TabLLM: Few-shot Classification of Tabular Data with Large Language Models

Hegselmann et al., AISTATS 2023

Presenter: *Haeyoung Lee*

December 02, 2025

Seoul National University

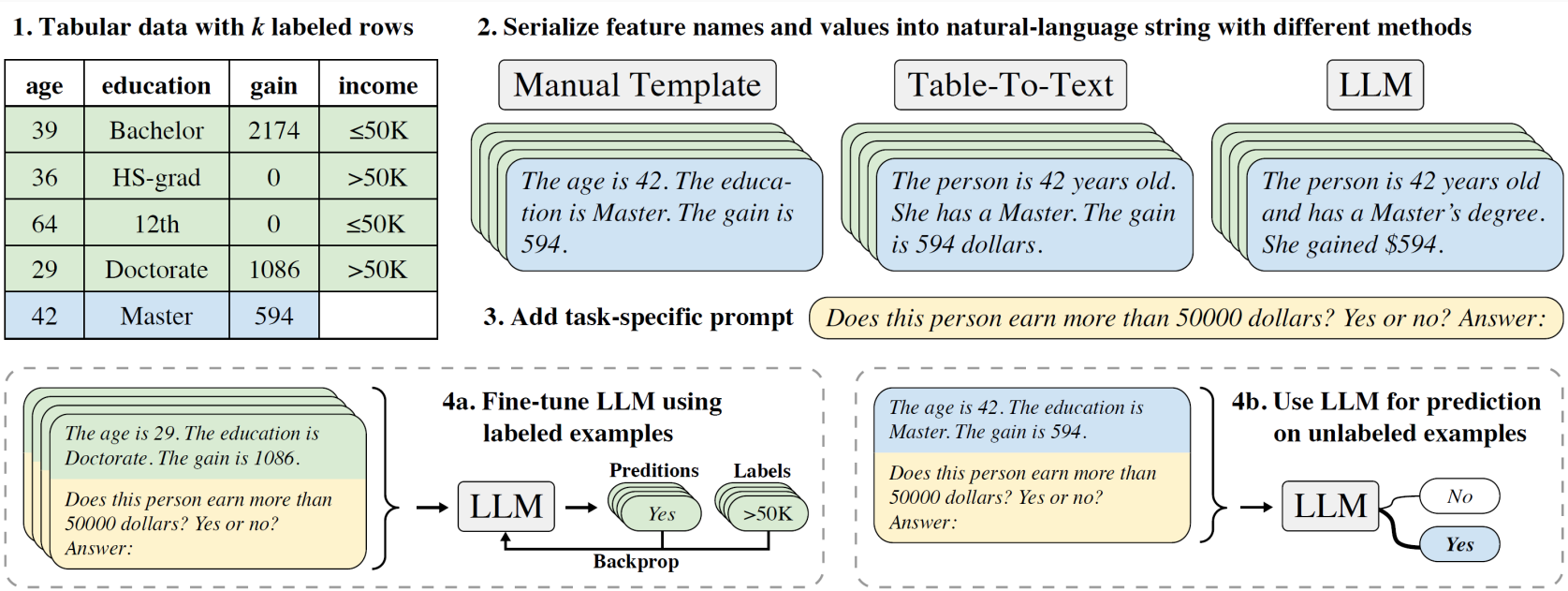
Motivation

- Real-world records are predominantly stored as *tabular* data across domains (e.g., healthcare, climate, finance), yet obtaining labels for supervised classification is difficult.
- Unlike in computer vision and NLP, deep learning has not consistently outperformed strong *gradient-boosted tree* ensembles on tabular data, which typically have weak locality, mixed feature types, and relatively few columns.
- LLMs pretrained on massive text corpora exhibit strong few-shot generalization across diverse domains and can achieve competitive performance with little or no labeled training data by leveraging the knowledge encoded in their parameters.
- **Key question:** Can we leverage an LLM for tabular prediction?
- **TabLLM** prompts an LLM with (i) a natural-language serialization of each table row and (ii) a short task description; multiple serialization strategies are explored, and zero-shot and few-shot classification is evaluated.

Notation

- Tabular dataset with n rows and d columns (features).
- x_i: a d-dimensional feature vector (the i-th row).
- $y_i \in C$: class label; C is the set of classes.
- $D = \{(x_i, y_i)\}_{i=1}^{n}$: dataset of row–label pairs.
- $F = \{f_1, \ldots, f_d\}$: feature/column names (e.g., "age", "education").
- k: number of labeled training examples.
- D_k: a subset of size k sampled from D (with replacement) used for fine-tuning.
- $\texttt{serialize}(F, x)$: a function that takes the column names F and feature values x for a row as inputs and creates a textual representation.
- p: task-specific prompt (short description/question appended after serialization).
- $\texttt{LLM}$: an large language model with vocabulary V.
- $\texttt{LLM}\big((\texttt{serialize}(F, x), p)\big)$: the prompted output of the LLM.
- Verbalizer: manually specified mapping from LLM output tokens (answer choices) to class labels in C.

Overview



- Steps:
  1. Serialize a row (F, x) into a natural language string.
  2. Append a task-specific prompt.
  3. Obtain output probabilities from the LLM for verbalizer tokens (e.g., "Yes" / "No").
  4. In few-shot setting, do parameter-efficient fine-tuning (T-Few) on k labeled examples.

From LLM Text to Class Label

Income Dataset:

```
answer_choices:  'No ||| Yes'
jinja:  '{{serialization}}

Does this person earn more than 50000
dollars per year?  Yes or no?
Answer:
|||
{{ answer_choices[label] }}'
```

Car Dataset:

```
answer_choices:  'Unacceptable |||
Acceptable ||| Good ||| Very good'
jinja:  '{{serialization}}

How would you rate the decision to buy
this car?  Unacceptable, acceptable,
good or very good?
Answer:
|||
{{ answer_choices[label] }}'
```

End Of Life Task:

```
answer_choices:  'No ||| Yes'
jinja:  '{{serialization}}

Does this patient die in the next nine
months?  Yes or no?
Answer:
|||
{{ answer_choices[label] }}'
```

- An LLM outputs a probability distribution over tokens, not class IDs.
- A **verbalizer** maps each class in C to a verbalizer token (e.g., $\leq 50\text{K} \mapsto \texttt{No}, \quad > 50\text{K} \mapsto \texttt{Yes}$).
- **Inference:** compute the normalized likelihood of each verbalizer answer and choose the most probable class.

9 Serialization Strategies

Income Dataset (List Template):

```
- Age: 30
- Race: Asian-Pac-Islander
- Sex: Female
- Marital status: never married
- Rel. to head of the household: Own
- Native country: Taiwan
- Occupation: execution and management
- Work class: private sector employee
- Capital gain last year: 0
- Capital loss last year: 0
- Education: bachelor's degree
- Work hours per week: 52
```

Income Dataset (Text Template):

```
The Age is 30. The Race is
Asian-Pac-Islander. The Sex is Female.
The Marital status is never married.
The Relation to head of the household is
Own-child. The Native country is
Taiwan. The Occupation is execution and
management. The Work class is private
sector employee. The Capital gain last
year is 0. The Capital loss last year
is 0. The Education is bachelor's
degree. The Work hours per week is 52.
```

Income Dataset (Table-To-Text):

```
The age was 30 years, and was the
youngest ever to enter the age. The
race was held in the Asian-Pac-Islander,
and was won by the race. The sex of the
village was Female. The first female to
be married is Marital status never
reported. the family has the head of
the household. The Chinese: native
region of Taiwan. He was the
executioners of the execution and
management of the city of New York City.
the private sector employee is a private
sector employee. The capital was
Capital of the State of India. The
capital loss of the state was 0.5%. The
bachelor's degree in Education was
bachelor's degree. the week 52 was the
52-hour week.
```

Income Dataset (Text T0):

```
Kim is a 30-year-old Asian-Pacific
Islander. She is never married and has
never had children. The man is the
owner of the house and he is the only
child. A woman is executing a contract
as a private sector employee. The
company had a capital loss of $ 0 last
year. The man has a bachelor's degree
and works 52 hours a week.
```

Income Dataset (Text GPT-3):

```
The person is 30 years old,
Asian-Pac-Islander, female, never
married, and an own-child relation to
the head of the household. The person
is from Taiwan and is an execution and
management occupation in the private
sector employee work class. The person
has 0 dollars in capital gain and 0
dollars in capital loss from the
previous year. The person has a
bachelor's degree and works 52 hours per
week.
```

- **Templates (manual):**
  - List Template: "feature: value, ..."
  - Text Template: "The feature is value., ..."
- **LLM-based generation:**
  - Table-to-Text: an LLM fine-tuned on a table-to-text generation task. Feed each `(column, value)` tuple separately and concatenate the generated outputs.
  - Text T0: use T0 with 11B params, split a row into *pairs* of two `(column, value)` tuples, prompt each pair with ``` ``Write this information as a sentence:'' ```, then combine the resulting sentences.
  - Text GPT-3: use GPT-3 via API; provide the *full list* of features at once with ``` ``Rewrite all list items in the input as a natural text.'' ```

9 Serialization Strategies

- **Ablations (based on the List Template):**
  - **List Only Values:** *remove column names* (values only) to test whether feature/column names contribute to performance.
  - **List Permuted Names:** *permute the column names* so that each value is paired with an incorrect name. (e.g., `sex:30, age:male`)

    Goal: test how important the correct name–value association is.
  - **List Permuted Values:** *permute values* within each column using a fixed, column-specific mapping applied to all examples.

    Categorical ex.: `Bachelor`→`PhD, Master`→`12th, ...`

    Continuous ex.: bin into 10 uniform ranges, then permute bins (e.g., age 20–30→70–80).

    Goal: test whether the LLM relies on fine-grained value information beyond column names.
  - **List Short:** keep *at most 10 features*; used for the healthcare dataset to satisfy the LLM input-length limit and test the effect of reduced information.

Experimental Setup: LLM and Fine-tuning

- **LLM: T0 (11B params)**; a T5-style encoder–decoder model that is further **instruction-tuned on a large variety of task-specific prompts**, which yields strong **zero-shot generalization** across diverse tasks.
  - Input budget: **1024 tokens** (roughly 400 words).
- **Few-shot fine-tuning:** Fine-tune on the k-shot set using the parameter-efficient **T-Few** recipe.
  - Public tabular datasets: **30 epochs** for all few-shot runs.
  - Healthcare dataset: fewer epochs for larger k (e.g., 10 epochs up to 256 shots; 3 epochs for 1,024+ shots) to reduce runtime and overfitting.
- **Zero-shot baseline: GPT-3** with no fine-tuning.

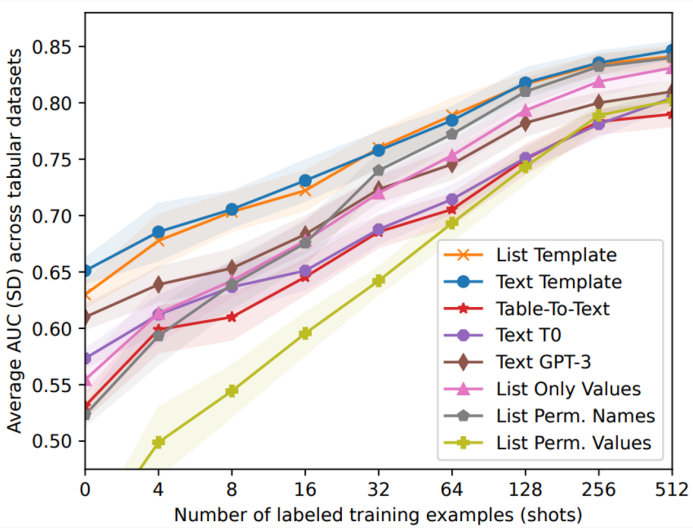
Experimental Setup: Datasets

- **9 Public tabular benchmarks :**
  - Requirements: $\leq$ 50,000 rows, $\leq$ 30 columns for cost and token budget of T0.
  - Require *textual* feature names; exclude datasets with *derived* features (e.g., mean pixel values).
  - Included datasets (rows, features):
    Bank (45,211, 16), Blood (748, 4), California (20,640, 8), Car (1,728, 8), Credit-g (1,000, 20), Income (48,842, 14), Jungle (44,819, 6), Diabetes (768, 8), Heart (918, 11).
- **Healthcare claims:**
  - 3 binary tasks: End-of-Life (EoL), Surgery, Likelihood of Hospitalization (LoH).

Baselines

- **Classical baselines:**
  - Logistic Regression (LR)
  - Gradient-boosted trees: XGBoost, LightGBM
- **Deep tabular baselines:**
  - TabNet (attention over columns), SAINT (attention over rows & columns), NODE (differentiable tree ensemble)
  - TabPFN (pretrained Bayesian NN on synthetic tabular data)
- **Details:**
  - Hyperparameter tuning for all baselines *except* TabPFN.
  - Few-shot setting: no separate validation set $\Rightarrow$ **4-fold CV** on the k shots for model selection.
  - Categorical features: **one-hot encoding** (ordinal encoding tested but worse).
- **Healthcare claims:** only LR + LightGBM (runtime constraints).
- **Metric:** Area under the curve (AUC).

Results: Effects of Serialization



Car Dataset (Text Template):

```
The Buying price is low.  The Doors is
three.  The Maintenance costs is low.
The Persons is more than four.  The
Safety score is medium.  The Trunk size
is medium.
```

Car Dataset (Text GPT-3):

```
This car a good choice for those who are
looking for a low-priced vehicle with
low maintenance costs.  It is also a
good choice for families or groups of
friends who need a car with a bit more
space than a smaller car.  The safety
score is medium, so it is not the best
choice for those who are looking for a
car with the highest safety rating.
```

- **Text Template** performs best overall.
- In the **zero-shot** setting, **Text Template** outperforms the List Template,
  - likely because its natural-sentence format is closer to T0's training distribution.
  - This gap largely vanishes by ∼**8 shots**, suggesting that sophisticated serializations might be unnecessary when some training data exists.
- **LLM-based serialization is less reliable:**
  - generated text may **hallucinate** additional content or **omit** features, which can introduce misleading cues and lead to lower performance than fixed templates.
- **Ablations converge with more data:**
  - `Only Values` and `Permuted Names` perform poorly at **zero / very few shots**, but **match the main templates** once enough training examples are provided.

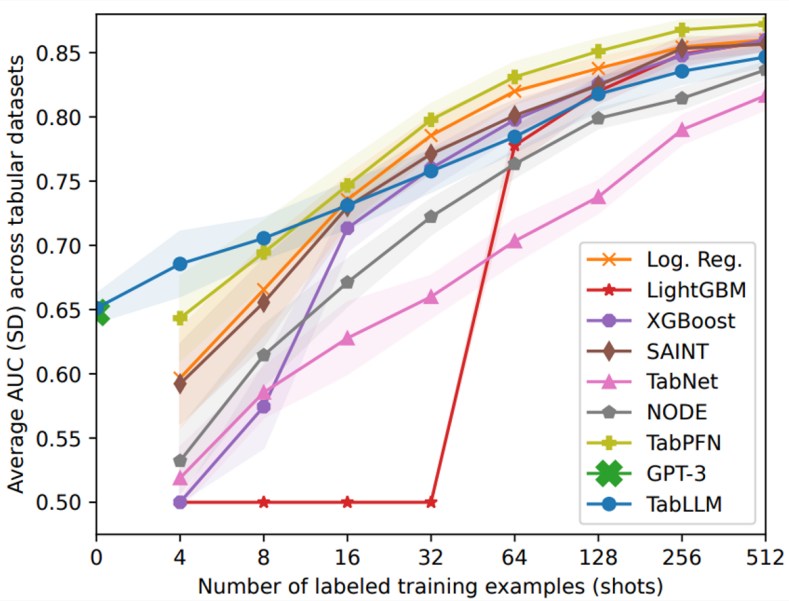
Results: Effects of Serialization

Method	Number of Shots							
	0	16	64	256	1,024	4,096	16,384	all
End of Life (EoL)								
TabLLM (T0 + List Template)	0.70	0.74	0.78	0.78	0.79	0.81	0.81	—
TabLLM (T0 + Text Template)	0.63	0.71	0.74	0.76	0.78	0.79	0.80	—
TabLLM (T0 + List Short)	0.68	0.71	0.76	0.79	0.80	0.81	0.82	—
TabLLM (T0 + List Perm. Names)	0.62	0.66	0.70	0.74	0.75	0.77	0.79	—

Method	EoL	Surgery	LoH
Age, sex, and race	0.59	0.57	0.65
Least frequent conditions	0.57	0.64	0.67
Least frequent procedures	0.59	0.59	0.65
Least frequent concepts (cond. + proc.)	0.55	0.55	0.66
Most frequent conditions	0.67	0.66	0.69
Most frequent procedures	0.59	0.58	0.65
Most frequent concepts (cond. + proc.)	0.62	0.61	0.65
Oldest conditions	0.65	0.66	0.69
Oldest procedures	0.59	0.58	0.65
Oldest concepts (cond. + proc.)	0.60	0.60	0.67
Most recent conditions	0.65	0.66	0.69
Most recent procedures	0.55	0.59	0.65
Most recent concepts (cond. + proc.)	0.59	0.60	0.66

- **Healthcare claims: List Template** slightly outperforms Text Template. **List Short** is only slightly worse, suggesting robustness to incomplete feature sets.
- **Claims-specific finding:** selecting the **most frequent conditions per patient** works best.

Results: TabLLM vs. Baselines



Dataset	Method	Number of Shots 0	4	8	16	32	64	128	256	512	all
Bank	XGBoost	—	$0.50_{.00}$	$0.56_{.09}$	$0.68_{.04}$	$0.76_{.03}$	$\mathbf{0.83}_{.02}$	$0.85_{.03}$	$0.88_{.01}$	$0.90_{.01}$	$\mathbf{0.94}_{.00}$
	TabPFN	—	$0.59_{.14}$	$\mathbf{0.66}_{.08}$	$\mathbf{0.69}_{.02}$	$0.76_{.03}$	$0.82_{.03}$	$\mathbf{0.86}_{.02}$	$\mathbf{0.89}_{.00}$	$0.90_{.00}$	$0.91_{.00}$
	TabLLM	$\mathbf{0.63}_{.01}$	$0.59_{.10}$	$0.64_{.05}$	$0.65_{.05}$	$0.64_{.06}$	$0.69_{.03}$	$0.82_{.05}$	$0.87_{.01}$	$0.88_{.01}$	0.92 †
Blood	XGBoost	—	$0.50_{.00}$	$0.58_{.07}$	$0.66_{.04}$	$0.67_{.06}$	$0.68_{.05}$	$0.71_{.06}$	$0.70_{.07}$	$0.67_{.06}$	$0.71_{.04}$
	TabPFN	—	$0.52_{.08}$	$0.64_{.04}$	$\mathbf{0.67}_{.01}$	$\mathbf{0.70}_{.04}$	$\mathbf{0.73}_{.04}$	$\mathbf{0.75}_{.04}$	$\mathbf{0.76}_{.04}$	$\mathbf{0.76}_{.03}$	$\mathbf{0.74}_{.03}$
	TabLLM	$\mathbf{0.61}_{.04}$	$\mathbf{0.58}_{.09}$	$\mathbf{0.66}_{.03}$	$0.66_{.07}$	$0.68_{.04}$	$0.68_{.04}$	$0.68_{.06}$	$0.70_{.08}$	$0.68_{.04}$	$0.70_{.04}$
Calhousing	XGBoost	—	$0.50_{.00}$	$0.62_{.10}$	$0.74_{.03}$	$0.79_{.04}$	$0.82_{.04}$	$0.87_{.01}$	$0.90_{.01}$	$0.92_{.01}$	$\mathbf{0.97}_{.00}$
	TabPFN	—	$0.63_{.13}$	$\mathbf{0.63}_{.11}$	$\mathbf{0.80}_{.03}$	$\mathbf{0.85}_{.03}$	$\mathbf{0.89}_{.01}$	$\mathbf{0.91}_{.01}$	$\mathbf{0.92}_{.00}$	$\mathbf{0.93}_{.00}$	$0.94_{.00}$
	TabLLM	$\mathbf{0.61}_{.01}$	$0.63_{.05}$	$0.60_{.07}$	$0.70_{.08}$	$0.77_{.08}$	$0.77_{.04}$	$0.81_{.02}$	$0.83_{.01}$	$0.86_{.02}$	$0.95_{.00}$
Car	XGBoost	—	$0.50_{.00}$	$0.59_{.04}$	$0.70_{.08}$	$0.82_{.03}$	$0.91_{.02}$	$0.95_{.01}$	$0.98_{.01}$	$0.99_{.01}$	$1.00_{.00}$
	TabPFN	—	$0.64_{.06}$	$0.75_{.05}$	$\mathbf{0.87}_{.04}$	$\mathbf{0.92}_{.02}$	$\mathbf{0.97}_{.00}$	$\mathbf{0.99}_{.01}$	$\mathbf{1.00}_{.00}$	$1.00_{.00}$	$1.00_{.00}$
	TabLLM	$\mathbf{0.82}_{.02}$	$\mathbf{0.83}_{.03}$	$\mathbf{0.85}_{.03}$	$0.86_{.03}$	$0.91_{.02}$	$0.96_{.02}$	$0.98_{.01}$	$0.99_{.00}$	$1.00_{.00}$	$1.00_{.00}$
Credit-g	XGBoost	—	$0.50_{.00}$	$0.51_{.07}$	$0.59_{.05}$	$0.66_{.03}$	$0.67_{.06}$	$0.68_{.02}$	$0.73_{.02}$	$0.75_{.03}$	$\mathbf{0.78}_{.04}$
	TabPFN	—	$0.58_{.08}$	$0.59_{.03}$	$0.64_{.06}$	$0.69_{.07}$	$0.70_{.07}$	$\mathbf{0.72}_{.06}$	$\mathbf{0.75}_{.04}$	$0.75_{.02}$	$0.75_{.03}$
	TabLLM	$\mathbf{0.53}_{.05}$	$\mathbf{0.69}_{.04}$	$\mathbf{0.66}_{.04}$	$\mathbf{0.66}_{.05}$	$\mathbf{0.72}_{.06}$	$0.70_{.07}$	$0.71_{.07}$	$0.72_{.03}$	$0.72_{.02}$	$0.70_{.02}$
Diabetes	XGBoost	—	$0.50_{.00}$	$0.59_{.16}$	$\mathbf{0.72}_{.07}$	$0.69_{.08}$	$0.73_{.05}$	$0.78_{.05}$	$0.80_{.03}$	$0.80_{.01}$	$\mathbf{0.84}_{.03}$
	TabPFN	—	$0.61_{.13}$	$\mathbf{0.67}_{.11}$	$0.71_{.07}$	$\mathbf{0.77}_{.03}$	$\mathbf{0.82}_{.03}$	$\mathbf{0.83}_{.03}$	$\mathbf{0.83}_{.03}$	$\mathbf{0.81}_{.02}$	$0.81_{.03}$
	TabLLM	$\mathbf{0.68}_{.06}$	$0.61_{.09}$	$0.63_{.08}$	$0.69_{.07}$	$0.68_{.04}$	$0.73_{.03}$	$0.79_{.04}$	$0.78_{.02}$	$0.78_{.04}$	$0.80_{.04}$
Heart	XGBoost	—	$0.50_{.00}$	$0.55_{.14}$	$0.84_{.07}$	$0.88_{.04}$	$0.91_{.01}$	$0.91_{.01}$	$0.90_{.01}$	$0.92_{.01}$	$0.94_{.01}$
	TabPFN	—	$\mathbf{0.84}_{.06}$	$\mathbf{0.88}_{.05}$	$0.87_{.06}$	$\mathbf{0.91}_{.02}$	$\mathbf{0.92}_{.02}$	$\mathbf{0.92}_{.02}$	$0.92_{.01}$	$0.92_{.02}$	$0.92_{.02}$
	TabLLM	$\mathbf{0.54}_{.04}$	$0.76_{.14}$	$0.83_{.05}$	$0.87_{.04}$	$0.87_{.06}$	$0.91_{.01}$	$0.90_{.01}$	$0.92_{.01}$	$0.92_{.01}$	$0.94_{.01}$
Income	XGBoost	—	$0.50_{.00}$	$0.59_{.06}$	$0.77_{.02}$	$0.79_{.03}$	$0.82_{.02}$	$0.84_{.01}$	$0.87_{.01}$	$0.88_{.00}$	$\mathbf{0.93}_{.00}$
	TabPFN	—	$0.73_{.08}$	$0.71_{.09}$	$0.76_{.09}$	$0.80_{.04}$	$0.82_{.04}$	$0.84_{.01}$	$0.86_{.01}$	$0.87_{.01}$	$0.89_{.00}$
	TabLLM	$\mathbf{0.84}_{.00}$	$\mathbf{0.84}_{.01}$	$\mathbf{0.84}_{.02}$	$\mathbf{0.84}_{.04}$	$\mathbf{0.84}_{.01}$	$\mathbf{0.84}_{.02}$	$\mathbf{0.86}_{.01}$	$0.87_{.00}$	$\mathbf{0.89}_{.01}$	$0.92_{.00}$
Jungle	XGBoost	—	$0.50_{.00}$	$0.58_{.07}$	$\mathbf{0.72}_{.05}$	$0.78_{.03}$	$0.81_{.02}$	$0.84_{.02}$	$0.87_{.01}$	$0.91_{.01}$	$0.98_{.00}$
	TabPFN	—	$\mathbf{0.65}_{.08}$	$\mathbf{0.72}_{.04}$	$0.71_{.07}$	$0.78_{.02}$	$0.81_{.01}$	$0.84_{.01}$	$\mathbf{0.88}_{.01}$	$0.91_{.00}$	$0.93_{.00}$
	TabLLM	$\mathbf{0.60}_{.00}$	$0.64_{.01}$	$0.64_{.02}$	$0.65_{.03}$	$0.71_{.02}$	$0.78_{.02}$	$0.81_{.02}$	$0.84_{.01}$	$0.89_{.01}$	$\mathbf{1.00}$ †

- TabLLM performs best in **zero-shot / very-few-shot**.
  - **Zero-shot: TabLLM** achieves strong performance on most tasks and is **on par with GPT-3** despite T0 being much smaller (11B vs. 175B).
  - **Few-shot:** Performance improves with more shots; in the **very-few-shot** regime, TabLLM is often **substantially better** than most baselines.
- **TabPFN** is best overall; a small number of shots is often sufficient to catch up to TabLLM.
- LR is frequently the second-best baseline due to extensive parameter tuning.

Discussion

- **Zero-shot capability & prior knowledge:** Simple template serializations (List/Text) yield **meaningful zero-shot** performance, indicating effective use of the LLM's prior knowledge.
- Sample efficiency varies by task: stronger on **semantically rich** datasets (e.g., INCOME) and weaker on **mostly numeric** datasets (e.g., BLOOD).
- TabLLM relies on the LLM's **pretrained semantics**; gains may be limited when column names/values are **out-of-domain** (e.g., gene identifiers).
  - T0's instruction-tuning contains no medical tasks, which may reduce gains on medical datasets.
- **Ethics & responsible use:** Since LLMs can inherit biases and stereotypes from historical training data, applying TabLLM to sensitive tasks (e.g., income or health) requires **careful interpretation**.

Thank you!