unnormalized counterpart of Figure 3, multiplied by 100: the ROC-AUC averaged over the 32 classification datasets, and the R^2 (clipped at 0) averaged over the 76 regression datasets, each dataset scored by its average over 3 cross-validation folds. The $\pm$ is the standard error of that mean due to the cross-validation splits. Labels in parentheses are STRABLE's preprocessing variants of the same pipeline. Best result in bold.

Pipeline	**Classification AUC (32)**	**Regression R^2 (76)**
TILE + Causilo	**86.4** ±0.1	**73.0** ±0.1
TILE + TabPFN-2.5	86.0 ±0.1	72.1 ±0.1
TILE + TabICLv2	85.9 ±0.1	71.4 ±0.1
TILE + TabPFN-3	85.7 ±0.1	71.3 ±0.1
TF-IDF + Causilo	**86.4** ±0.1	70.7 ±0.2
TF-IDF + TabPFN-2.5	85.9 ±0.1	69.7 ±0.2
Qwen3-Embedding-8B (no PCA) + TabPFN-2.5	85.5 ±0.1	70.0 ±0.1
E5-small-v2 + TabPFN-2.5	85.7 ±0.1	67.5 ±0.2
FastText + TabPFN-2.5	85.2 ±0.1	67.7 ±0.2
TF-IDF + TabICLv2	85.4 ±0.1	68.3 ±0.2
Qwen3-Embedding-8B + TabM	84.4 ±0.1	68.4 ±0.1
Qwen3-Embedding-8B + TabICLv2 (no PCA)	85.1 ±0.1	68.6 ±0.2
All-MiniLM-L6-v2 + TabPFN-2.5	85.4 ±0.2	67.2 ±0.3
Llama-3.1-8B + TabM	84.1 ±0.1	68.0 ±0.2
TF-IDF + XGBoost-tuned	85.0 ±0.2	67.5 ±0.2
ContextTab	84.2 ±0.1	68.1 ±0.2
TF-IDF + RealMLP	83.8 ±0.1	67.9 ±0.2
All-MiniLM-L6-v2 + TabM	83.6 ±0.1	67.2 ±0.1
E5-small-v2 + XGBoost-tuned	84.7 ±0.1	66.4 ±0.2
E5-small-v2 + ExtraTrees-tuned	84.5 ±0.1	66.9 ±0.2
TF-IDF + ExtraTrees-tuned	84.5 ±0.1	66.6 ±0.2
All-MiniLM-L6-v2 + ExtraTrees-tuned	84.3 ±0.1	66.7 ±0.2
FastText + XGBoost-tuned	84.1 ±0.2	66.3 ±0.1
All-MiniLM-L6-v2 + RealMLP	83.0 ±0.1	67.3 ±0.2
All-MiniLM-L6-v2 + XGBoost-tuned	84.3 ±0.1	66.0 ±0.2
TF-IDF + TabM	83.7 ±0.1	66.1 ±0.2
FastText + ExtraTrees-tuned	83.8 ±0.2	66.6 ±0.2
Llama-3.1-8B + ExtraTrees-tuned	84.4 ±0.1	66.2 ±0.2
Llama-3.1-8B + RealMLP	83.3 ±0.2	66.3 ±0.4
Qwen3-Embedding-8B + ExtraTrees-tuned	84.7 ±0.1	65.9 ±0.2
TargetEncoder + TabPFN-2.5	82.8 ±0.1	66.6 ±0.2
Qwen3-Embedding-8B + RealMLP	83.4 ±0.1	66.6 ±0.2
Jasper-0.6B + ExtraTrees-tuned	84.5 ±0.1	65.3 ±0.2
TabSTAR	83.7 ±0.3	64.3 ±0.2
Target encoding + XGBoost-tuned	82.2 ±0.2	65.2 ±0.2
Qwen3-Embedding-8B + TabICLv2	84.7 ±0.2	62.3 ±0.3
E5-small-v2 + ExtraTrees	81.2 ±0.1	66.0 ±0.2
Target encoding + ExtraTrees-tuned	82.0 ±0.2	64.9 ±0.2
CatBoost	82.2 ±0.1	64.2 ±0.2
TARTE + TabPFN-2.5	81.3 ±0.2	65.2 ±0.2
CatBoost-tuned	82.0 ±0.2	64.2 ±0.2
Llama-3.1-8B + XGBoost-tuned	84.4 ±0.2	61.2 ±0.5
Qwen3-Embedding-8B + XGBoost-tuned	84.5 ±0.1	61.1 ±0.4
Jasper-0.6B + XGBoost-tuned	83.9 ±0.2	61.2 ±0.4
TF-IDF + XGBoost	82.7 ±0.1	63.3 ±0.2
Qwen3-Embedding-8B + TabPFN-2.5	85.1 ±0.1	58.4 ±0.5
Llama-3.1-8B + TabPFN-2.5	84.5 ±0.1	57.8 ±0.4
Jasper-0.6B + TabPFN-2.5	85.0 ±0.1	57.9 ±0.4
Llama-3.1-8B + TabICLv2 (StandScal + PCA (30))	84.3 ±0.2	57.8 ±0.5
Llama-3.1-8B + TabICLv2	84.4 ±0.2	57.4 ±0.6
TARTE + XGBoost-tuned	79.8 ±0.2	60.1 ±0.1
Qwen3-Embedding-8B + Ridge (OHE | CT=30)	80.8 ±0.1	56.5 ±0.2
Qwen3-Embedding-8B + Ridge	80.8 ±0.1	56.1 ±0.2
TARTE + ExtraTrees-tuned	78.3 ±0.1	56.5 ±0.1
Mambular	75.8 ±0.2	49.9 ±0.2

Table H.6: **Regression results for TextTabBench datasets (part 1 of 3).** Raw R^2 scores (clipped at 0 if negative) multiplied by 100, averaged over 5 random seeds for a train size of 1,024 samples, for TabPFN-2.5, TabICLv2 and TabPFN-3, each fed with TILE, CASE, the cell-wise LLM encoder (Llama-3.1-8B) or TF-IDF. Best in bold, second best underlined, both computed over all 35 methods.

Method	**TabPFN-2.5**				**TabICLv2**				**TabPFN-3**			
String encoding	TILE	CASE	LLM encoder	TF-IDF	TILE	CASE	LLM encoder	TF-IDF	TILE	CASE	LLM encoder	TF-IDF
Airbnb	73.78	72.38	55.48	73.77	71.20	69.86	52.16	71.66	72.17	70.80	64.30	71.24
Beer	66.99	57.30	62.73	58.71	66.33	58.02	62.68	58.78	66.19	59.08	62.91	58.45
COVID Trials	22.49	23.22	18.62	25.07	23.20	22.50	14.19	24.82	22.55	22.30	18.64	23.90
California Houses	69.63	65.18	67.11	67.24	68.08	66.22	66.64	66.53	68.26	63.67	66.49	66.60
IT Salary	**48.22**	46.77	39.18	46.93	45.57	44.35	0.84	33.50	47.85	38.81	42.39	44.74
Insurance Complaints	42.55	39.68	35.48	42.34	45.15	44.46	35.87	44.30	44.09	43.69	35.85	42.98
Mercari	43.30	24.09	32.50	26.53	41.63	23.81	32.83	26.84	41.58	26.24	29.71	25.27
San Francisco Permits	61.44	62.87	57.09	63.65	62.31	63.46	47.41	63.99	62.38	63.34	60.36	64.18
Stack Overflow	99.70	99.82	85.96	99.67	99.78	99.78	87.65	99.86	99.59	97.08	79.10	99.70
Wine	87.79	74.79	69.19	76.90	87.23	73.21	65.46	73.73	86.78	75.50	71.08	74.92

Table H.7: **Regression results for TextTabBench datasets (part 2 of 3).** Raw R^2 scores (clipped at 0 if negative) multiplied by 100, averaged over 5 random seeds for a train size of 1,024 samples, for TabPFN-3.5, EXAONE-Tabular and Causilo, each fed with TILE, CASE, the cell-wise LLM encoder (Llama-3.1-8B) or TF-IDF. Best in bold, second best underlined, both computed over all 35 methods.

Method	**TabPFN-3.5**				**EXAONE-Tabular**				**Causilo**			
String encoding	TILE	CASE	LLM encoder	TF-IDF	TILE	CASE	LLM encoder	TF-IDF	TILE	CASE	LLM encoder	TF-IDF
Airbnb	72.27	70.12	66.51	73.15	**74.52**	73.57	64.80	74.27	<u>74.29</u>	73.30	64.22	74.21
Beer	66.80	64.28	64.45	61.84	**67.50**	59.20	64.30	60.66	<u>67.24</u>	59.75	64.02	59.32
COVID Trials	25.00	26.13	23.90	<u>26.32</u>	23.03	24.81	22.51	25.58	25.85	26.21	20.01	**26.76**
California Houses	68.93	66.28	66.92	68.01	<u>69.99</u>	68.51	68.13	68.51	**70.74**	68.07	68.10	68.14
IT Salary	<u>47.98</u>	46.43	38.28	45.50	47.63	46.54	41.60	45.44	46.68	44.38	20.59	29.52
Insurance Complaints	44.81	45.79	26.50	<u>45.98</u>	44.34	44.46	40.51	45.39	44.98	45.46	39.28	44.65
Mercari	42.41	31.58	34.47	29.91	<u>44.79</u>	26.67	35.13	29.58	**44.96**	29.74	34.63	29.36
San Francisco Permits	62.82	63.67	57.83	<u>64.37</u>	62.93	63.68	60.91	64.20	63.33	63.91	61.15	**64.47**
Stack Overflow	99.79	99.81	77.97	99.78	99.93	**99.94**	99.88	<u>99.93</u>	99.87	99.89	99.26	99.86
Wine	88.09	77.83	78.81	77.63	<u>88.41</u>	76.19	76.96	78.33	**88.78**	74.86	79.91	77.29

Table H.8: **Regression results for TextTabBench datasets (part 3 of 3).** Raw R^2 scores (clipped at 0 if negative) multiplied by 100, averaged over 5 random seeds for a train size of 1,024 samples, for the remaining learners, fed with the cell-wise LLM encoder (Llama-3.1-8B) or TF-IDF, and the methods with internal string handling. Best in bold, second best underlined, both computed over all 35 methods.

Method	**XGBoost (tuned)**		**CatBoost**			**RealMLP (tuned)**		**Ridge**		**ContextTab**	**TabSTAR**
String encoding	LLM encoder	TF-IDF	LLM encoder	TF-IDF	internal	LLM encoder	TF-IDF	LLM encoder	TF-IDF	internal	internal
Airbnb	60.09	71.36	48.67	72.27	71.90	65.84	71.47	62.41	68.56	72.74	69.71
Beer	59.74	56.62	60.18	56.38	56.55	61.33	56.68	53.68	48.06	62.22	49.16
COVID Trials	15.25	21.50	13.18	22.67	18.39	13.15	16.48	4.07	14.99	15.77	14.98
California Houses	61.95	62.50	59.23	60.85	62.03	62.03	62.00	53.55	58.26	65.82	64.72
IT Salary	28.56	36.56	24.09	33.31	43.51	33.82	37.86	25.08	28.13	45.77	41.59
Insurance Complaints	32.40	41.74	29.68	42.71	42.79	28.58	40.83	36.90	37.19	**48.72**	32.26
Mercari	30.62	25.55	30.63	25.43	15.53	31.38	24.86	25.72	21.42	26.93	41.42
San Francisco Permits	53.23	62.59	51.92	61.86	59.84	36.24	62.40	54.77	54.11	60.27	59.41
Stack Overflow	96.85	97.90	94.04	98.22	89.78	90.71	97.47	31.92	41.50	82.93	97.53
Wine	71.74	70.34	65.02	68.92	69.20	68.77	74.97	76.15	69.35	73.04	67.75

Table H.9: **Regression results for CARTE datasets (part 1 of 3).** Raw R^2 scores (clipped at 0 if negative) multiplied by 100, averaged over 5 random seeds for a train size of 1,024 samples, for TabPFN-2.5, TabICLv2 and TabPFN-3, each fed with TILE, CASE, the cell-wise LLM encoder (Llama-3.1-8B) or TF-IDF. Best in bold, second best underlined, both computed over all 35 methods.

Method	**TabPFN-2.5**				**TabICLv2**				**TabPFN-3**			
String encoding	TILE	CASE	LLM encoder	TF-IDF	TILE	CASE	LLM encoder	TF-IDF	TILE	CASE	LLM encoder	TF-IDF
Anime Planet	64.08	46.31	54.11	54.43	63.69	39.48	56.66	54.29	63.21	45.92	57.03	54.52
Babies R Us	81.78	76.26	53.45	79.38	82.64	71.75	68.73	79.39	82.46	77.76	70.49	79.84
Beer Ratings	61.27	51.78	57.67	53.38	60.23	52.03	57.78	54.36	59.89	52.80	56.61	52.32
BikeDekho	**80.86**	78.12	63.47	78.81	80.01	77.75	71.36	78.66	<u>80.22</u>	78.85	74.62	78.73
BikeWale	80.77	72.46	76.74	75.04	80.26	72.70	77.14	75.32	80.50	75.36	78.82	74.96
Buy Buy Baby	81.88	76.43	68.66	77.79	81.51	74.99	77.98	78.22	81.84	78.71	78.86	77.76
CLEAR Corpus	66.39	61.68	69.13	60.34	65.62	60.38	65.13	60.58	63.82	62.15	65.58	60.23
CarDekho	92.92	82.36	88.87	92.28	91.84	81.94	81.26	91.13	91.62	85.45	88.22	90.99
Company Employees	83.52	80.43	59.13	79.87	83.77	80.43	36.47	79.53	<u>84.24</u>	80.97	66.93	79.96
Employee Remuneration	65.60	54.13	62.95	61.19	66.22	55.52	63.83	61.89	65.79	56.71	61.65	61.49
Employee Salaries	91.03	74.90	74.92	89.10	90.86	75.54	54.72	89.57	<u>91.58</u>	79.64	88.88	89.97
FIFA 22 Players	89.22	73.91	86.22	82.55	86.74	67.56	86.19	79.42	88.29	73.19	85.42	83.24
FilmTV Movies	47.65	41.79	44.09	43.64	48.51	42.35	46.07	44.17	48.30	43.76	44.77	43.94
Japanese Anime	67.63	63.49	59.07	65.66	65.80	54.47	58.78	63.00	66.95	48.04	59.32	63.64
Journal JCR	68.44	41.21	54.33	48.97	67.05	36.91	54.55	47.94	67.90	27.78	54.39	48.60
Journal SJR	<u>85.03</u>	44.33	73.15	80.68	84.22	42.43	72.09	80.02	84.07	43.85	70.69	79.78
K-Drama	54.96	51.25	47.60	55.01	53.42	54.09	34.56	52.47	53.79	52.38	44.46	50.99
ML/DS Salaries	47.30	46.21	25.09	46.90	47.75	46.31	24.22	45.75	47.78	46.86	37.67	47.35
Movies	**73.78**	52.96	67.49	68.29	72.43	48.25	66.94	66.99	72.38	46.76	68.12	67.45
Museums	71.53	55.68	61.77	59.70	71.98	51.94	61.01	57.67	71.66	55.75	57.47	58.58
MyDramaList	57.96	51.07	55.79	56.66	57.04	47.07	52.88	53.97	58.10	50.38	50.66	54.82
Prescription Drugs	86.06	79.06	86.02	84.37	85.95	78.42	86.08	83.73	85.88	77.57	85.65	84.15
Rotten Tomatoes	<u>57.22</u>	46.98	52.01	47.94	53.40	45.20	48.02	45.93	55.62	46.56	50.43	46.95
US Accidents Counts	38.17	22.68	29.99	21.35	38.46	20.41	30.36	22.90	38.72	22.83	28.70	22.02
US Presidential	77.22	61.79	46.82	60.84	77.14	62.63	52.26	61.21	77.07	62.53	64.37	61.21
Used Cars 24	89.94	87.92	86.55	87.71	89.53	88.58	73.33	88.01	89.37	88.57	84.54	88.08
Used Cars Benz Italy	92.48	88.53	90.84	91.64	92.07	89.72	90.41	91.66	92.44	89.55	92.10	91.99
Used Cars Pakistan	91.96	91.55	84.62	91.30	91.95	91.83	86.84	91.42	91.86	92.26	91.33	91.55
Used Cars Saudi Arabia	64.62	62.91	60.88	61.97	63.38	63.27	39.19	61.08	63.89	62.90	63.22	61.53
UsedCars.Com	88.39	67.53	86.17	83.59	88.55	70.81	88.15	83.90	87.77	77.38	87.49	83.23
Videogame Sales	57.12	41.06	44.55	33.41	57.19	40.67	45.02	33.78	57.20	42.99	41.48	32.84
Vivino Prices	83.22	77.48	79.08	77.51	83.40	77.15	79.12	77.15	83.28	77.68	78.36	77.44
Vivino Ratings	59.75	59.41	59.10	59.97	60.26	59.84	58.94	59.61	60.13	60.13	58.78	60.03
WikiLIQ Beer	68.91	61.65	64.35	64.88	<u>70.20</u>	61.65	66.60	64.90	69.46	61.73	67.03	64.55
WikiLIQ Spirit	64.91	50.91	60.19	53.79	64.84	51.03	60.59	54.18	63.69	51.80	57.88	53.11
Wina Poland	76.05	65.45	60.62	67.11	76.60	63.98	59.19	67.36	75.46	68.62	56.31	64.92
Wine.com Prices	83.29	73.51	79.33	74.62	82.59	73.05	78.75	72.99	82.40	73.67	78.72	72.84
Wine.com Ratings	27.45	24.23	27.42	26.77	27.42	19.89	26.45	27.02	26.69	23.86	25.60	26.14
WineEnthusiasts Prices	64.77	55.34	62.45	58.75	64.14	54.83	62.27	59.15	62.77	56.62	60.84	59.37
WineEnthusiasts Ratings	74.79	49.36	69.31	58.39	74.41	50.02	69.85	58.78	72.73	51.75	68.92	58.01

Table H.10: **Regression results for CARTE datasets (part 2 of 3).** Raw R^2 scores (clipped at 0 if negative) multiplied by 100, averaged over 5 random seeds for a train size of 1,024 samples, for TabPFN-3.5, EXAONE-Tabular and Causilo, each fed with TILE, CASE, the cell-wise LLM encoder (Llama-3.1-8B) or TF-IDF. Best in bold, second best underlined, both computed over all 35 methods.

Method	**TabPFN-3.5**				**EXAONE-Tabular**				**Causilo**			
String encoding	TILE	CASE	LLM encoder	TF-IDF	TILE	CASE	LLM encoder	TF-IDF	TILE	CASE	LLM encoder	TF-IDF
Anime Planet	64.06	50.64	59.31	55.22	64.54	48.39	58.93	55.93	**64.55**	48.04	59.41	55.42
Babies R Us	82.90	80.86	73.11	81.67	82.48	78.95	73.90	79.73	83.73	80.35	70.64	81.19
Beer Ratings	60.54	56.35	58.86	56.52	**61.62**	53.08	59.23	55.36	61.52	53.37	58.87	54.57
BikeDekho	79.56	79.95	74.81	78.85	79.84	78.16	73.78	77.90	79.71	79.28	70.28	79.11
BikeWale	80.20	76.53	79.00	76.44	**80.96**	78.82	78.25	75.90	80.84	79.61	77.80	75.73
Buy Buy Baby	**82.58**	81.50	80.37	78.66	82.31	79.25	78.25	79.13	82.08	80.08	78.03	79.26
CLEAR Corpus	65.90	63.49	69.81	61.77	66.05	63.11	**70.22**	60.79	66.93	63.04	69.35	60.84
CarDekho	91.92	87.77	89.61	92.00	93.06	90.54	91.07	92.70	**93.15**	90.60	84.88	92.36
Company Employees	**84.41**	81.03	31.84	79.84	83.94	80.61	16.20	80.09	84.08	80.91	49.74	80.13
Employee Remuneration	66.64	60.82	65.52	63.24	66.43	59.60	63.79	62.29	66.18	60.81	65.00	63.27
Employee Salaries	91.56	85.70	80.65	89.94	91.20	89.30	82.83	89.99	**91.59**	89.78	82.02	90.26
FIFA 22 Players	89.31	78.31	**91.04**	89.15	89.91	81.96	89.96	87.58	89.80	82.27	89.11	85.17
FilmTV Movies	48.76	44.18	45.98	45.07	48.89	43.19	45.61	43.75	**48.93**	42.87	46.42	44.45
Japanese Anime	67.73	64.34	61.75	66.16	**68.70**	65.39	65.44	66.55	68.07	65.11	61.79	65.44
Journal JCR	68.46	39.90	56.15	50.53	**69.43**	48.09	55.85	50.02	68.58	43.87	56.46	50.00
Journal SJR	84.76	56.78	73.31	81.92	84.99	70.18	70.35	81.04	**85.33**	69.59	71.45	81.47
K-Drama	55.32	52.26	49.84	52.74	55.42	52.23	47.90	51.90	56.96	53.18	51.27	55.00
ML/DS Salaries	**47.90**	47.22	24.49	47.39	47.81	46.45	27.22	47.62	47.04	46.27	30.44	47.56
Movies	73.12	51.49	68.38	68.27	73.75	66.50	68.01	68.52	73.75	65.45	68.77	68.11
Museums	72.18	57.11	62.21	61.63	71.88	56.70	63.60	60.75	**72.25**	58.03	62.85	60.13
MyDramaList	59.16	52.68	54.82	56.78	**59.84**	51.50	57.30	58.06	59.20	51.55	56.56	57.10
Prescription Drugs	86.51	81.31	87.19	85.50	86.77	83.29	86.55	85.27	86.14	82.59	**87.39**	85.06
Rotten Tomatoes	55.98	48.87	51.88	47.36	**58.46**	50.08	52.79	48.72	57.03	49.25	51.65	47.82
US Accidents Counts	**39.75**	29.19	30.41	24.59	39.26	29.40	30.34	23.77	39.27	30.20	30.63	24.50
US Presidential	77.83	62.94	34.88	61.60	76.97	61.43	30.27	61.21	**77.90**	63.16	43.98	61.99
Used Cars 24	89.63	**90.56**	86.04	88.88	89.89	89.92	86.50	88.54	89.92	90.30	84.03	88.58
Used Cars Benz Italy	92.41	91.71	92.18	92.23	**92.59**	90.96	92.15	92.18	92.49	90.82	92.32	92.20
Used Cars Pakistan	92.13	92.52	91.50	91.81	92.38	92.20	90.97	91.75	92.28	**92.62**	90.98	91.93
Used Cars Saudi Arabia	64.96	64.95	59.36	**65.65**	64.70	63.55	60.93	63.11	64.37	64.30	63.21	63.26
UsedCars.Com	88.20	82.08	88.14	84.38	**89.34**	85.05	88.75	84.64	88.93	85.39	88.35	84.25
Videogame Sales	57.87	43.69	42.97	36.35	58.06	43.45	46.41	34.35	**58.96**	43.52	45.82	34.77
Vivino Prices	83.73	79.21	78.81	78.31	83.74	78.02	80.35	78.41	**83.81**	78.74	80.05	78.71
Vivino Ratings	**61.04**	60.41	58.75	60.97	60.74	60.45	60.76	60.79	60.75	60.25	59.90	60.86
WikiLIQ Beer	70.15	62.83	66.20	64.82	70.00	62.50	66.54	65.36	**70.66**	62.51	66.46	65.70
WikiLIQ Spirit	65.04	56.39	60.94	56.05	**66.65**	53.02	61.73	55.39	66.31	53.70	62.52	55.88
Wina Poland	77.12	72.03	67.91	69.48	77.66	68.79	67.07	69.49	**78.54**	68.18	58.54	69.42
Wine.com Prices	83.10	75.99	79.44	74.99	83.63	73.74	80.58	75.18	**83.69**	74.84	80.85	74.90
Wine.com Ratings	27.63	24.68	26.29	27.14	27.91	26.78	**28.84**	27.72	27.45	26.58	27.97	27.78
WineEnthusiasts Prices	64.63	57.69	62.70	59.74	**65.28**	56.39	63.40	59.83	65.14	56.92	63.30	59.53
WineEnthusiasts Ratings	74.69	52.07	69.88	60.50	**75.97**	52.16	70.99	60.35	75.42	51.94	71.06	60.01

Table H.11: **Regression results for CARTE datasets (part 3 of 3).** Raw R^2 scores (clipped at 0 if negative) multiplied by 100, averaged over 5 random seeds for a train size of 1,024 samples, for the remaining learners, fed with the cell-wise LLM encoder (Llama-3.1-8B) or TF-IDF, and the methods with internal string handling. Best in bold, second best underlined, both computed over all 35 methods.

Method	**XGBoost (tuned)**		**CatBoost**			**RealMLP (tuned)**		**Ridge**		**ContextTab**	**TabSTAR**
String encoding	LLM encoder	TF-IDF	LLM encoder	TF-IDF	internal	LLM encoder	TF-IDF	LLM encoder	TF-IDF	internal	internal
Anime Planet	55.91	50.86	50.86	50.23	45.05	54.56	50.39	56.03	47.90	53.09	55.79
Babies R Us	55.41	74.83	51.23	74.65	54.22	58.82	77.73	58.43	62.45	74.28	**85.33**
Beer Ratings	53.63	50.72	53.82	50.51	50.76	54.48	49.88	49.15	40.39	56.77	36.91
BikeDekho	66.46	75.47	64.53	76.17	65.78	68.16	75.24	69.76	38.79	77.25	76.17
BikeWale	73.11	70.53	69.96	70.60	55.43	70.22	72.23	70.15	47.20	71.47	78.68
Buy Buy Baby	77.45	76.98	67.67	76.47	66.89	77.52	75.96	68.41	62.21	78.38	73.73
CLEAR Corpus	64.09	57.32	60.21	57.36	55.88	66.37	58.07	66.34	59.09	56.90	63.97
CarDekho	77.85	88.59	66.87	88.90	84.82	81.37	89.91	87.76	86.81	90.36	88.43
Company Employees	12.54	79.04	16.70	77.50	74.08	6.93	78.44	79.08	15.91	79.96	77.65
Employee Remuneration	59.94	59.02	59.38	58.56	39.50	59.50	57.19	51.40	43.19	57.53	**68.46**
Employee Salaries	35.92	87.42	41.27	87.75	82.35	86.43	88.05	83.69	80.57	88.44	88.45
FIFA 22 Players	79.45	77.12	76.97	76.25	78.41	77.76	75.29	68.48	45.04	81.85	78.13
FilmTV Movies	41.64	41.29	40.42	40.08	40.33	40.46	39.02	34.82	39.28	44.11	39.97
Japanese Anime	60.12	60.78	56.68	61.86	60.96	57.61	60.11	57.90	54.08	64.26	59.32
Journal JCR	49.40	45.83	47.09	44.13	39.03	48.99	43.54	43.56	32.09	46.98	44.49
Journal SJR	70.13	77.70	68.32	78.17	58.99	70.29	79.02	72.60	75.91	76.14	80.16
K-Drama	41.70	43.02	33.25	43.55	34.90	44.95	47.56	41.71	45.94	49.24	**57.83**
ML/DS Salaries	22.69	46.44	19.41	46.74	43.17	18.19	46.82	46.53	46.36	46.85	45.67
Movies	65.38	64.52	58.11	63.89	64.04	61.07	59.66	55.45	43.29	61.79	62.64
Museums	52.68	57.09	52.43	56.09	43.25	58.89	54.93	56.23	0.00	56.32	55.72
MyDramaList	50.52	50.38	49.21	51.06	47.51	43.01	47.77	25.68	30.33	46.33	48.87
Prescription Drugs	84.66	83.55	83.74	82.51	77.98	85.44	83.11	77.24	60.63	83.32	80.37
Rotten Tomatoes	43.48	41.28	44.86	43.14	37.05	44.35	39.94	36.08	30.99	39.65	54.43
US Accidents Counts	27.46	20.40	26.99	19.61	20.32	27.66	19.54	28.48	14.68	27.02	16.76
US Presidential	38.87	60.19	35.68	59.98	57.01	19.24	59.28	67.59	59.13	62.29	58.43
Used Cars 24	77.48	85.73	72.03	85.65	77.74	73.15	85.74	85.73	78.79	87.07	85.23
Used Cars Benz Italy	89.48	89.66	82.00	89.61	86.51	87.98	90.33	89.94	89.08	90.08	89.78
Used Cars Pakistan	86.37	90.04	69.29	90.06	88.41	80.01	90.26	87.86	87.71	90.22	89.79
Used Cars Saudi Arabia	56.26	58.54	49.19	58.56	56.08	48.62	58.17	51.86	47.05	62.19	58.60
UsedCars.Com	85.77	80.53	82.10	80.81	75.86	86.11	81.61	82.84	72.74	82.27	81.90
Videogame Sales	42.43	33.36	39.48	32.64	27.52	39.92	29.68	40.49	27.71	35.20	36.64
Vivino Prices	76.87	75.26	71.34	76.70	74.20	77.50	75.79	74.07	71.02	78.87	73.96
Vivino Ratings	57.30	57.55	52.28	58.12	56.65	56.05	57.14	53.89	23.84	58.59	57.42
WikiLIQ Beer	64.38	64.32	52.77	63.45	48.91	62.31	64.14	62.35	63.70	63.79	65.25
WikiLIQ Spirit	56.91	50.92	55.82	50.35	46.01	59.16	51.62	53.69	48.36	53.91	51.98
Wina Poland	58.28	63.32	52.09	62.52	61.44	45.13	64.86	64.00	60.15	67.35	63.62
Wine.com Prices	75.11	71.06	60.55	71.12	67.58	70.74	71.34	58.52	67.27	72.46	69.05
Wine.com Ratings	24.93	24.95	21.87	24.21	20.57	23.66	24.55	16.36	16.67	26.40	22.28
WineEnthusiasts Prices	57.33	56.19	57.44	56.77	52.91	59.31	56.11	53.64	53.40	56.26	52.27
WineEnthusiasts Ratings	63.84	53.90	64.19	54.56	38.36	67.30	55.80	64.50	49.01	46.60	66.85

Table H.12: **Regression results for WikiDBs datasets (part 1 of 3).** Raw R^2 scores (clipped at 0 if negative) multiplied by 100, averaged over 5 random seeds for a train size of 1,024 samples, for TabPFN-2.5, TabICLv2 and TabPFN-3, each fed with TILE, CASE, the cell-wise LLM encoder (Llama-3.1-8B) or TF-IDF. Best in bold, second best underlined, both computed over all 35 methods.

Method	**TabPFN-2.5**				**TabICLv2**				**TabPFN-3**			
String encoding	TILE	CASE	LLM encoder	TF-IDF	TILE	CASE	LLM encoder	TF-IDF	TILE	CASE	LLM encoder	TF-IDF
Artworks Inventory	32.92	27.13	25.01	28.71	33.64	28.18	26.49	29.11	33.16	27.76	28.13	30.01
Business Locations	86.13	77.36	78.46	72.02	82.96	66.91	71.78	61.73	85.88	78.16	77.07	70.58
Dissolved Municipalities	62.71	37.54	56.63	54.56	62.04	46.09	56.10	53.42	62.17	41.34	56.60	53.92
Drawings Catalog	80.50	80.23	80.49	80.58	80.70	79.93	80.44	80.24	80.78	80.60	79.95	80.53
Eclipsing Binary Stars	64.01	27.11	62.06	60.07	63.28	0.00	60.14	58.61	63.49	13.68	61.97	59.87
Geopolitical Regions	66.65	53.63	55.40	50.08	64.23	28.50	49.97	45.14	62.49	52.32	51.89	45.54
Historical Figures	78.76	0.00	68.04	55.68	78.50	30.34	67.60	55.87	77.82	0.00	68.11	55.79
Municipal District Capitals	83.32	75.75	75.48	69.36	80.90	73.15	69.83	64.90	81.44	76.40	69.56	67.03
Poets	88.11	0.00	85.11	79.12	87.90	0.00	85.78	77.88	88.16	0.00	85.54	79.57
Registered Ships	94.63	93.86	93.72	93.87	94.42	94.06	93.82	93.92	94.91	93.62	94.10	94.00
Research Article Citations	**82.64**	53.38	66.73	45.29	81.92	48.19	62.93	42.53	81.27	53.47	59.97	41.63
Research Articles	98.80	98.79	98.76	98.74	98.68	98.04	98.16	97.69	98.98	98.85	98.77	98.80
Territorial Entities	87.22	77.09	78.46	74.15	85.57	75.45	74.89	69.38	85.08	77.75	76.32	71.44
Ukrainian Villages	77.71	79.15	76.96	78.23	70.74	75.86	76.26	77.01	73.82	**79.84**	79.02	78.92
WWI Personnel	39.55	1.89	32.51	20.10	40.04	3.94	31.92	22.36	40.92	1.33	31.80	18.53

Table H.13: **Regression results for WikiDBs datasets (part 2 of 3).** Raw R^2 scores (clipped at 0 if negative) multiplied by 100, averaged over 5 random seeds for a train size of 1,024 samples, for TabPFN-3.5, EXAONE-Tabular and Causilo, each fed with TILE, CASE, the cell-wise LLM encoder (Llama-3.1-8B) or TF-IDF. Best in bold, second best underlined, both computed over all 35 methods.

Method	**TabPFN-3.5**				**EXAONE-Tabular**				**Causilo**			
String encoding	TILE	CASE	LLM encoder	TF-IDF	TILE	CASE	LLM encoder	TF-IDF	TILE	CASE	LLM encoder	TF-IDF
Artworks Inventory	**33.75**	29.08	26.60	29.91	33.20	29.10	31.10	30.19	33.12	30.11	29.95	30.17
Business Locations	**86.70**	79.49	78.79	72.92	85.97	74.44	78.30	71.16	85.76	70.23	77.58	69.15
Dissolved Municipalities	63.24	46.76	58.34	55.88	63.03	54.18	58.66	55.69	**64.00**	53.60	57.85	55.37
Drawings Catalog	81.25	**81.68**	81.26	81.23	81.16	81.22	80.85	81.50	81.34	81.30	80.84	81.02
Eclipsing Binary Stars	63.61	57.36	61.82	59.37	64.13	61.45	61.96	60.47	**64.14**	60.42	62.34	60.62
Geopolitical Regions	65.83	56.56	56.39	50.28	**67.23**	36.73	56.05	53.43	65.41	34.45	56.42	51.06
Historical Figures	79.25	0.00	67.34	61.55	78.71	50.70	71.37	57.04	**79.78**	52.54	69.85	58.07
Municipal District Capitals	81.81	76.99	73.57	69.36	**83.40**	75.20	76.49	71.05	83.15	74.99	76.50	69.91
Poets	88.68	0.00	85.10	82.09	88.15	52.73	85.51	80.14	**88.88**	42.94	85.68	80.46
Registered Ships	**95.37**	94.52	94.55	94.21	94.54	93.91	94.38	94.22	95.17	94.27	94.49	94.49
Research Article Citations	82.37	67.88	66.90	58.83	82.31	50.65	72.28	51.62	82.50	53.25	66.10	46.85
Research Articles	**99.07**	98.74	98.68	98.83	98.89	98.82	98.87	98.87	99.01	98.62	98.92	98.85
Territorial Entities	86.20	79.15	77.49	74.35	**87.61**	77.34	78.60	74.02	87.35	78.23	78.25	73.48
Ukrainian Villages	74.31	78.49	78.51	78.37	77.26	79.10	77.88	78.87	74.61	79.62	79.22	79.34
WWI Personnel	**41.62**	1.75	34.20	25.18	39.15	12.53	33.37	22.20	41.51	10.96	33.39	23.18

Table H.14: **Regression results for WikiDBs datasets (part 3 of 3).** Raw R^2 scores (clipped at 0 if negative) multiplied by 100, averaged over 5 random seeds for a train size of 1,024 samples, for the remaining learners, fed with the cell-wise LLM encoder (Llama-3.1-8B) or TF-IDF, and the methods with internal string handling. Best in bold, second best underlined, both computed over all 35 methods.

Method	**XGBoost (tuned)**		**CatBoost**			**RealMLP (tuned)**		**Ridge**		**ContextTab**	**TabSTAR**
String encoding	LLM encoder	TF-IDF	LLM encoder	TF-IDF	internal	LLM encoder	TF-IDF	LLM encoder	TF-IDF	internal	internal
Artworks Inventory	28.24	26.32	15.49	21.32	24.82	18.26	19.18	19.36	10.86	28.66	28.23
Business Locations	74.57	70.45	73.63	69.25	69.78	75.22	65.22	72.31	51.89	72.30	66.10
Dissolved Municipalities	50.72	52.56	52.25	53.27	46.94	53.50	49.12	45.22	43.26	56.92	50.97
Drawings Catalog	78.93	78.89	67.25	78.05	78.29	73.80	76.87	69.62	65.16	78.87	75.74
Eclipsing Binary Stars	59.42	58.34	57.96	57.42	48.29	57.29	57.43	60.11	50.47	60.81	55.52
Geopolitical Regions	49.99	46.68	50.20	48.47	48.02	46.47	43.14	41.04	23.83	50.60	49.05
Historical Figures	65.34	50.12	57.95	49.96	41.08	65.16	52.18	60.85	45.76	70.82	69.84
Municipal District Capitals	68.70	65.77	66.71	64.47	59.40	69.61	62.02	67.39	54.97	65.95	64.53
Poets	82.41	71.43	75.37	70.65	34.29	83.94	76.49	79.81	53.91	82.96	84.34
Registered Ships	92.90	93.35	89.53	91.83	92.73	91.42	92.41	87.65	87.92	94.53	92.19
Research Article Citations	60.05	39.96	55.06	41.70	41.16	65.70	40.83	62.01	29.96	73.47	49.76
Research Articles	98.23	98.32	98.42	98.48	98.52	92.72	92.73	79.31	78.89	98.23	98.39
Territorial Entities	75.10	71.56	74.17	70.51	70.98	74.45	67.63	71.43	58.24	73.03	67.08
Ukrainian Villages	69.91	69.78	56.67	69.33	73.68	66.40	68.27	39.30	39.83	63.71	67.18
WWI Personnel	33.38	17.74	27.35	16.45	6.53	29.79	14.92	27.06	17.18	34.71	26.92

Table H.15: **Classification results for TextTabBench datasets (part 1 of 3).** Raw ROC AUC scores multiplied by 100, averaged over 5 random seeds for a train size of 1,024 samples, for TabPFN-2.5, TabICLv2 and TabPFN-3, each fed with TILE, CASE, the cell-wise LLM encoder (Llama-3.1-8B) or TF-IDF. Best in bold, second best underlined, both computed over all 36 methods.

Method	**TabPFN-2.5**				**TabICLv2**				**TabPFN-3**			
String encoding	TILE	CASE	LLM encoder	TF-IDF	TILE	CASE	LLM encoder	TF-IDF	TILE	CASE	LLM encoder	TF-IDF
Customer Complaints	89.05	88.61	89.23	89.41	89.22	88.57	88.90	89.19	89.12	88.60	88.33	88.89
Diabetes	97.72	97.70	97.69	97.72	97.61	97.63	97.70	97.77	97.89	97.79	97.89	97.87
Job Frauds	99.12	97.82	98.69	98.73	99.10	97.32	98.99	98.76	99.04	97.43	98.59	98.65
Kickstarter	86.70	78.65	85.05	84.67	86.77	85.33	85.18	84.91	87.10	79.64	84.62	84.29
Lending Club	69.94	69.48	69.20	69.16	70.22	69.64	69.01	69.26	70.36	69.48	68.98	68.52
OSHA Accidents	59.26	58.84	58.54	59.28	**60.62**	57.77	59.12	59.32	60.07	58.77	58.04	59.25
Spotify	98.37	94.87	97.83	95.92	98.25	95.34	97.66	95.96	98.36	94.79	97.54	95.55

Table H.16: **Classification results for TextTabBench datasets (part 2 of 3).** Raw ROC AUC scores multiplied by 100, averaged over 5 random seeds for a train size of 1,024 samples, for TabPFN-3.5, EXAONE-Tabular and Causilo, each fed with TILE, CASE, the cell-wise LLM encoder (Llama-3.1-8B) or TF-IDF. Best in bold, second best underlined, both computed over all 36 methods.

Method	**TabPFN-3.5**				**EXAONE-Tabular**				**Causilo**			
String encoding	TILE	CASE	LLM encoder	TF-IDF	TILE	CASE	LLM encoder	TF-IDF	TILE	CASE	LLM encoder	TF-IDF
Customer Complaints	89.29	89.06	88.80	89.48	89.12	89.17	89.65	**89.86**	89.28	88.92	89.08	89.45
Diabetes	97.90	97.90	97.89	**98.01**	97.93	97.82	97.87	97.92	97.92	97.85	97.87	97.90
Job Frauds	99.16	99.04	98.85	98.92	99.15	98.78	99.04	98.79	**99.22**	98.71	98.70	98.81
Kickstarter	**87.46**	86.07	86.15	85.93	86.61	86.03	86.23	85.48	87.32	85.33	85.62	85.03
Lending Club	**70.65**	69.68	69.96	69.31	69.76	69.71	69.43	69.43	69.89	69.74	68.99	69.27
OSHA Accidents	60.40	59.19	58.16	59.85	59.46	59.85	58.66	58.84	60.28	60.06	58.55	59.32
Spotify	98.58	97.40	98.15	96.99	98.45	96.17	98.00	96.90	**98.59**	96.49	98.06	96.26

Table H.17: **Classification results for TextTabBench datasets (part 3 of 3).** Raw ROC AUC scores multiplied by 100, averaged over 5 random seeds for a train size of 1,024 samples, for the remaining learners, fed with the cell-wise LLM encoder (Llama-3.1-8B) or TF-IDF, and the methods with internal string handling; TabLLM has no result on two datasets (–). Best in bold, second best underlined, both computed over all 36 methods.

Method	**XGBoost (tuned)**		**CatBoost**			**RealMLP (tuned)**		**Ridge**		**ContextTab**	**TabSTAR**	**TabLLM**
String encoding	LLM encoder	TF-IDF	LLM encoder	TF-IDF	internal	LLM encoder	TF-IDF	LLM encoder	TF-IDF	internal	internal	internal
Customer Complaints	89.16	89.37	87.83	88.08	88.43	86.99	88.00	81.59	85.20	88.46	87.67	87.40
Diabetes	97.27	97.30	97.14	97.21	97.67	96.34	96.71	95.57	96.12	97.87	97.48	97.46
Job Frauds	97.63	97.56	98.21	98.10	98.27	98.51	98.18	96.49	96.92	98.92	97.13	–
Kickstarter	82.90	82.00	82.44	81.76	84.62	80.41	79.63	76.32	73.65	83.38	73.24	80.49
Lending Club	68.47	67.74	65.81	65.91	68.63	66.43	66.33	65.49	69.31	68.38	66.74	–
OSHA Accidents	57.67	57.97	54.52	57.37	56.84	54.69	57.82	58.06	58.35	57.99	57.94	58.76
Spotify	96.26	93.51	96.52	94.28	93.88	95.81	92.94	91.81	88.22	96.44	92.94	96.86

Table H.18: **Classification results for CARTE datasets (part 1 of 3).** Raw ROC AUC scores multiplied by 100, averaged over 5 random seeds for a train size of 1,024 samples, for TabPFN-2.5, TabICLv2 and TabPFN-3, each fed with TILE, CASE, the cell-wise LLM encoder (Llama-3.1-8B) or TF-IDF. Best in bold, second best underlined, both computed over all 36 methods.

Method	**TabPFN-2.5**				**TabICLv2**				**TabPFN-3**			
String encoding	TILE	CASE	LLM encoder	TF-IDF	TILE	CASE	LLM encoder	TF-IDF	TILE	CASE	LLM encoder	TF-IDF
Chocolate Bar Ratings	82.44	78.28	80.17	80.62	82.25	78.80	80.71	80.29	81.61	78.50	78.47	79.06
Coffee Ratings	93.21	84.68	91.81	93.48	92.72	85.27	92.45	93.44	92.31	84.67	90.85	92.51
Michelin	90.89	84.69	89.07	88.20	90.86	86.42	89.20	88.01	90.73	86.05	88.45	87.70
NBA Draft	90.59	89.38	89.25	89.05	90.66	89.32	88.99	88.98	90.87	89.37	88.90	88.89
Ramen Ratings	69.63	68.15	69.57	69.05	70.40	69.45	70.33	69.15	70.12	69.03	68.26	67.76
Roger Ebert	84.43	78.53	78.82	75.84	84.40	78.31	78.44	75.69	84.29	78.36	78.21	75.74
Spotify	**98.78**	94.94	95.42	90.57	98.55	94.95	95.55	90.23	98.50	94.86	94.76	90.37
US Accidents Severity	79.04	77.64	78.03	78.51	79.66	78.16	77.92	78.06	79.44	77.76	75.12	77.31
Whisky	88.13	86.52	86.99	86.50	88.37	86.72	87.20	86.63	88.30	86.70	85.22	86.07
Yelp	80.45	79.54	78.10	76.45	80.46	79.36	77.91	76.00	80.59	79.29	76.52	76.08
Zomato	72.72	70.94	71.40	71.39	73.04	71.27	71.04	70.90	72.61	70.82	70.56	70.31

Table H.19: **Classification results for CARTE datasets (part 2 of 3).** Raw ROC AUC scores multiplied by 100, averaged over 5 random seeds for a train size of 1,024 samples, for TabPFN-3.5, EXAONE-Tabular and Causilo, each fed with TILE, CASE, the cell-wise LLM encoder (Llama-3.1-8B) or TF-IDF. Best in bold, second best underlined, both computed over all 36 methods.

Method	**TabPFN-3.5**				**EXAONE-Tabular**				**Causilo**			
String encoding	TILE	CASE	LLM encoder	TF-IDF	TILE	CASE	LLM encoder	TF-IDF	TILE	CASE	LLM encoder	TF-IDF
Chocolate Bar Ratings	**82.85**	80.17	80.98	81.58	82.62	78.44	81.06	81.26	82.73	79.06	80.67	80.60
Coffee Ratings	93.00	87.04	92.57	93.58	93.23	85.97	91.91	93.73	93.26	85.51	92.39	93.37
Michelin	90.96	87.19	88.66	88.31	90.57	87.32	89.45	88.23	**91.01**	87.19	88.94	88.12
NBA Draft	90.97	89.25	89.21	88.99	90.68	89.08	89.17	89.06	**90.97**	89.37	89.32	89.06
Ramen Ratings	70.50	70.05	70.34	70.13	69.80	69.28	70.72	69.21	70.41	69.55	70.46	69.31
Roger Ebert	84.48	78.72	78.99	76.04	84.40	77.74	78.77	75.90	**84.60**	78.37	78.68	76.09
Spotify	98.75	95.61	96.24	91.71	98.59	95.04	96.43	92.04	98.74	95.24	95.96	90.71
US Accidents Severity	**80.05**	79.21	78.28	79.60	79.51	79.53	79.45	79.08	79.80	78.83	77.75	78.47
Whisky	88.43	86.96	85.10	86.75	88.03	86.56	86.96	86.63	**88.48**	86.81	86.83	86.62
Yelp	80.69	79.57	78.09	76.61	80.28	79.54	78.44	77.54	**80.82**	79.50	77.84	76.49
Zomato	**73.26**	72.64	71.03	71.85	72.87	71.94	70.57	71.79	73.20	71.53	70.88	71.46

Table H.20: **Classification results for CARTE datasets (part 3 of 3).** Raw ROC AUC scores multiplied by 100, averaged over 5 random seeds for a train size of 1,024 samples, for the remaining learners, fed with the cell-wise LLM encoder (Llama-3.1-8B) or TF-IDF, and the methods with internal string handling; TabLLM has no result on two datasets (–). Best in bold, second best underlined, both computed over all 36 methods.

Method	**XGBoost (tuned)**		**CatBoost**			**RealMLP (tuned)**		**Ridge**		**ContextTab**	**TabSTAR**	**TabLLM**
String encoding	LLM encoder	TF-IDF	LLM encoder	TF-IDF	internal	LLM encoder	TF-IDF	LLM encoder	TF-IDF	internal	internal	internal
Chocolate Bar Ratings	77.68	78.97	77.69	78.55	70.29	77.11	78.27	77.89	77.53	75.65	80.92	82.30
Coffee Ratings	90.30	91.88	90.42	92.34	74.94	90.90	92.25	89.29	92.38	82.71	93.75	**95.45**
Michelin	88.31	87.50	88.25	87.13	84.77	88.09	86.55	88.25	86.25	86.71	86.58	87.10
NBA Draft	88.13	88.25	86.71	85.89	88.17	87.87	87.81	86.39	88.04	88.27	88.19	85.90
Ramen Ratings	68.95	68.10	68.26	68.08	67.74	68.85	67.71	69.07	68.88	68.68	**79.11**	67.37
Roger Ebert	78.09	75.42	76.76	74.58	75.31	76.79	74.05	76.22	75.77	75.49	76.21	76.28
Spotify	94.22	88.92	94.39	89.85	90.13	93.53	89.06	94.52	86.66	93.93	91.72	96.27
US Accidents Severity	76.94	76.90	75.56	76.82	78.76	75.47	77.26	76.64	75.97	79.49	74.11	74.46
Whisky	81.41	85.52	84.07	85.29	84.95	75.74	86.18	86.42	86.53	86.47	85.10	84.10
Yelp	75.73	74.48	74.87	73.14	73.33	73.90	72.56	75.51	72.45	76.51	73.97	73.72
Zomato	69.39	70.47	69.64	70.25	67.16	66.34	70.02	71.07	71.71	71.27	67.27	68.57

Table H.21: **Classification results for WikiDBs datasets (part 1 of 3).** Raw ROC AUC scores multiplied by 100, averaged over 5 random seeds for a train size of 1,024 samples, for TabPFN-2.5, TabICLv2 and TabPFN-3, each fed with TILE, CASE, the cell-wise LLM encoder (Llama-3.1-8B) or TF-IDF. Best in bold, second best underlined, both computed over all 36 methods.

Method	**TabPFN-2.5**				**TabICLv2**				**TabPFN-3**			
String encoding	TILE	CASE	LLM encoder	TF-IDF	TILE	CASE	LLM encoder	TF-IDF	TILE	CASE	LLM encoder	TF-IDF
Artists Copyright	80.00	74.83	78.61	78.53	80.45	75.09	78.60	77.12	79.64	75.16	76.22	77.36
Artworks Catalog	97.53	97.23	97.21	97.31	97.59	97.38	97.37	97.09	97.30	97.29	97.00	96.99
CC Authors	99.75	99.44	99.31	91.77	99.78	99.28	99.33	92.86	99.74	99.43	99.14	92.29
Defenders	99.48	97.80	99.01	95.23	99.47	97.74	99.04	95.75	99.42	97.76	99.02	94.83
Forward Players	89.23	87.13	88.96	88.80	89.43	87.57	89.08	88.55	89.01	87.10	88.25	88.06
Geographers	99.54	98.38	99.12	93.62	99.60	98.49	99.12	93.95	**99.62**	98.46	99.07	92.08
Islands	99.95	99.98	99.91	99.97	99.97	99.99	99.97	99.97	99.97	99.99	99.97	99.97
Magic Narratives	71.05	67.76	66.95	68.04	72.36	69.14	70.41	68.97	69.59	67.10	65.38	65.65
Museums	99.98	99.97	99.97	99.94	99.98	99.98	99.98	99.93	99.98	99.98	99.98	99.93
Noble Individuals	83.20	79.19	80.84	80.09	83.41	79.49	82.83	80.78	82.97	77.36	80.73	79.67
Notable Trees	96.54	88.27	96.30	97.13	96.45	88.24	96.28	96.95	96.37	88.32	96.08	96.89
Philosophers	98.70	95.85	98.05	93.11	98.72	96.97	98.17	93.00	98.72	95.42	98.06	91.71
Sculptures	89.35	86.62	88.62	87.57	89.29	87.44	88.54	87.33	89.09	86.59	87.40	86.20
Spring Locations	99.98	99.98	99.99	99.98	99.99	99.97	99.99	99.99	99.99	99.98	99.99	99.98
State Schools	99.99	99.98	99.99	99.97	99.99	99.98	99.99	99.97	99.99	99.98	99.99	99.97
Transport Stations	99.98	99.94	99.94	99.86	99.99	99.94	99.97	99.86	99.98	99.96	99.96	99.82
US Music Albums	95.51	87.42	93.90	91.56	95.43	89.43	94.12	91.70	95.18	86.17	92.34	86.94

Table H.22: **Classification results for WikiDBs datasets (part 2 of 3).** Raw ROC AUC scores multiplied by 100, averaged over 5 random seeds for a train size of 1,024 samples, for TabPFN-3.5, EXAONE-Tabular and Causilo, each fed with TILE, CASE, the cell-wise LLM encoder (Llama-3.1-8B) or TF-IDF. Best in bold, second best underlined, both computed over all 36 methods.

Method	**TabPFN-3.5**				**EXAONE-Tabular**				**Causilo**			
String encoding	TILE	CASE	LLM encoder	TF-IDF	TILE	CASE	LLM encoder	TF-IDF	TILE	CASE	LLM encoder	TF-IDF
Artists Copyright	80.38	76.89	77.06	78.61	80.31	76.81	79.13	78.56	**80.74**	77.36	78.43	78.48
Artworks Catalog	97.66	97.72	97.57	97.68	97.73	97.67	97.56	97.63	**97.75**	97.66	97.48	97.45
CC Authors	**99.78**	99.54	99.38	93.28	99.72	99.48	99.28	93.04	99.77	99.46	99.31	93.02
Defenders	99.44	98.56	99.12	96.06	99.40	98.23	99.13	95.91	**99.51**	98.14	99.08	95.55
Forward Players	89.63	89.25	88.96	89.02	89.58	88.53	89.46	89.07	**89.71**	88.35	89.26	88.95
Geographers	99.61	98.83	98.97	95.45	99.55	98.73	99.11	94.17	99.62	98.67	99.07	94.12
Islands	99.98	**99.99**	99.98	99.98	99.97	99.99	99.98	99.98	99.98	99.99	99.97	99.98
Magic Narratives	72.03	71.10	70.74	69.80	73.17	69.67	70.78	69.86	72.76	71.67	71.27	69.15
Museums	**99.99**	99.98	99.98	99.96	99.98	99.98	99.97	99.95	99.99	99.98	99.98	99.95
Noble Individuals	83.38	80.50	82.93	80.86	83.41	81.44	82.44	80.90	**83.73**	82.03	82.72	81.12
Notable Trees	96.74	90.39	96.52	97.22	96.63	91.45	96.57	**97.35**	96.66	91.15	96.45	97.25
Philosophers	98.75	97.11	98.24	94.05	98.67	97.66	98.23	93.93	**98.77**	97.72	98.26	93.53
Sculptures	89.68	87.96	88.58	87.77	89.55	88.24	89.40	88.03	**89.81**	88.16	89.10	87.64
Spring Locations	100.00	100.00	**100.00**	99.99	99.99	99.99	99.99	99.99	100.00	99.99	99.99	99.99
State Schools	**99.99**	99.99	99.99	99.98	99.99	99.99	99.99	99.98	99.99	99.98	99.99	99.98
Transport Stations	99.99	99.96	99.97	99.87	99.98	99.96	99.96	99.88	**99.99**	99.95	99.97	99.87
US Music Albums	**95.80**	94.11	93.76	92.25	95.63	92.09	94.44	92.52	95.75	91.22	94.14	92.01

Table H.23: **Classification results for WikiDBs datasets (part 3 of 3).** Raw ROC AUC scores multiplied by 100, averaged over 5 random seeds for a train size of 1,024 samples, for the remaining learners, fed with the cell-wise LLM encoder (Llama-3.1-8B) or TF-IDF, and the methods with internal string handling; TabLLM has no result on two datasets (–). Best in bold, second best underlined, both computed over all 36 methods.

Method	**XGBoost (tuned)**		**CatBoost**			**RealMLP (tuned)**		**Ridge**		**ContextTab**	**TabSTAR**	**TabLLM**
String encoding	LLM encoder	TF-IDF	LLM encoder	TF-IDF	internal	LLM encoder	TF-IDF	LLM encoder	TF-IDF	internal	internal	internal
Artists Copyright	75.88	75.59	76.87	75.43	74.86	75.21	75.94	74.56	73.73	76.92	73.27	75.68
Artworks Catalog	96.38	95.91	96.94	96.57	93.54	95.90	96.35	93.31	93.92	96.82	93.45	96.25
CC Authors	98.48	87.91	97.96	87.95	83.69	98.96	85.61	98.99	80.54	97.86	98.85	99.29
Defenders	98.66	93.47	98.91	94.11	88.96	98.64	93.62	98.59	90.19	97.30	96.61	98.58
Forward Players	88.68	87.67	88.87	87.64	85.98	88.05	87.24	87.22	86.18	88.73	84.08	88.61
Geographers	98.70	91.71	98.94	92.79	86.14	99.09	91.41	99.19	88.40	95.70	96.58	98.94
Islands	99.89	99.89	99.87	99.86	99.93	99.44	99.47	98.22	97.14	99.88	99.44	99.89
Magic Narratives	68.95	67.29	69.87	66.47	53.51	66.07	67.01	68.54	65.67	68.20	62.87	**75.11**
Museums	99.95	99.84	99.92	99.83	99.51	99.94	99.80	99.21	97.55	99.79	99.65	99.72
Noble Individuals	78.90	78.02	80.67	78.22	63.49	79.75	79.04	77.55	76.86	80.06	79.01	80.98
Notable Trees	94.73	96.16	94.96	96.14	85.14	94.50	96.40	93.15	95.51	93.68	95.77	97.31
Philosophers	97.92	91.81	97.85	92.17	86.10	98.12	91.50	97.84	88.82	96.70	95.96	97.36
Sculptures	87.79	86.20	88.05	86.29	81.84	86.77	85.87	85.86	84.83	87.66	82.57	87.00
Spring Locations	99.96	99.95	99.97	99.96	99.97	99.93	99.92	98.60	98.59	99.84	99.86	99.79
State Schools	99.95	99.93	99.95	99.92	98.68	99.98	99.89	99.86	99.50	99.90	99.88	99.93
Transport Stations	99.89	99.73	99.89	99.72	99.14	99.86	99.73	99.71	98.71	99.89	99.68	99.65
US Music Albums	91.81	87.63	92.36	88.45	88.93	91.50	87.57	88.20	83.19	92.55	86.38	94.48

I MODELS, SOFTWARE AND LICENSES

TILE is an architecture rather than a model, and thus relies on existing, publicly released artifacts. We list here everything we used, with its source, so that our results can be reproduced.

Language models All backbones are downloaded from Hugging Face. From the Llama family (Grattafiori et al., 2024) we use Llama-3.1-70B and Llama-3.1-8B (`meta-llama/Llama-3.1-70B` and `meta-llama/Llama-3.1-8B`), under the Llama 3.1 Community License, and Llama-3.2-3B and Llama-3.2-1B (`meta-llama/Llama-3.2-3B` and `meta-llama/Llama-3.2-1B`), under the Llama 3.2 Community License. From the Qwen3 family (Yang et al., 2025) we use the three base checkpoints `Qwen/Qwen3-8B-Base`, `Qwen/Qwen3-4B-Base` and `Qwen/Qwen3-0.6B-Base`, released under the Apache 2.0 license. Llama-3.1-8B is the default backbone of TILE, and is also the model behind the cell-wise LLM encoder baseline.

Table foundation models and tabular learners TILE and the encoding baselines are evaluated on top of TabPFN-2.5 (Grinsztajn et al., 2025) and TabPFN-3 (Grinsztajn et al., 2026), from Prior Labs (`https://github.com/PriorLabs/TabPFN`), and TabICLv2 (Qu et al., 2026) (`https://github.com/soda-inria/tabicl`). The other learners are RealMLP (Holzmüller et al., 2024), used through `pytabkit` (`https://github.com/dholzmueller/pytabkit`), XGBoost (Chen & Guestrin, 2016), CatBoost (Prokhorenkova et al., 2018), and Ridge regression from `scikit-learn`.

End-to-end baselines ContextTab (Spinaci et al., 2025) is used through SAP's `sap-rpt-oss` release (`https://github.com/SAP-samples/contexttab`), TabSTAR (Arazi et al., 2025) through the `tabstar` package (`https://github.com/eyalbd2/TabSTAR`), and TabLLM (Hegselmann et al., 2023) through the authors' code together with the t-few recipe it builds on; we run it with T0-3B (`bigscience/T0_3B`).

Software The string-encoding baselines and all table preprocessing use `skrub` (`https://skrub-data.org`), whose `StringEncoder`, `LLMEncoder` and `TableVectorizer` we rely on throughout, together with `scikit-learn`, `sentence-transformers` for the cell-wise embeddings, and PyTorch and Hugging Face `transformers` for the LLM forward passes.

Apart from the Llama checkpoints, which carry the community licenses quoted above, every model and library listed here is distributed under a permissive open-source license (Apache 2.0, BSD-3-Clause or MIT). We use each of them for research purposes and in accordance with its terms.