MakeItHappen workshop - January 29th, 2026

TabPFN

or when a tabular foundational model finally beats XGBoost

Hollmann et al., 2025, Nature

The problem

Can deep learning outperform tree-based methods in tabular data?

Tabular Data

homogeneous
(contains a single type of data)

columns / rows

	0	1	2
0	1.5	21.0	76.4
1	4.0	35.0	99.7
2	3.0	17.0	85.3
3	4.0	53.0	90.7

All numerical data

	0	1	2
0	'a'	'Mia'	'1'
1	'c'	'Lucas'	'x-1'
2	'e'	'Ang'	'zz'
3	'b'	'Jia'	'0.3'

All string data

heterogeneous
(contains multiple types of data)

	0	1	2
0	1.5	21.0	'Mia'
1	4.0	35.0	'Lucas'
2	3.0	17.0	'Ang'
3	4.0	53.0	'Jia'

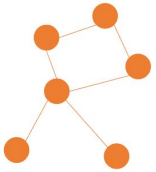
Numerical and string datatypes

Non-tabular Data Examples

Unstructured Text

```
'Twas brillig, and
the slithy toves
Did gyre and gimble
in the wabe:
All mimsy were the
borogoves,
And the mome raths
outgrabe.
```

Network Data



Geospatial Data

Image Data

Video Data

The problem

Can deep learning outperform tree-based methods in tabular data?

A. Classical tabular DL = "train per dataset"

- MLP
- NODE, 2019
- FT-Transformer, 2021

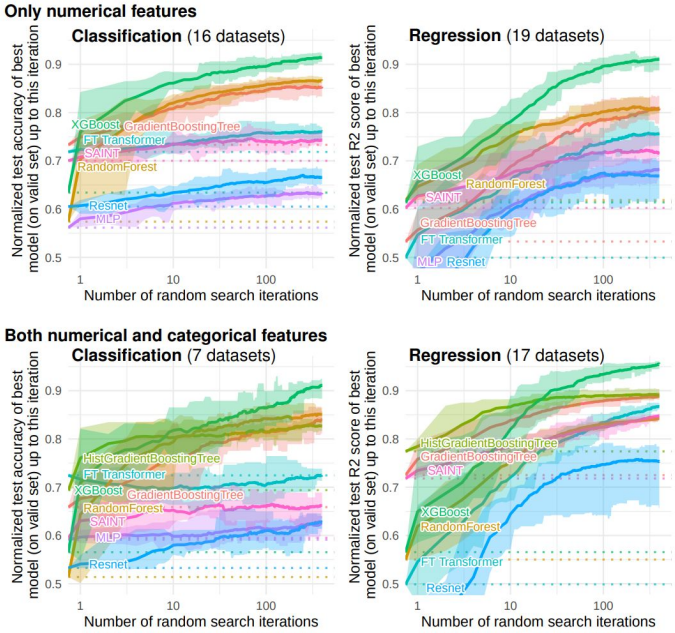
→ Usually worse than trees.

Why do tree-based models still outperform deep learning on typical tabular data?

Léo Grinsztajn
Soda, Inria Saclay
leo.grinsztajn@inria.fr

Edouard Oyallon
MLIA, Sorbonne University

Gaël Varoquaux
Soda, Inria Saclay



The problem

Can deep learning outperform tree-based methods in tabular data?

A. Classical tabular DL = “train per dataset”

B. Tabular transformers with pre-training = “foundational, but still fine-tuned”

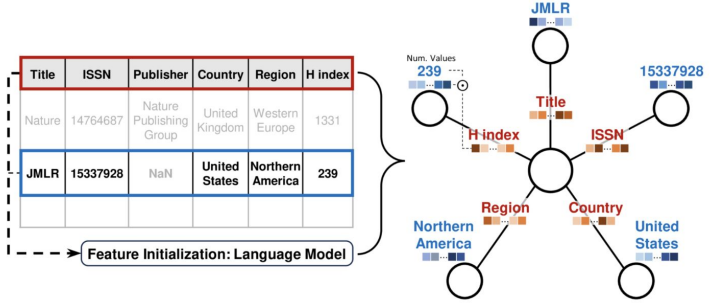
- XTab, 2023
- CARTE, ICML 2024[1,2] (Inria Soda) ⭐

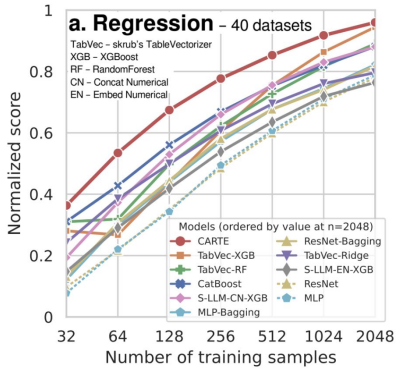
Characteristics:

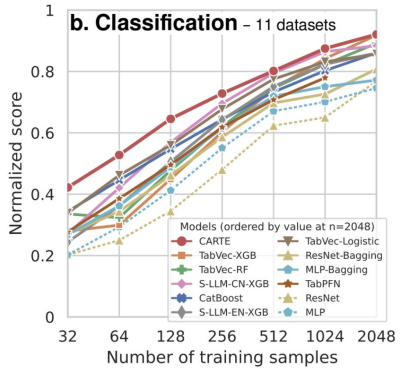
- Pretrain on real datasets
- Still require fine-tuning per dataset
- Still classical training loop: no in-context learning (ICL)

CARTE: Pretraining and Transfer for Tabular Learning

Myung Jun Kim [1] Léo Grinsztajn [1] Gaël Varoquaux [1 2]







1: *Arthur's workshop*
2: *https://gael-varoquaux.info/science/tabicl-pretraining-the-best-tabular-learner.html*

The problem

Can deep learning outperform tree-based methods in tabular data?

A. Classical tabular DL = "train per dataset"

B. Tabular transformers with pre-training = "foundational, but still fine-tuned"

C. In-Context-Learning (ICL) for tabular = "algorithm learning, no training at inference"

- Mitra, NeurIPS 2024 (Amazon Science / AutoGluon)
- TabPFN, Nature 2025 (Prior Labs) ⭐
- TabICL, ICML 2025 (Inria Soda)

Characteristics:

- No training at inference
- Learn an algorithm by pretraining on tasks
- Treat whole training data + evaluation data as input for inference

→ This is the **"new paradigm"**[1,2].

2022

Bojan Tunguz ✔
@tunguz

Neural networks for tabular data is Machine Learning equivalent of wearing socks with sandals.

2026

From Text To Tables: Why Structured Data Is AI's Next $600 Billion Frontier

By Rocio Wu, Contributor. ⊙ Rocio writes about how to Think Global, Act Loca... Follow Author

Published Jan 15, 2026, 02:46pm EST, Updated Jan 16, 2026, 01:39pm EST

1: https://www.linkedin.com/posts/noah-hollmann-668b9010b_three-years-ago-we-released-tabpfnv1-the-activity-7419394986342973441-tQYg/
2: https://www.forbes.com/sites/rociowu/2026/01/15/from-text-to-tables-why-structured-data-is-ais-next-600b-frontier/

What TabPFN actually is

TabPFN = a foundation model that learns a tabular learning algorithm.

Instead of training XGBoost / NN **per dataset,** they:

- pretrain *once* on ~100M **synthetic datasets**,
- learn a transformer that, given (X_train, y_train, X_test), directly outputs p(y_test | X_train, y_train, X_test) in **one forward pass**.

→ itself implements a learned inference algorithm.

At test time there is:

- no gradient descent
- no fitting loop
- no hyperparameter tuning

Just a forward pass that behaves like "Bayesian prediction under a learned prior".

→ This comes from the **Prior-Data Fitted Network (PFN)** idea (Müller et al. 2022) applied to tabular data.

TRANSFORMERS CAN DO BAYESIAN INFERENCE

Samuel Müller [1] **Noah Hollmann** [2] **Sebastian Pineda** [1] **Josif Grabocka** [1] **Frank Hutter** [1,3]
[1] University of Freiburg, [2] Charité Berlin, [3] Bosch Center for Artificial Intelligence
Correspondence to Samuel Müller: muellesa@cs.uni-freiburg.de

PRIOR

PFN + In-Context-Learning

Learning an algorithm, not a model

They train across *datasets*, not samples.

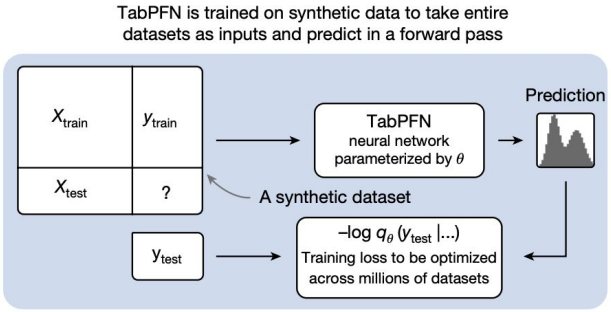
During pre-training:

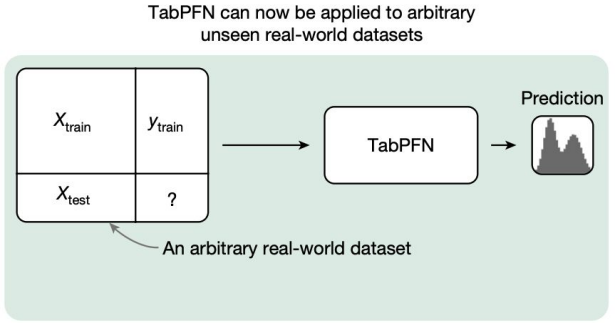
- Each training example = an entire synthetic dataset
- Input = [X_train, y_train, X_test, mask(y_test)]
- Target = true y_test

→ TabPFN ≈ amortized Bayesian inference over a huge family of generative tabular models.

This is why:

- It generalizes **without retraining**
- It outputs **full predictive distributions**
- It adapts instantly to new datasets





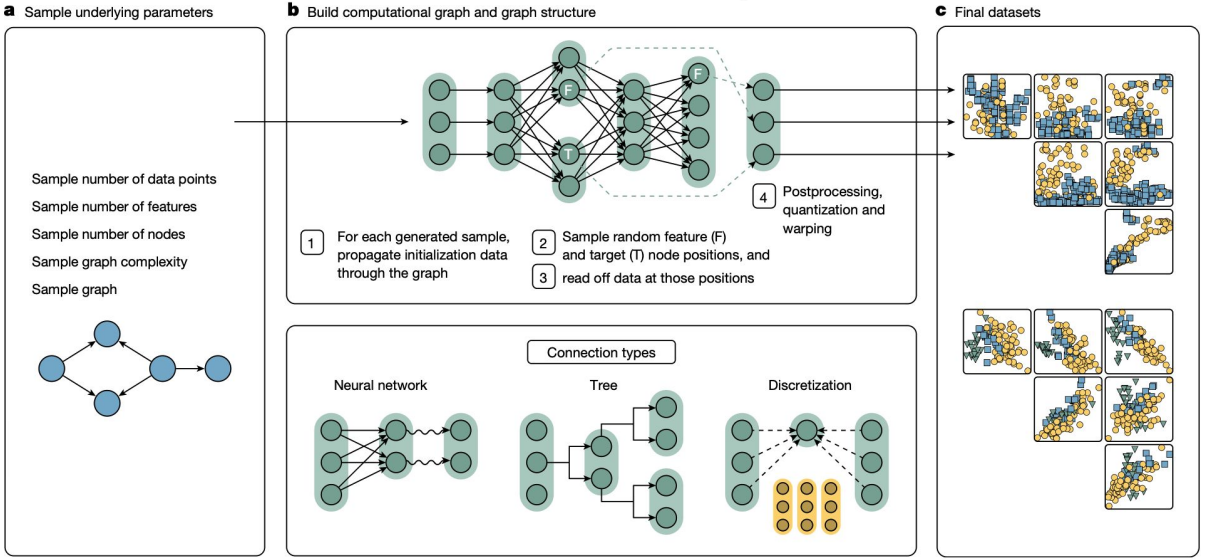
The synthetic data prior

The real (& crazy) secret sauce

They don't pretrain on OpenML or real datasets.
They generate ~**100 million synthetic datasets using structural causal models (SCMs)**, i.e. they:

1. Generate a causal graph for each dataset
2. Apply diverse mechanisms one edges
3. Corrupt data to increase realism

e.g. warping distributions with *Kumaraswamy*



The architecture

A table-aware 2D Transformer

TabPFN treats data as a matrix of cells:

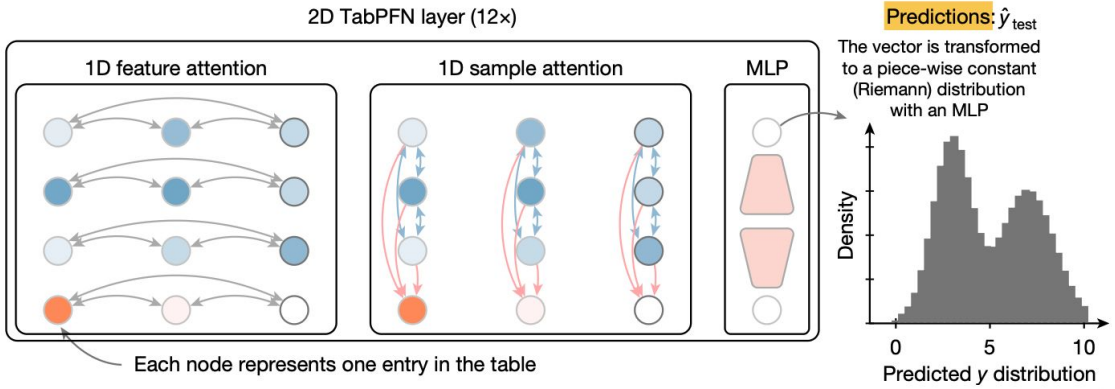
- Each cell = one token.
- **Feature-wise attention** (within a row) → lets features within a sample interact
- **Sample-wise attention** (within a column) → lets different samples interact for the same feature

⚠ Test samples **cannot attend** to each other, they only attend to training samples.

Complexity:

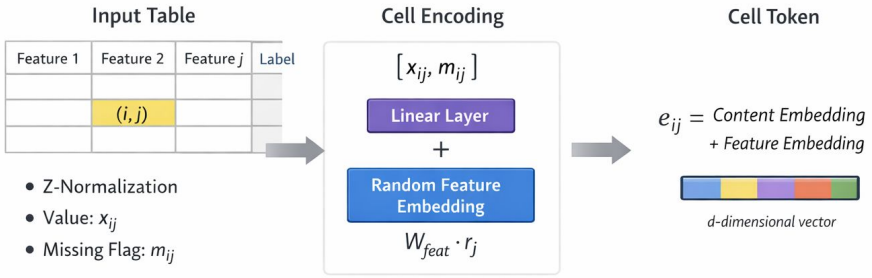
- compute: $O(n^2+m^2)$
- memory: $O(nm)$

→ not scalable to 100k+ rows.



The architecture

How a cell is encoded



Encoding a table cell into a token in TabPFN.

Training & inference mechanics

Training

- Input = whole synthetic dataset
- Mask some y's as "test"
- Optimize log-likelihood of masked targets

$$-\log q_\theta(y_{test} \mid X_{train}, y_{train}, X_{test})$$

They train once for **~2 weeks on 8 GPUs**.

After that, the model is **frozen**.

Training & inference mechanics

Inference on a real dataset

At test time, you give X_train, y_train, X_test and you perform a **single forward pass**:

1. Input embedding for all cells
2. 12× (feature attn → sample attn → MLP)
3. For each test cell corresponding to (x_test, y_masked):
    a. Take its final embedding
    b. Pass through an output MLP head

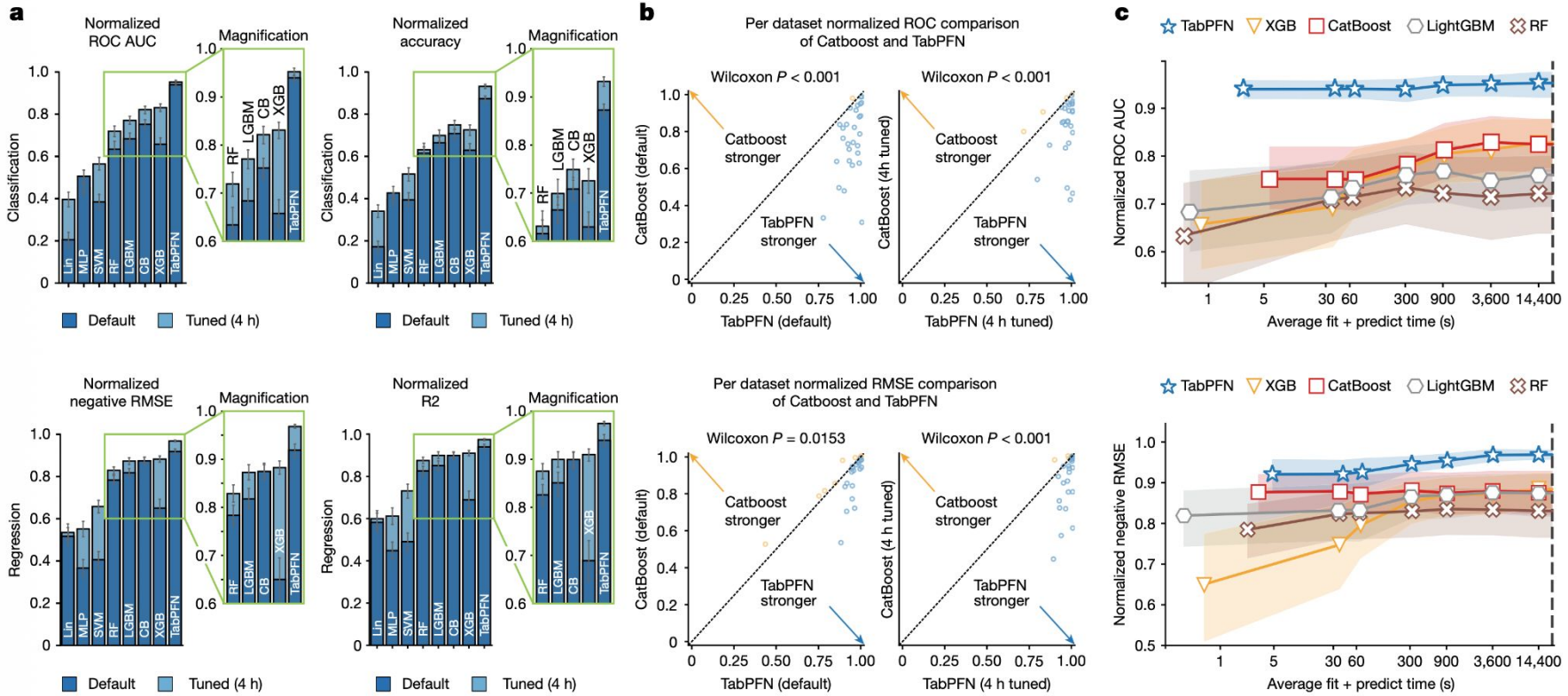
→ For regression: Output = **logits over bins → piecewise constant density**

→ For classification: Output = **class logits**

No fitting. No boosting. No CV.

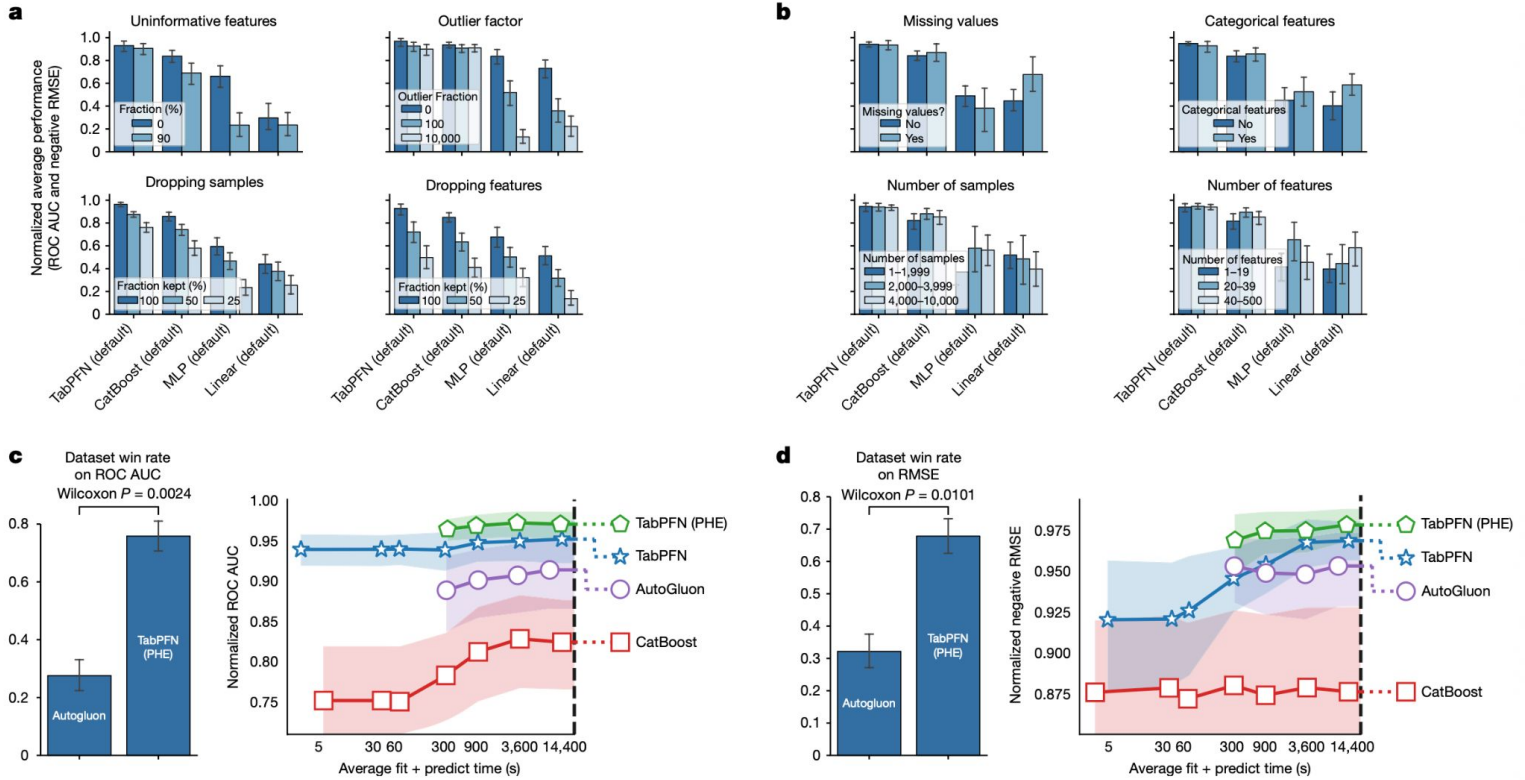
Performance

It beats tree-based methods on small datasets



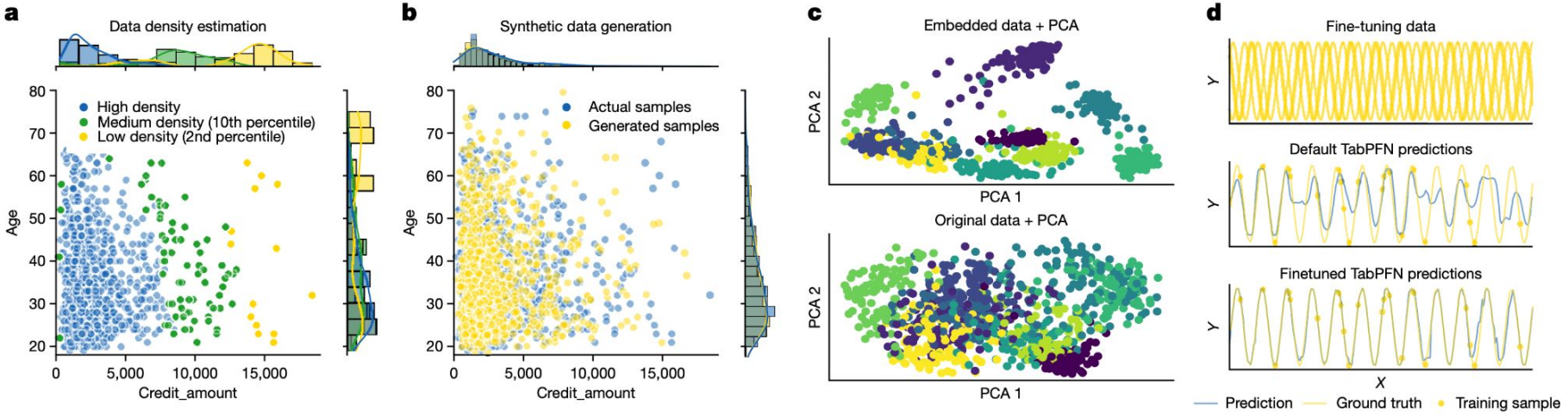
Performance

It really beats tree-based methods on small datasets



Performance

It comes with a great list of useful things



Performance

Why does this beat trees on small data?

3 main reasons:

1. **Richer inductive bias**
   → Pretrained on a broad family of synthetic tabular problems, not limited to axis-aligned rules.

2. **Implicit Bayesian averaging**
   → Approximates posterior predictions by integrating over many possible data-generating mechanisms.

3. **No dataset-specific tuning noise**
   → No hyperparameter search, no early stopping, no CV instability.

Limitations

When it fails

The authors are very explicit:

- Designed for **≤10k rows, ≤500 features**
- Quadratic cost → does not scale well
- On large datasets, CatBoost / LightGBM still win
- If your real data distribution is far from the synthetic prior → performance drops
- No native support of text data, like XGBoost, so text columns (if they can't be categorized) need to be embedded (e.g. BERT)

Landscape

TabPFN vs other ICL approaches

MITRA (Amazon Science)

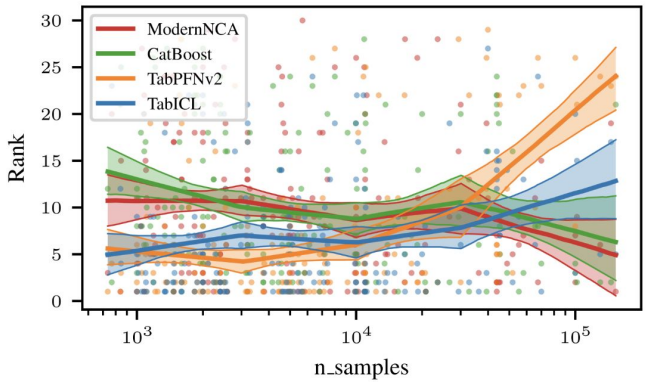
- Learn ICL regression with a transformer
- Focus on theoretical understanding of how ICL emulates kernel regression / Bayesian updates

→ More a theoretical foundation, TabPFN = industrial-scale realization of this idea for tabular data

TabICL[1] (Inria Soda)

- Uses ICL on tabular datasets
- Trains on real datasets (OpenML style)
- Treats each dataset as a sequence of (x, y) pairs
- Standard transformer, not 2D

→ Pre-trained on real data, with no prior from synthetic data compared to TabPFN.
→ Better on larger datasets.



Rank (lower is best) as a function of dataset size.

1: https://gael-varoquaux.info/science/tabicl-pretraining-the-best-tabular-learner.html

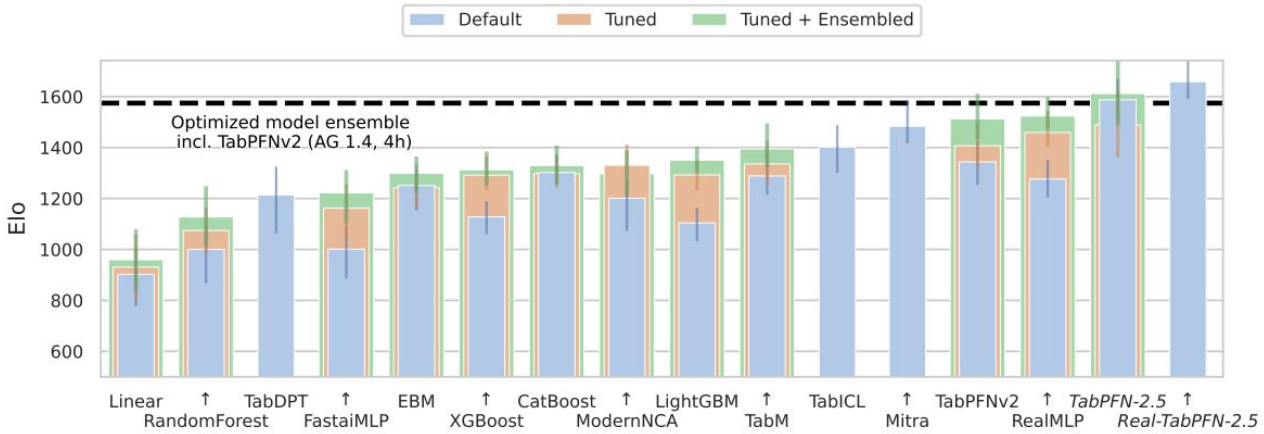
Landscape

TabPFN vs other ICL approaches

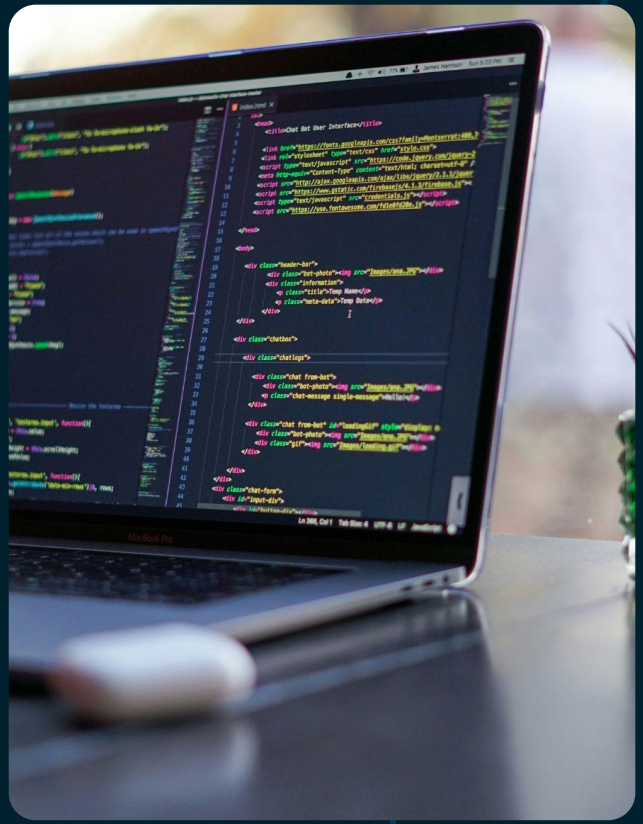
TabPFN-2.5, 2025 (Prior Labs)

- **Increase scale:** Designed for up to **~50,000 samples and 2,000 features** (vs ~10k/500 in v2).
- **Improve architecture:** Deeper transformer layers (18 for regression, 24 for classification) and **larger feature groups** for faster training/inference.
- **Fast inference:** A **distillation engine** produces compact MLP or tree ensembles, preserving accuracy with **lower latency**.
- **Learn from real-world:** A "Real-TabPFN-2.5" fine-tuned on **real datasets** further improves performance.

→ **SOTA**[1]



1: *TabArena leaderboard*



Hands-on

Colab notebook

→ Slightly derived from PriorLabs/tabPFN tutorial

Conclusion

Why does this beat trees on small data?

This paper shows:

- You can replace algorithm design (boosting, trees, kernels) with **prior design + foundation model training**
- Tabular learning can become a **zero-shot task**

→ "Attention + Synthetic Data Is All You Need"*

This has real-world applications, e.g. applying TabPFN-2.5 from scratch in all industries with tabular data e.g. *TabPFN-TS*[1].

→ 600B opportunity according to Forbes.

*: *On small data only so far!*
1: *From Tables to Time: How TabPFN-v2 Outperforms Specialized Time Series Forecasting Models, Hoo et al., 2025*

InstaDeep®

Thanks

Do you have any questions?