



清华大学
Tsinghua University



LAMDA
Learning And Mining from Data



THE HONG KONG
UNIVERSITY OF SCIENCE
AND TECHNOLOGY
(GUANGZHOU)



稳准智能
Stable AI



August 9–13, 2026
KDD2026
Jeju, Korea

Generalist Model for Structured Data: Foundations, Frontiers and Applications

Peng Cui¹, Xingxuan Zhang¹, Hanjia Ye², Jintai Chen³, Shuyang Li⁴

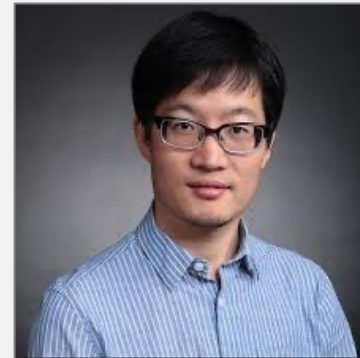
1. Tsinghua University, China

2. Nanjing University, China

3. Hong Kong University of Science and Technology, China

4. Stable AI Technology Co., Ltd.

Introduction



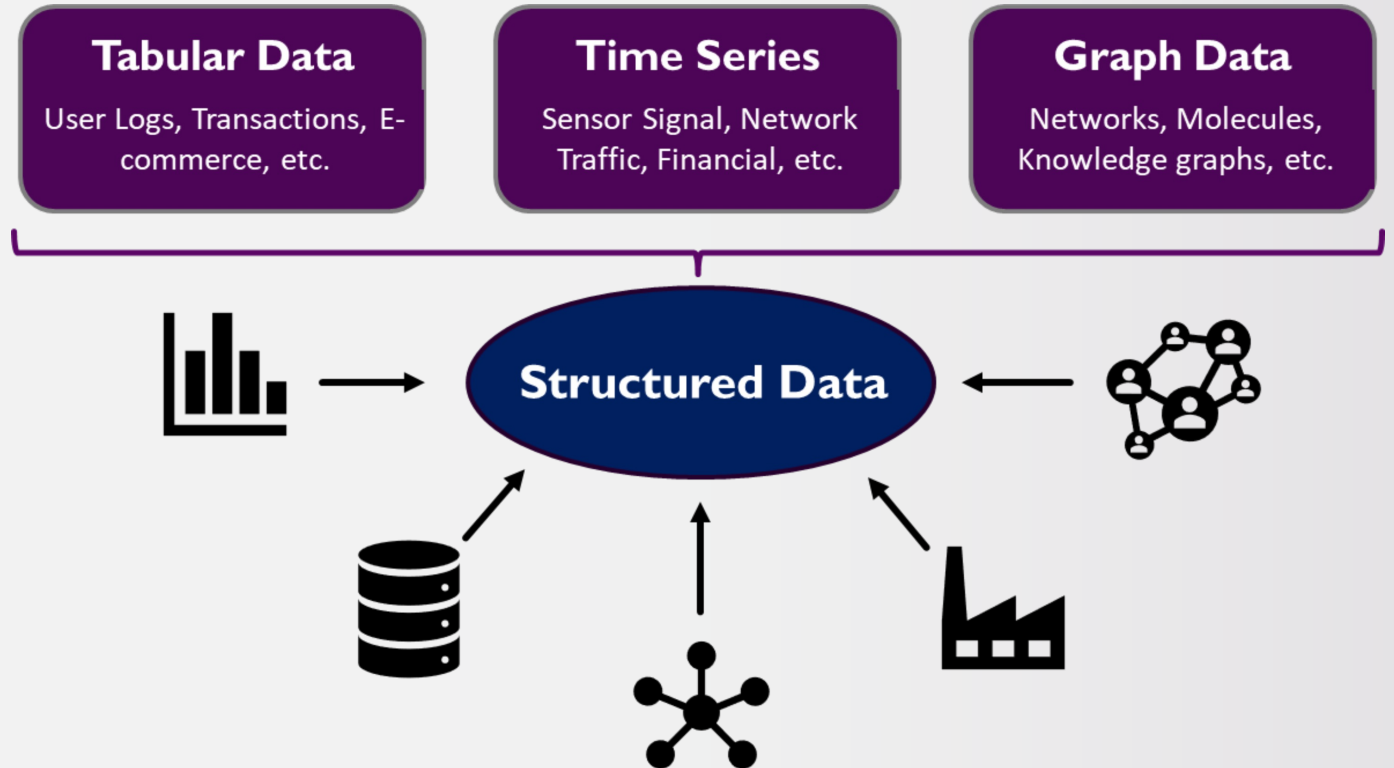
Presenter:

Prof. Peng Cui

Associate Professor, IEEE Fellow
Department of Computer Science
Tsinghua University
Email: cuip@tsinghua.edu.cn

How Structured Data Powers the Modern World

- **Structured data** forms the foundation of real-world systems across a wide range of domains;
- **Structured data** encodes relationships, hierarchies, and dependencies explicitly;
- Unlocking insights from **structured data** is essential for decision-making, automation, and reasoning at scale.



Structured Data in Web Applications



- **Structured Data** is the foundation of business intelligence, enabling organizations to make data-driven predictions and optimize processes;
- It powers personalized experiences such as tailored product recommendations, targeted advertising, and customized content;
- From search engine to finance, structured data is vital for critical operations, improving efficiency and decision-making.

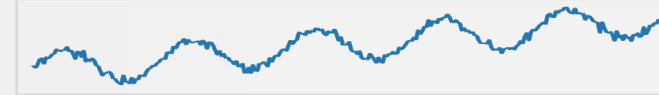
The Three Pillars of Structured Data

Tabular

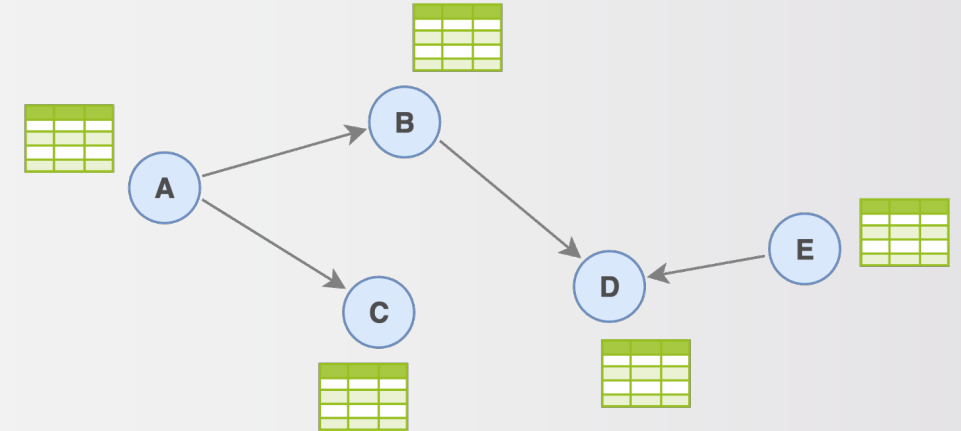
Age	Weight	Height
27	75.2	167
35	73.6	182
19	59.2	175
42	49.8	173
...	...	...

Time Series

Time	Uplink	Downlink
2020/1/1 00:30	12.785	77.023
2020/1/1 01:00	18.282	125.652
2020/1/1 01:30	17.945	143.974
2020/1/1 02:00	16.412	117.872
...	...	...



Graph



- **Tabular Data:** Organized in rows and columns (e.g., customer data, sales records);
- **Time Series:** Data indexed by time, such as network traffic, stock prices or sensor data;
- **Graph:** Represented as nodes (entities) and edges (relationships), common in social networks and recommendation systems.

The Old Guard: How We've Traditionally Modeled It



- **Tabular Data** has traditionally been modeled using techniques such as decision trees, logistic regression, and ML ensemble methods;
- **Time Series** has been modeled using methods like ARIMA, exponential smoothing, and LSTMs, but these methods often require domain expertise;
- **Graph** has been handled with graph neural networks (GNNs) and network analysis techniques, though they struggle with scalability and computational cost.

Why Traditional Models Are Holding Us Back

The pitfalls of traditional approaches:

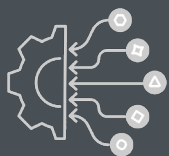
- **Traditional approaches are task-specific**, meaning they are designed for one task and struggle to adapt to other tasks or data types;
- **Traditional approaches require manual feature engineering**, which is time-consuming and prone to biases bias and inefficiency;
- **Traditional approaches often fail to scale** with large datasets or complex, high-dimensional data.

Why Traditional Models Are Holding Us Back

Some issues encountered in real-world applications:

- **Poor Transferability:** Many industrial applications still rely on manual feature engineering pipelines paired with ensemble models like XGBoost and LightGBM. Adapting these solutions to new datasets can be costly, requiring domain expertise and substantial manual effort;
- **Manual Model Updates and Maintenance:** Traditional models require frequent retraining with new data, demanding constant manual updates. For example, in stock market prediction, the model's performance degraded over time as new data (e.g., market conditions, external events) necessitated resource-intensive retraining;
- ...

Summary: Challenges in Structured Data Modeling



Heavy Reliance on Manual Preprocessing

Requires schema alignment, data cleaning, encoding categorical variables, imputation, and scaling.



Per-Dataset Training & Tuning

Models trained from scratch for each dataset or domain. Require hyperparameter tuning, cross-validation, and retraining when data changes.



Limited Transferability Across Tasks and Domains

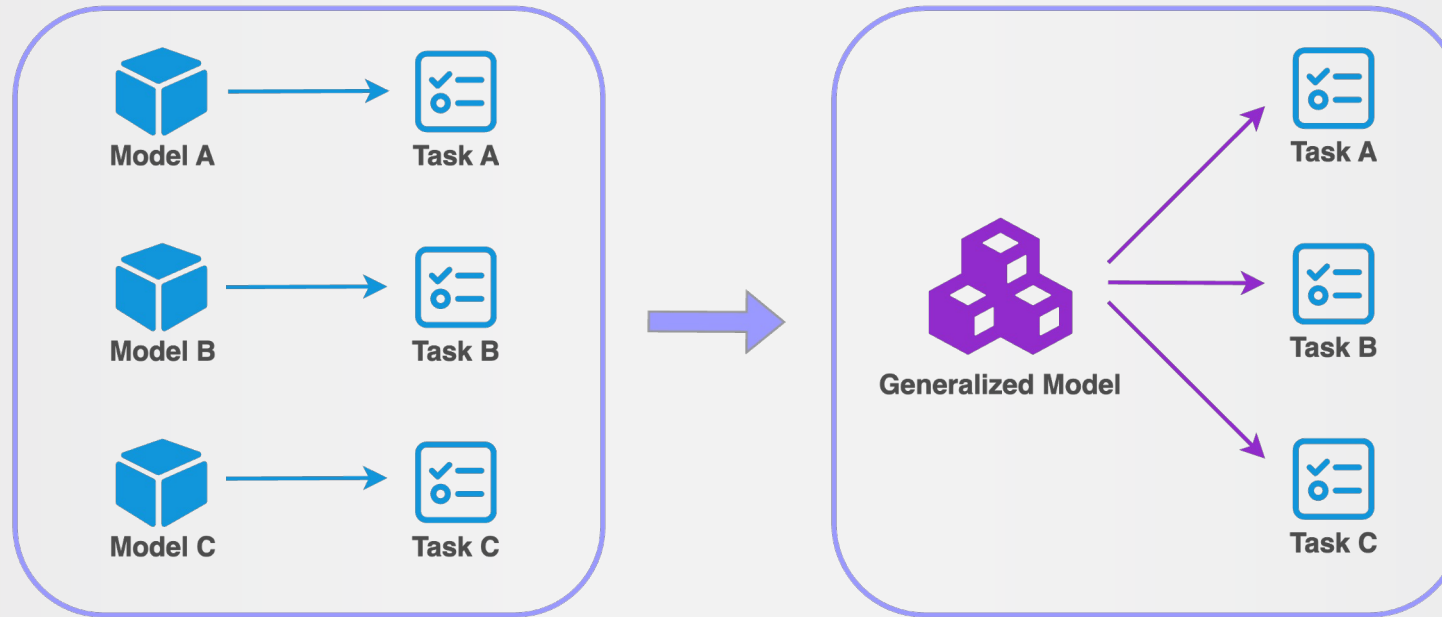
Difficult to transfer knowledge across various domains; poor generalization to unseen entities, attributes, or schema variations.



Inefficiency in Dynamic, Heterogeneous Environments

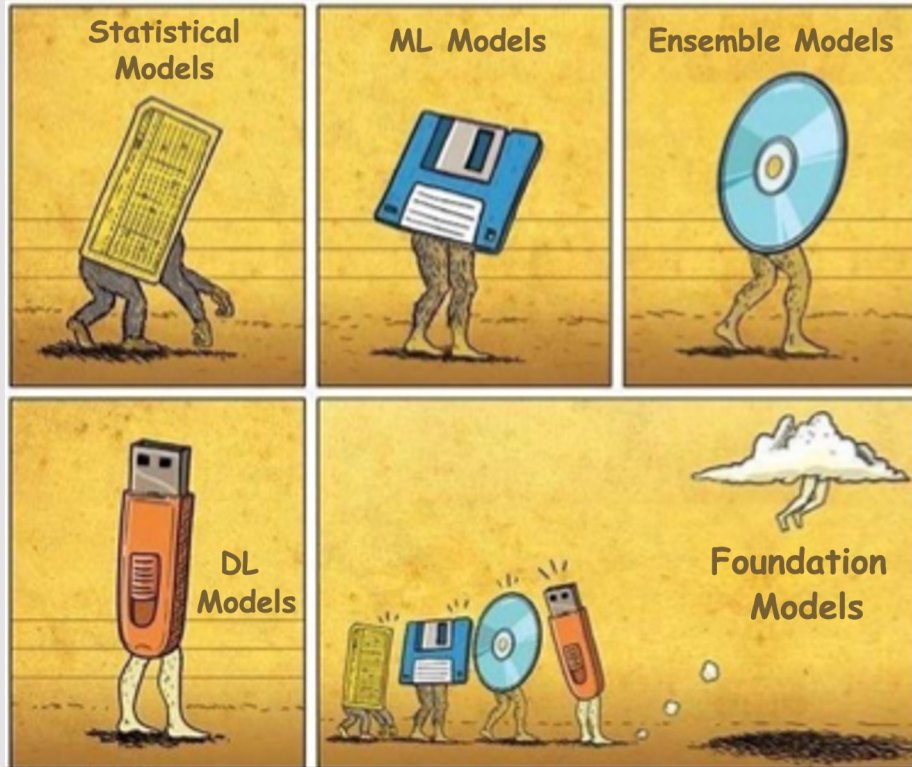
Real-world structured data evolves constantly, and static models struggle to adapt efficiently to changing features, distributions, and schemas.

From Task-Specific to Task-Agnostic Learning



- **Task-specific learning** is limited by their narrow scope and need frequent retraining for different tasks;
- **Task-agnostic learning** aims to develop one model that can handle multiple tasks across diverse data types.

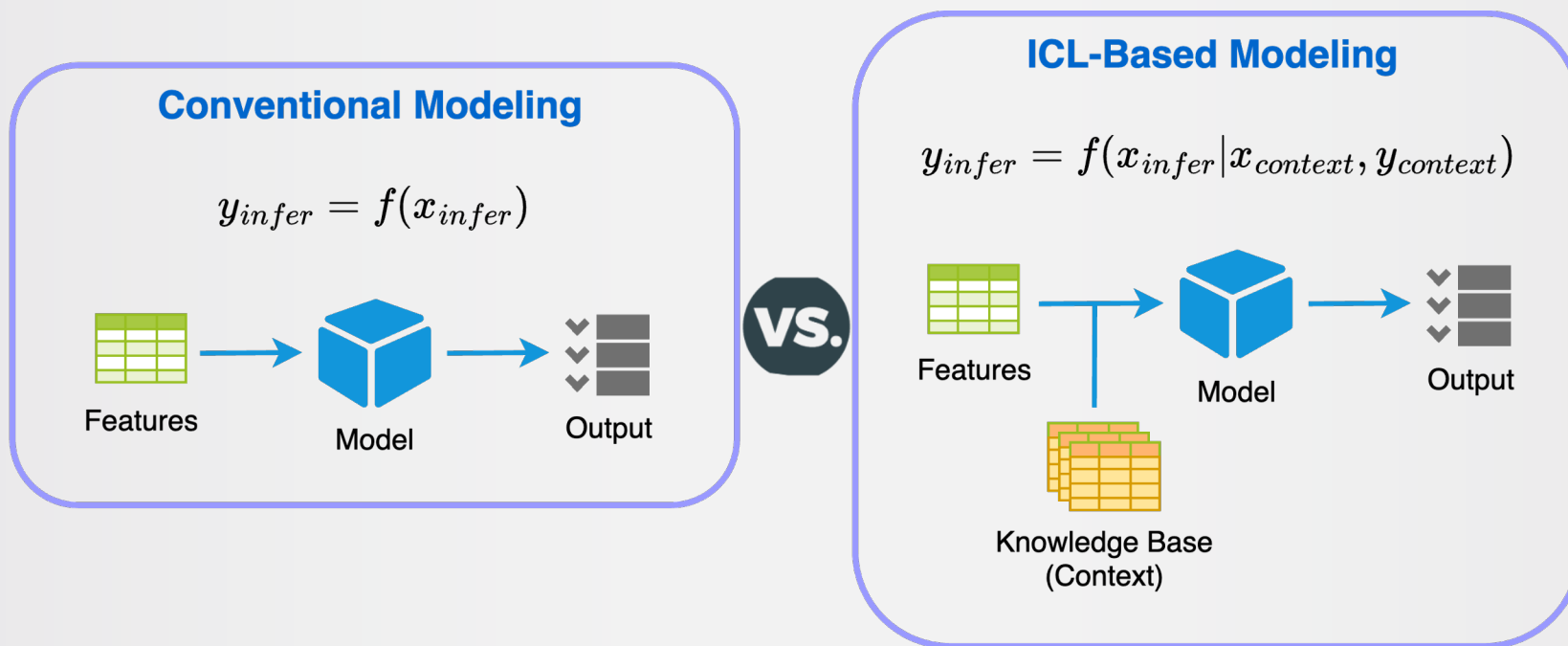
Foundation Models: A Game-Changer



Foundation models are large, pre-trained models that can be adapted to a wide range of tasks with little to no fine-tuning, achieving impressive performance with zero-shot learning.

The use of foundation models offers several benefits: it reduces the need for dataset-specific and task-specific feature engineering and training.

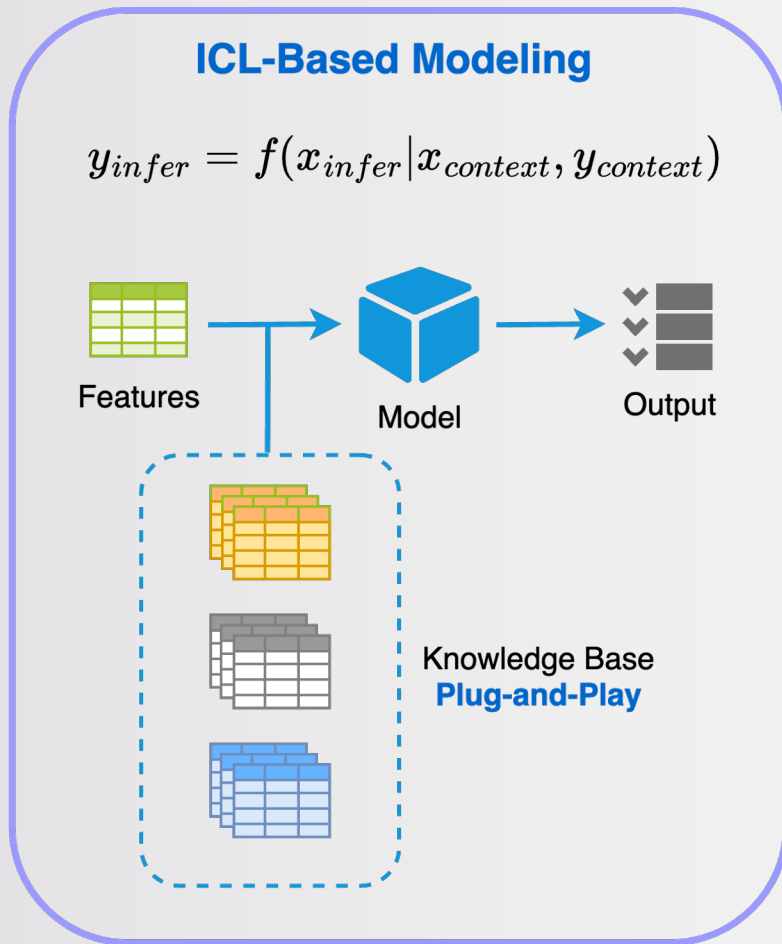
In-Context Learning: The Magic of Foundation Models



Conventional Modeling focuses on learning a static mapping from the given features to the desired ground truth, enabling direct-style inference.

ICL-Based Modeling focuses on learning a joint distribution between the given features and the provided knowledge base (context), facilitating query-style inference.

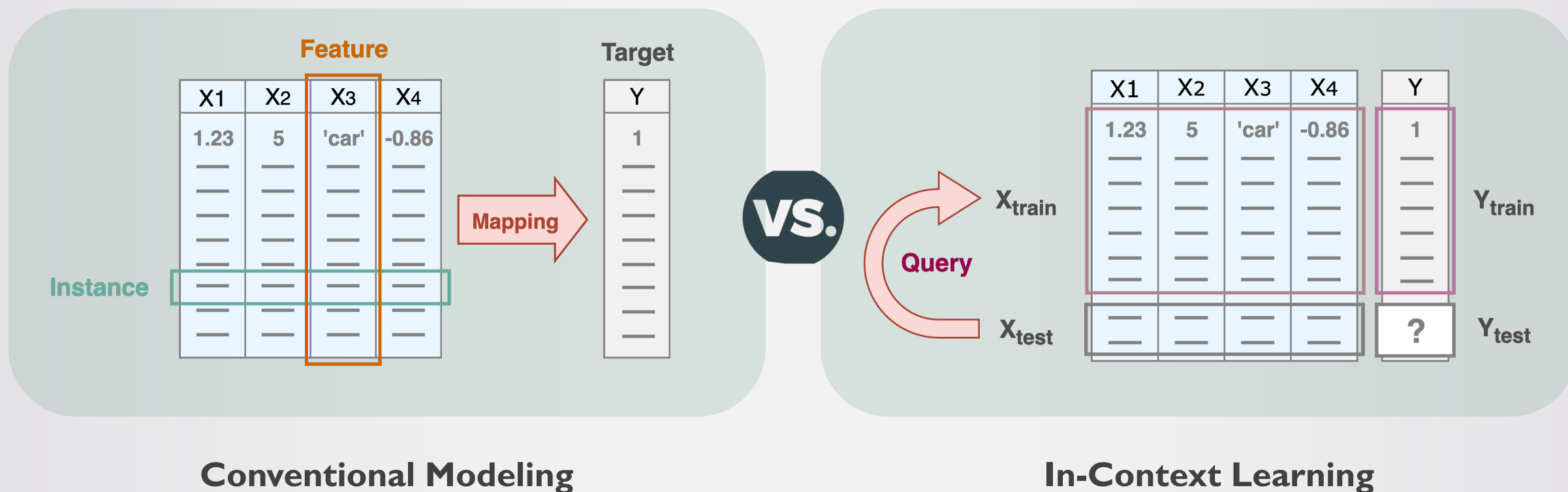
In-Context Learning: The Magic of Foundation Models



ICL-Based Modeling offers several key benefits:

- **No Finetuning Needed:** Leverages pre-trained models without extensive retraining.
- **Task Adaptability:** Handles new tasks with minimal data using context.
- **Flexibility:** Can perform various tasks from the same model.
- **Efficient Knowledge Use:** Integrates external knowledge directly through context.
- **Query-Style Inference:** Supports dynamic, context-driven responses.

Example: In-Context Learning for Tabular Modeling

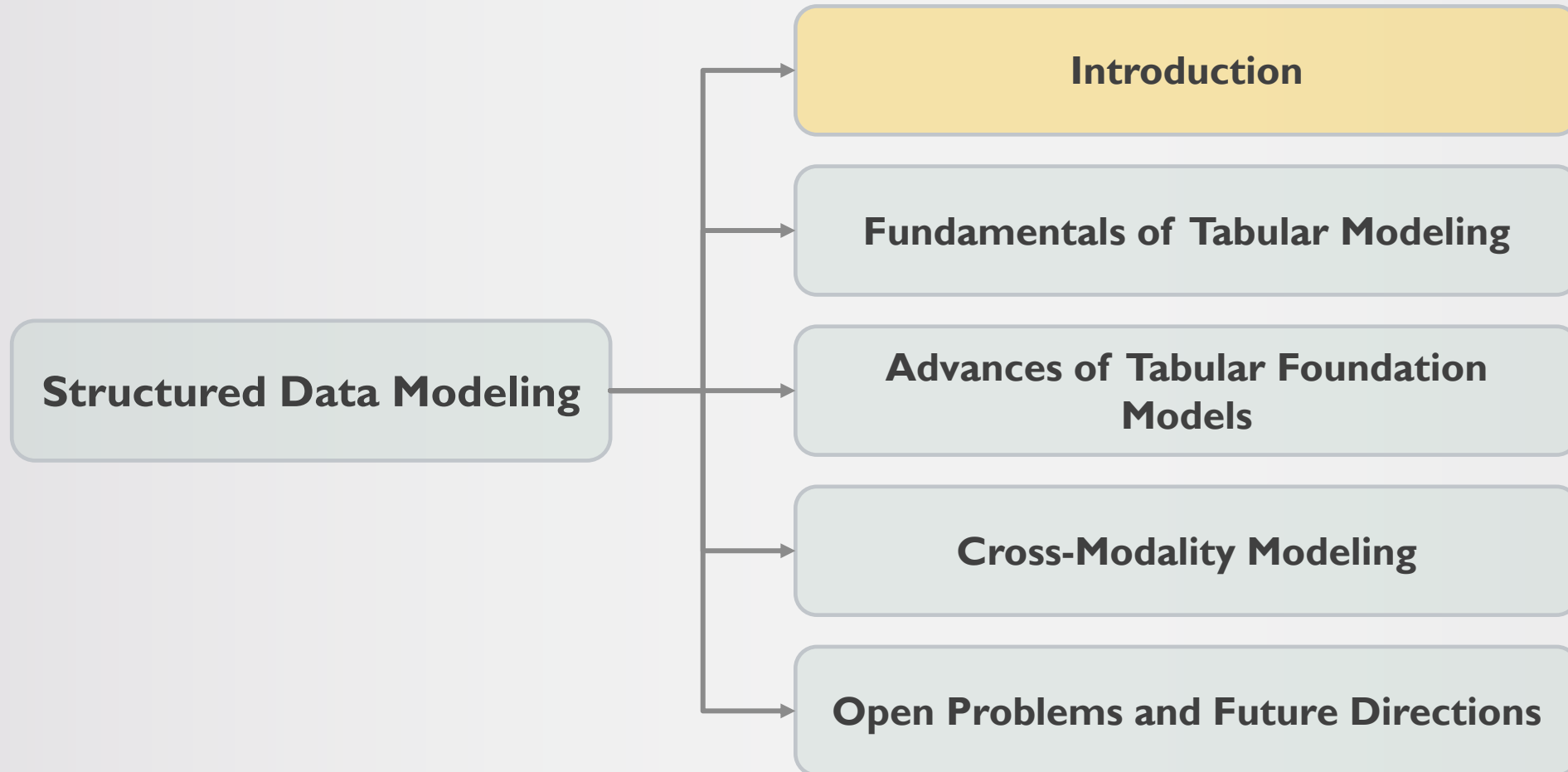


Challenges in Building Foundation Models

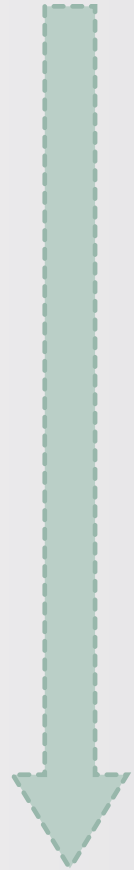
To build foundation models for structured data, there are few challenges:

- **Data Shortage:** High-quality structured data is harder to obtain, making pretraining more challenging;
- **Out-Of-Distribution (OOD) Adaptation:** Due to the inherent characteristics of structured data, OOD scenarios are more common, and they can be particularly challenging to handle;
- **Training Strategies:** A well-designed training strategy is crucial for unifying different tasks and modalities, though achieving this in practice can be quite challenging;
- **Lack of Universal Pre-trained Models:** Unlike NLP and CV, there are few widely accepted, pre-trained foundation models for structured data.

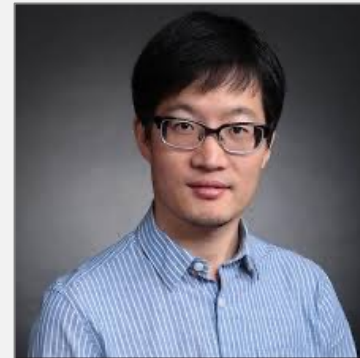
Tutorial Schedule



Timeline



Q&A



Presenter:

Prof. Peng Cui

Associate Professor, IEEE Fellow
Department of Computer Science
Tsinghua University
Email: cuip@tsinghua.edu.cn

Fundamentals of Tabular Modeling



Presenter:

Dr. Xingxuan Zhang

Assistant Researcher

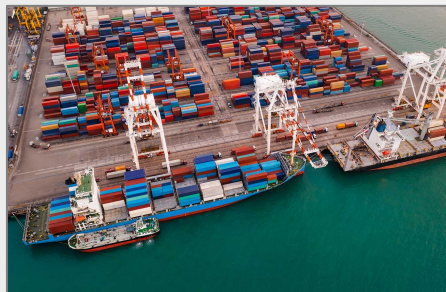
Department of Computer Science

Tsinghua University

Email: xingxuanzhang@hotmail.com

What Is Tabular Data?

Tabular data is ubiquitous across domains, with large volumes continuously generated every day, forming a fundamental layer of real-world information.



Tabular Data

Instance

Feature				Target
X1	X2	X3	X4	Y
8	1.03	'male'	1996	1
—	—	—	—	—
—	—	—	—	—
—	—	—	—	—
—	—	—	—	—
—	—	—	—	—
—	—	—	—	—
—	—	—	—	—
—	—	—	—	—
—	—	—	—	—

Numerical Feature: e.g., 1.03, -2.8, 9.65

Categorical Feature: e.g., male, female

Tabular Modeling: Where Most AI Journeys Begin



Many people start learning AI by solving simple tabular problems that train a model to approximate a function $f(\cdot)$ from data.

Tabular Modeling: Where Most AI Journeys Begin



Given a training set

$$\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^N,$$

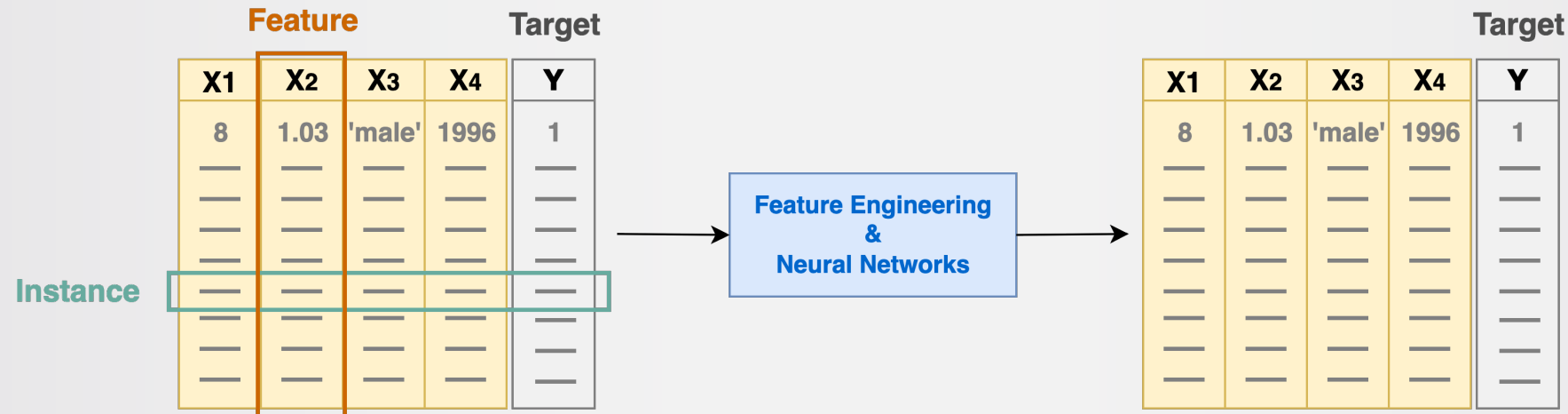
where N is the number of instances (rows) and d is the number of features (columns), we have:

- $\mathbf{x}_i \in \mathbb{R}^d$: feature vector (mix of categorical and numerical attributes)
- $y_i \in \{0, 1\}$ for binary classification
- $y_i \in \{1, \dots, C\}$ for multi-class classification
- $y_i \in \mathbb{R}$ for regression

The goal is to learn a function f that maps inputs to outputs. Given an unseen instance $\mathbf{x}^*$, we predict:

$$\hat{y}^* = f(\mathbf{x}^*, \mathcal{D})$$

Tabular Features



Traditional tabular features are typically discrete or continuous raw variables with clear real-world meaning (e.g., age, income, temperature). However, in modern tabular modeling, features can also be:

- **Mathematical / Statistical Features**

Extracted through feature engineering pipelines
(e.g., counts, ratios, aggregations, rolling statistics) ;

- **Learned Representations**

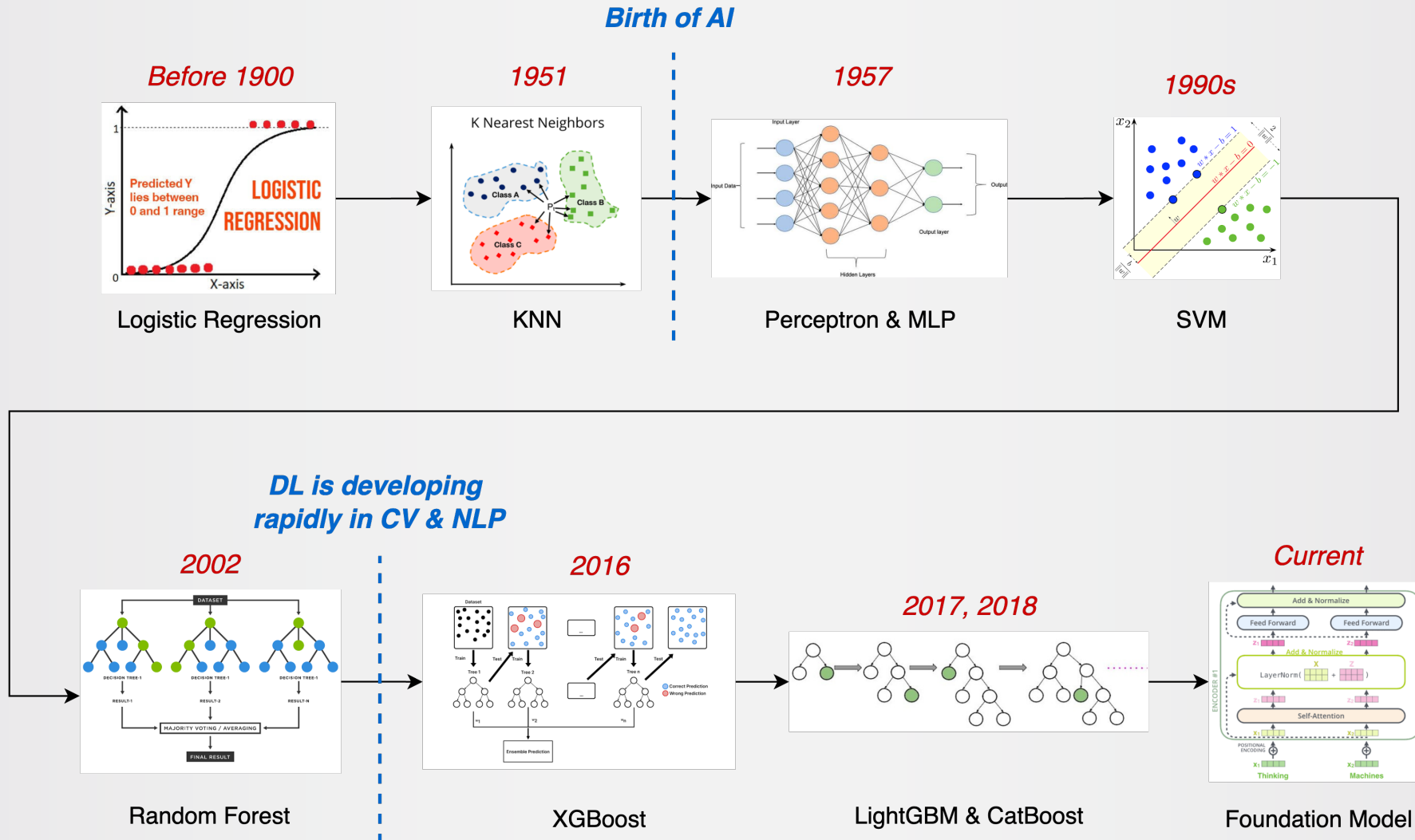
Generated by deep learning models
(e.g., embeddings, latent vectors, representation learning outputs).

In tabular modeling, raw data is rarely directly optimal for learning. Instead, we transform it into informative features that make patterns easier for models to capture.

Common Types of Feature Engineering:

- **Mathematical / Statistical Transformations**
e.g., Log transforms (e.g., log income), Ratios (e.g., debt-to-income), Normalization / scaling
- **Aggregation Features**
e.g., Mean / variance over time windows, User-level or group-level summaries
- **Time-based Features**
e.g., Recency (time since last event), Seasonality (day/week/month patterns)
- **Categorical Encoding**
e.g., One-hot encoding, Target encoding, Frequency encoding

Tabular Modeling Methods



Parametric vs. Non-Parametric Models

Given a tabular dataset $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^N$, learn a mapping f which predicts an unseen instance x_i .

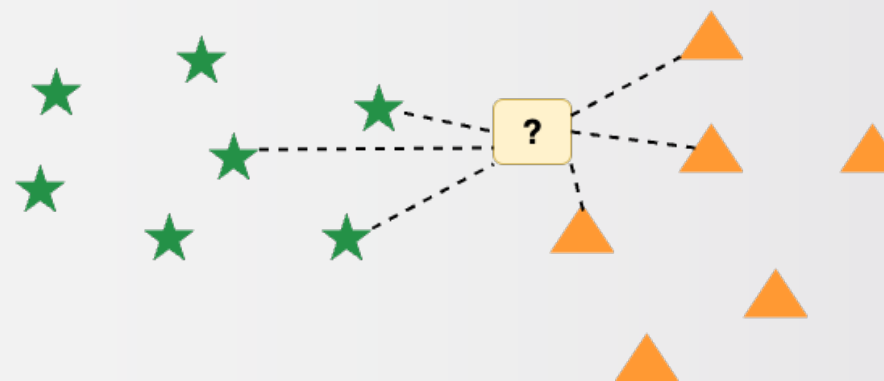
$$\min_f \sum_{(x_i, y_i) \in \mathcal{D}} \mathcal{L}(y, \hat{y}_i = f(x_i)) + \Omega(f)$$

Loss Function

Regularization

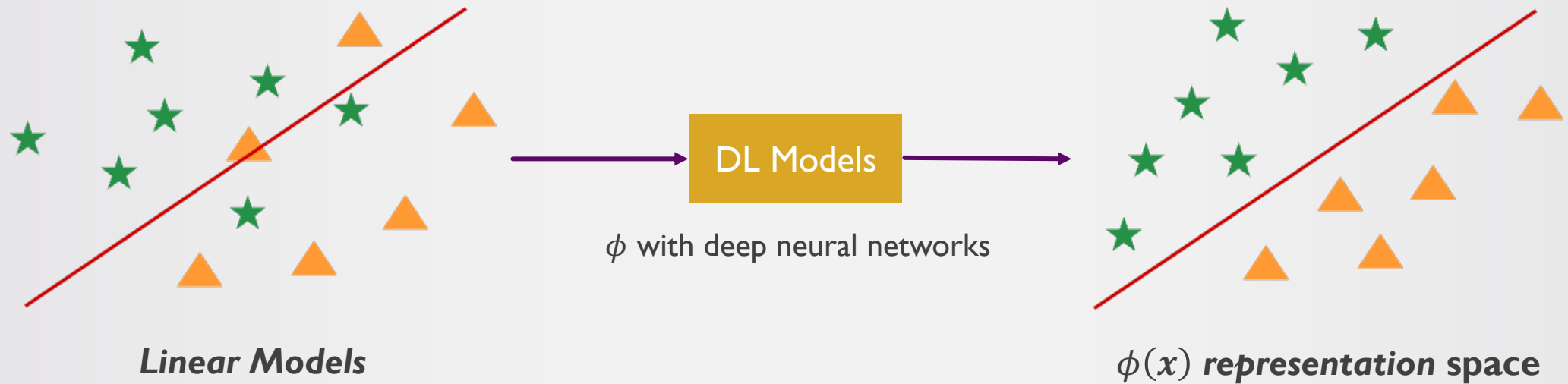


Parametric models



Non-Parametric models

Deep Representation Learning

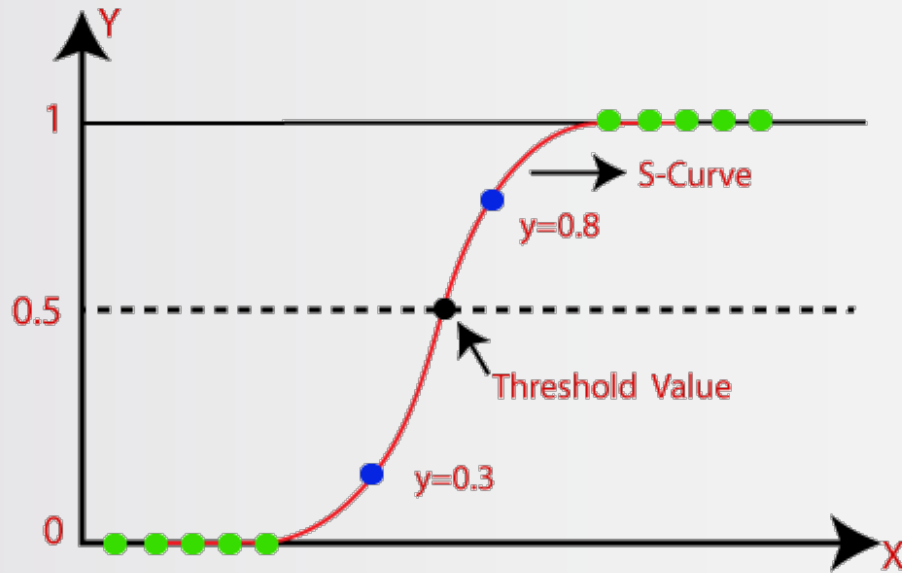


$$f(\mathbf{x}) = W^{\top} \phi(\mathbf{x}) + b$$

----- representation
space mapping

Deep learning learns representations in an embedding space, making complex problems easier to solve than in the original feature space.

Logistic Regression



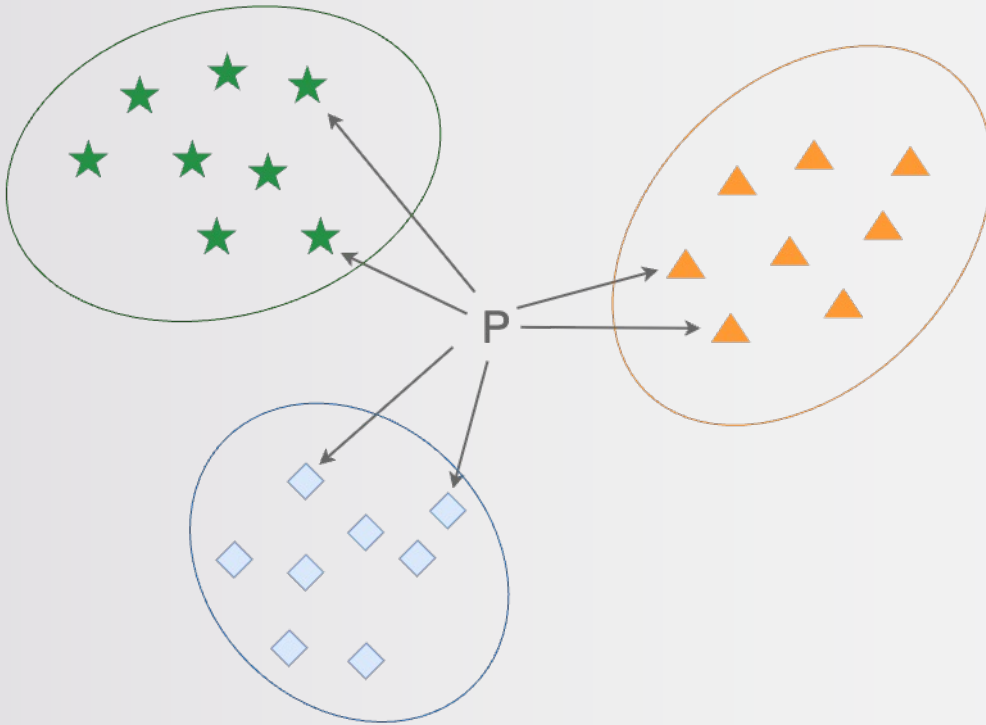
$$z = w \cdot X + b$$

$$P(y = 1) = \sigma(z)$$

$$P(y = 0) = 1 - \sigma(z)$$

$$\sigma(z) = \frac{1}{1 + e^{-z}}$$

Logistic regression is a supervised **parametric** machine learning algorithm for classification tasks, where the goal is to predict discrete categories rather than continuous values.



Given a tabular dataset $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^N$, the label of a query sample x_j is predicted by identifying its **K** nearest neighbors under a distance metric **$d(\cdot)$** , and aggregating their labels.

KNN can be applied in different representation spaces:

- **Raw Data Space**

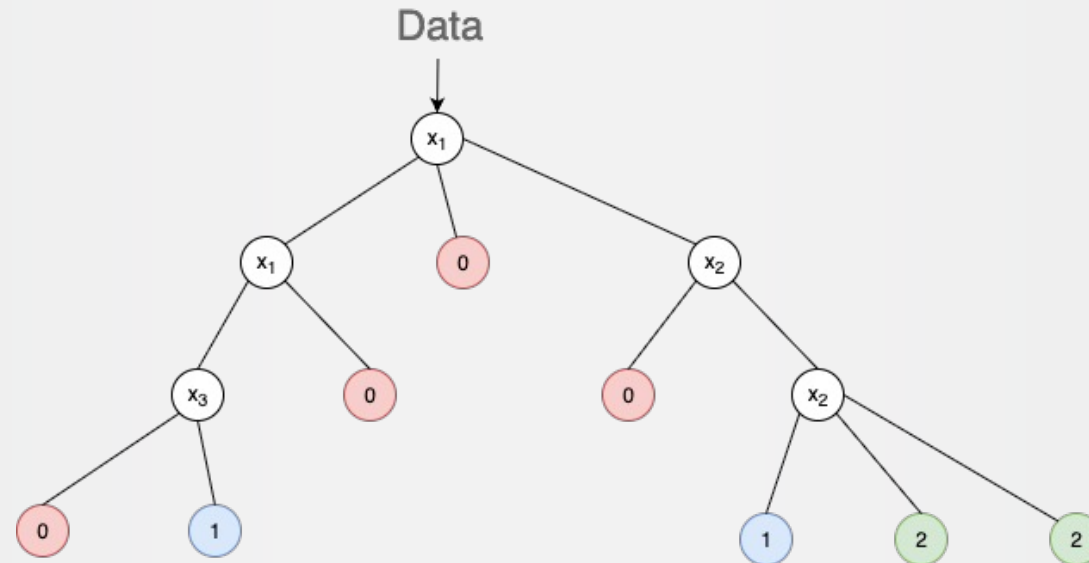
Distance computed directly on **original features**

Sensitive to scaling and feature quality

- **Embedding Space**

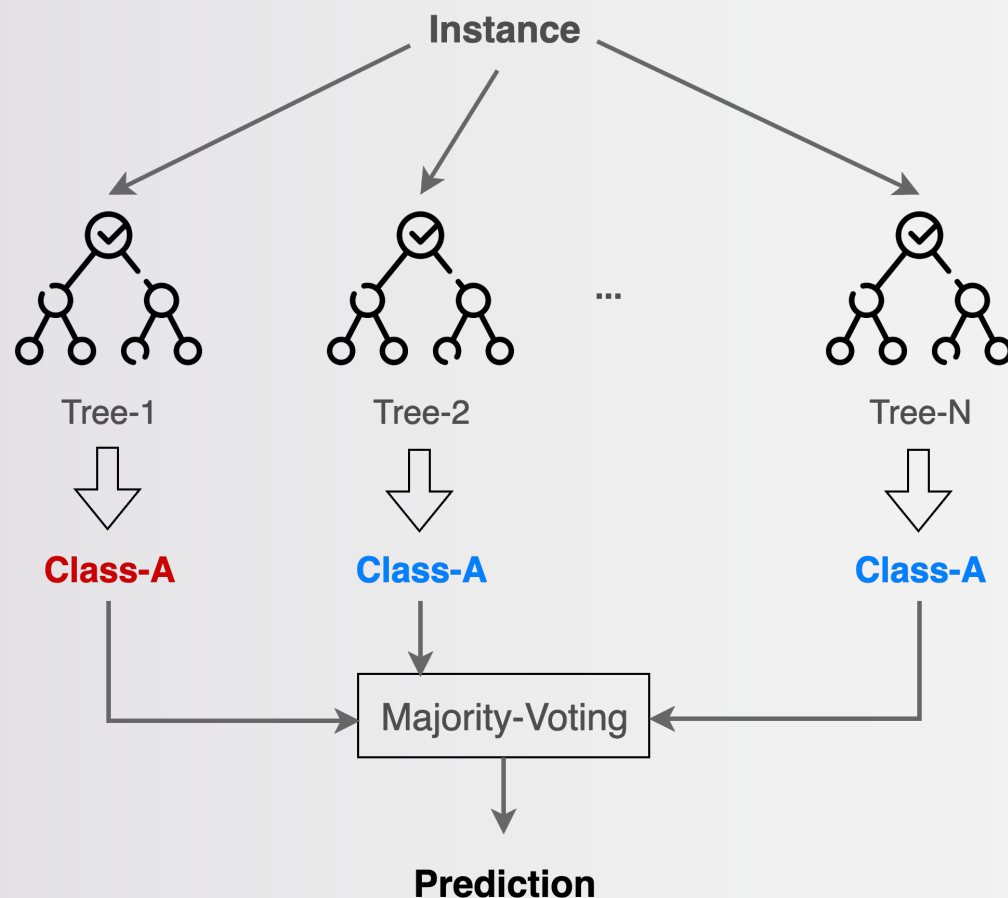
Distance computed on **learned representations**

Often improves semantic similarity



A **Decision Tree** is a supervised machine learning algorithm used for both **classification** and **regression**, which makes decisions by splitting data into smaller and smaller groups based on feature values; where:

- Each **internal node** = a decision based on a feature
- Each **leaf node** = final prediction

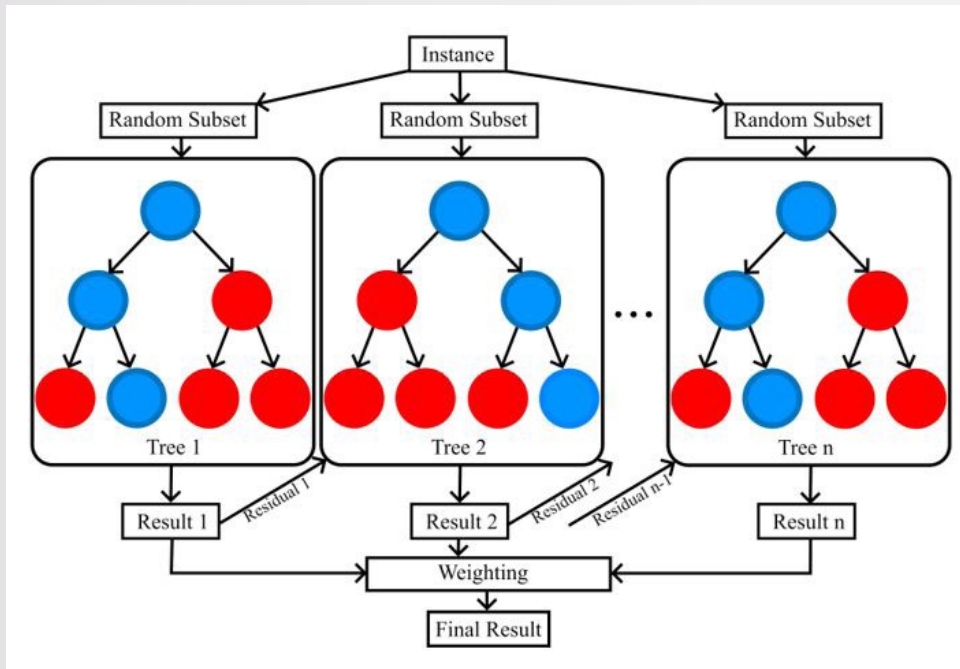


Random Forest is also an ensemble learning method, but unlike XGBoost/LightGBM, it is based on:

Bagging (Bootstrap Aggregation), not Boosting

It builds many decision trees independently and in parallel, then combines their outputs.

Random Forest reduces variance by averaging many independent trees, while boosting methods reduce bias by sequentially correcting errors.



Oztornaci, Burak, Baris Ata, and Serkan Kartal. "Analysing household food consumption in Turkey using machine learning techniques." *Agris on-line Papers in Economics and Informatics* 16.2 (2024): 97-105.

XGBoost (Extreme Gradient Boosting) is an ensemble learning method based on gradient boosting, which builds many small decision trees and combines them to make strong predictions.

Instead of building one big tree, it builds:

- *Tree 1* → makes predictions
- *Tree 2* → fixes errors from *Tree 1*
- *Tree 3* → fixes remaining errors
- ... and so on

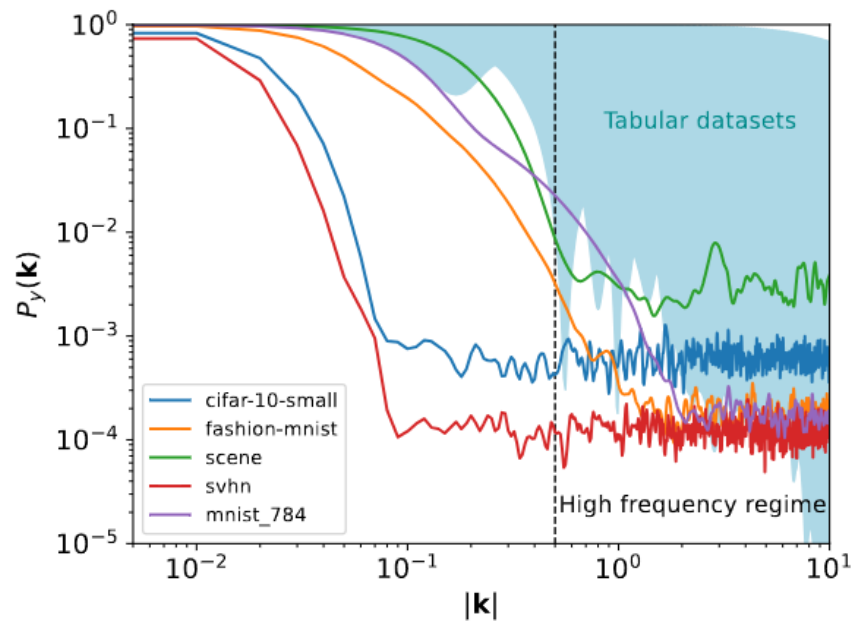
Finally, all trees are combined for the final prediction.

XGBoost, LightGBM and **CatBoost** are widely regarded as the most powerful machine learning methods for tabular modeling.

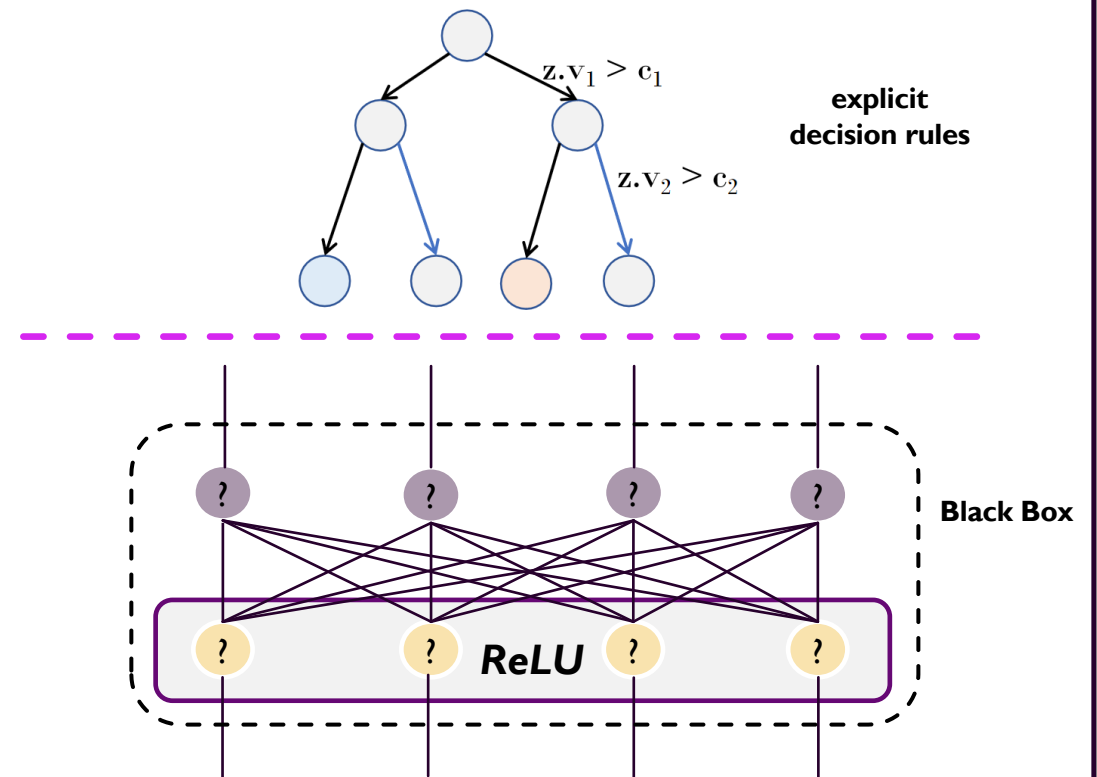
Tree Models vs. Deep Models

Does the tree model still have an advantage?

More advantages in processing
high-frequency data

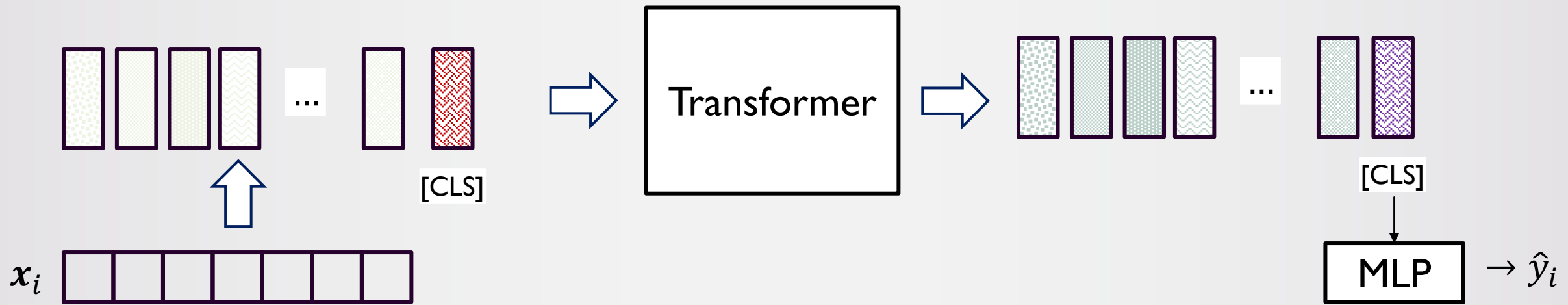


Better interpretability



FT-Transformer (FT-T)

- Tokenize instance x_i , transformed into a collection of d k -dimension tokens
- All tokens are input into a Transformer and concatenated with a [CLS] token. The corresponding input is then predicted through MLP



FT-T learns the interaction and fusion between features more flexibly through the self-attention mechanism. It introduces bias for each feature in the tokenizer and introduce [CLS] tokens to aggregate global information.

- Numerical feature encoding is not merely a “dimensional upgrade”, but rather a readjustment of feature scale and correlation in the latent space
- Through a set of learnable parameters $c_1 \dots c_k$, periodic embedding mapping values to vectors through sine and cosine functions

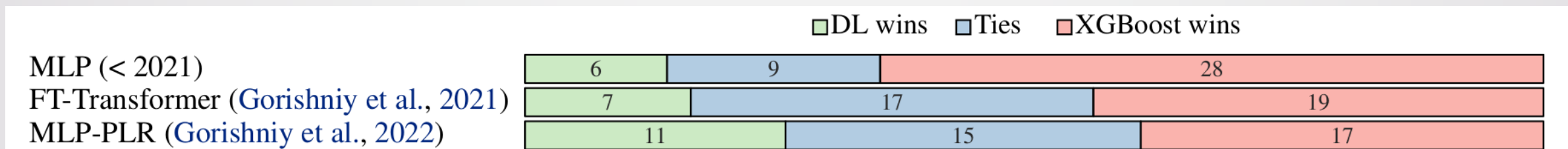
$$\text{Periodic}(x) = \text{concat}[\sin(v), \cos(v)], v = [2\pi c_1 x, \dots, 2\pi c_k x]$$

- PLR embedding adds Linear and ReLU layers after it:

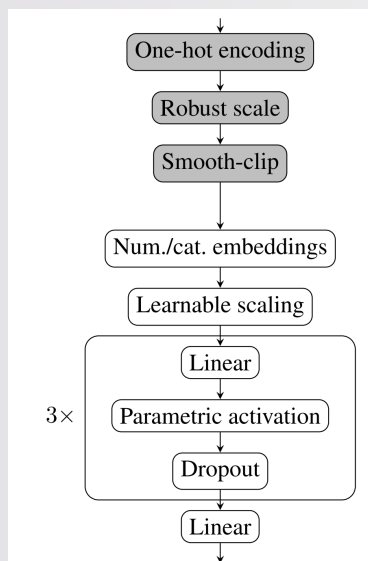
$$\text{PLR}(x) = \text{ReLU}(\text{Linear}(\text{Periodic}(x)))$$

- Learn richer semantic representations for numerical values by mapping numerical scalars to vectors with values between $[-1, 1]$

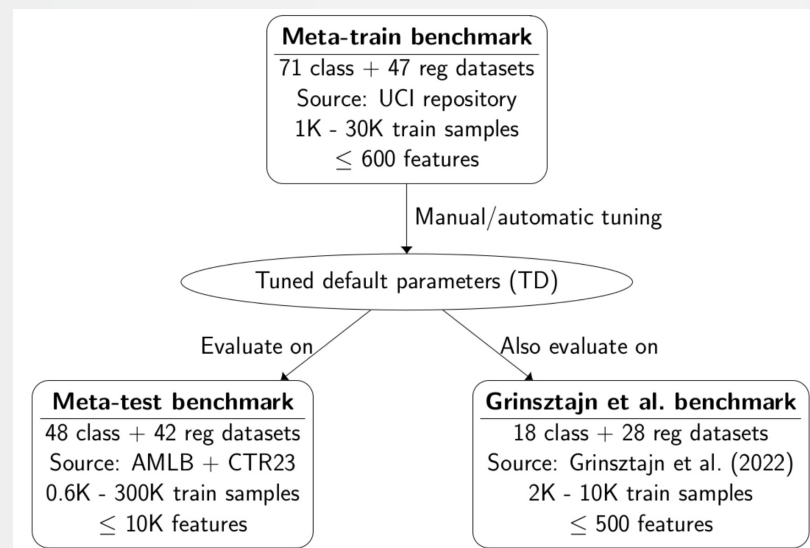
The performance of MLP combined with PLR (MLP-PLR) can be significantly improved, even surpassing FT-T.



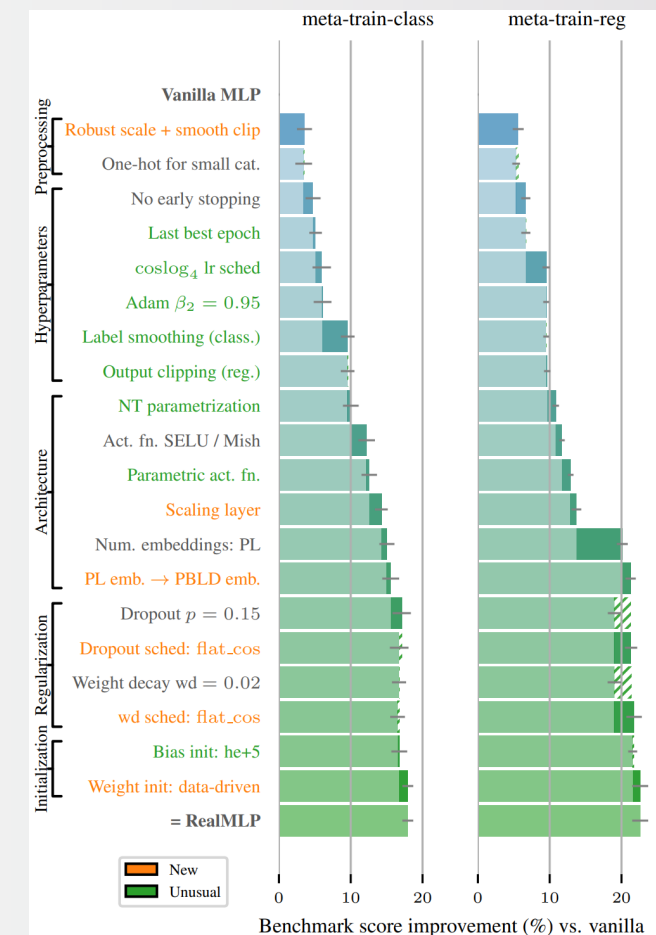
- A “better by default” model which can achieve performance comparable to the optimal parameter tuning results without hyperparameter search
- Improve traditional MLP from multiple perspectives such as structure, training, and hyperparameters
- Find a set of good default hyperparameters through meta-tuning



Architecture



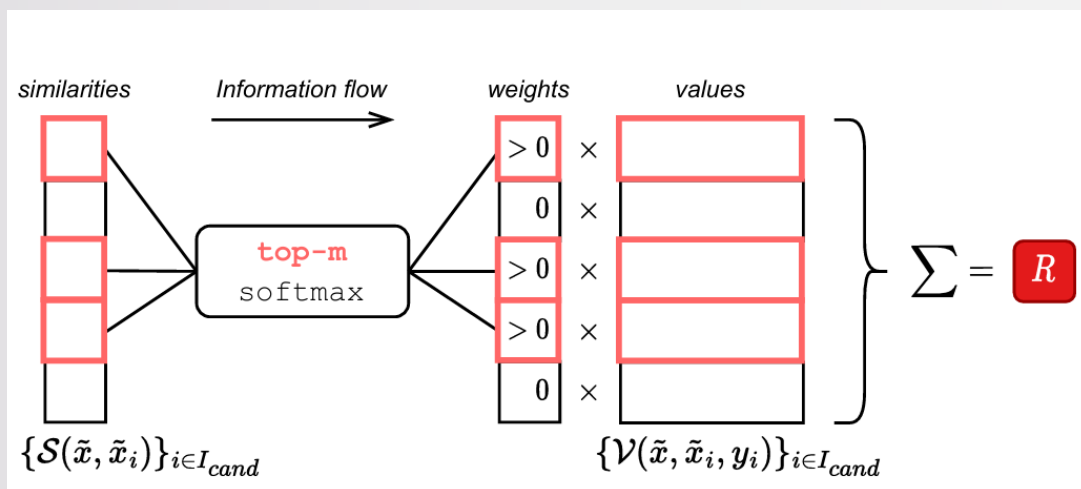
Meta-tuning



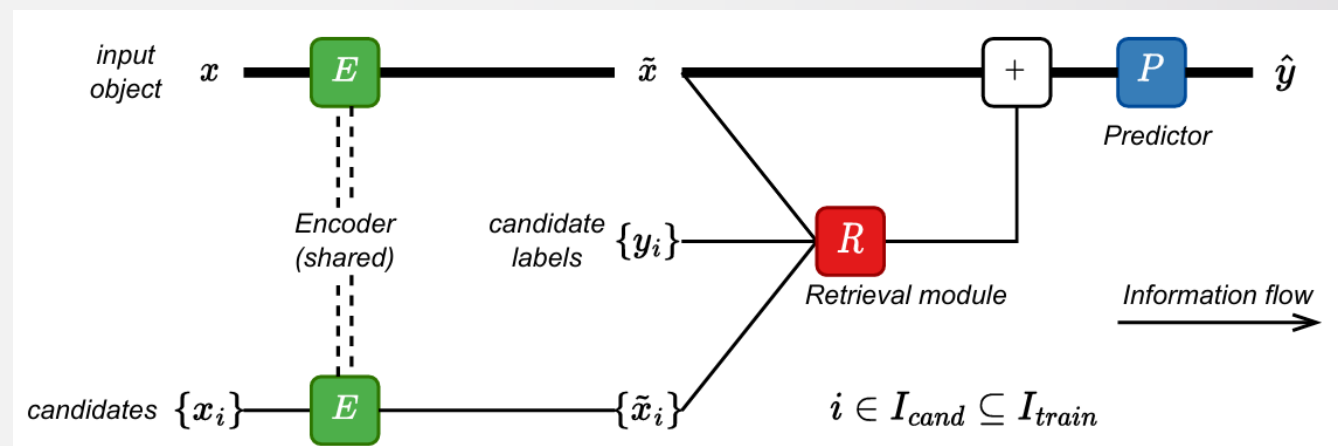
Improvements with tricks

- TabR retrieves the m nearest neighbors with the highest similarity through the attention structure
- Calculate the contribution of the nearest neighbors, add it to the current sample features through residuals, and then make predictions through the prediction module (similar to FFN)

TabR achieves the use of other sample information to assist in prediction through a retrieval mechanism similar to attention, but the computational cost is relatively high.



The process of TabR retrieval and neighbor fusion



The overall process of TabR

- The learning objective: soft-NN

$$\hat{y}_i = \sum_{(x_j, y_j) \in \mathcal{D}} p_{ij} y_j = \sum_{(x_j, y_j) \in \mathcal{D}} \frac{\exp\left(-\text{dist}^2\left(\phi(x_i), \phi(x_j)\right)\right)}{\sum_{(x_l, y_l) \in \mathcal{D}, x_l \neq x_i} \exp\left(-\text{dist}^2\left(\phi(x_i), \phi(x_l)\right)\right)} y_j$$

- Minimize the sum of negative log probability. Use soft-NN for prediction.

Architecture

- Set ϕ as MLP, and a single block is implemented by [Gorishniy et al., NIPS'21]
- One-hot encoding for categorical features
- PLR (lite) encoding for numerical features

Acceleration

Sample **a subset of training set** in a mini-batch to act as the neighbor candidates, while use **the whole training set** during the inference.

Evaluation of Tabular Models

Evaluation criteria for a single tabular dataset.

Classification Task

- **Accuracy:** Measure the proportion of samples whose predictions are correct among the total samples

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}}$$

- **F1:** The harmonic average of precision and recall, used to balance false positives and false negatives

$$\text{F1} = \frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

- **AUC:** Evaluate the model's ability to distinguish different categories at all possible thresholds

$$\text{AUC} = \int_0^1 \text{TPR}(\text{FPR}) d\text{FPR}$$

Regression Task

- **MSE:** Measure the average of the squares of the differences between the predicted value and the true value

$$\text{MSE} = \frac{1}{n} \sum_{i=0}^n (y_i - \hat{y}_i)^2$$

- **MAE:** The average of the absolute values that measure the difference between the predicted value and the true value is more robust and less affected by outliers

$$\text{MAE} = \frac{1}{n} \sum_{i=0}^n |y_i - \hat{y}_i|$$

- **R^2 :** The ability of a model to interpret data changes is measured. The closer the value is to 1, the better the fitting effect of the model

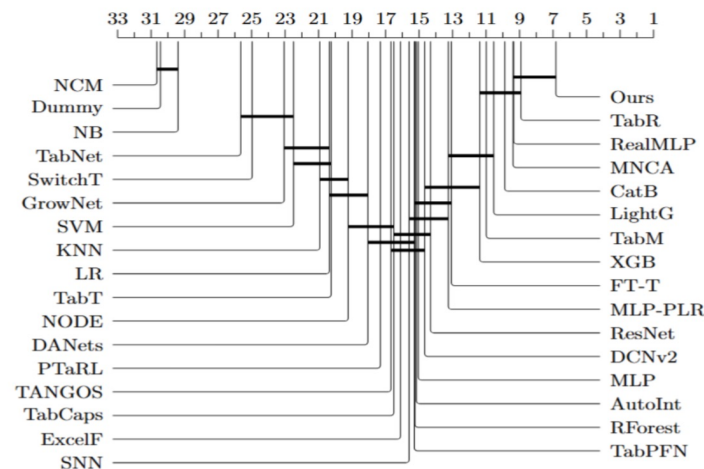
$$R^2 = 1 - \frac{\sum_{i=1}^n (y_i - \hat{y}_i)^2}{\sum_{i=1}^n (y_i - \bar{y}_i)^2}$$

Evaluation of Tabular Models

Evaluation criteria for multiple tabular datasets.

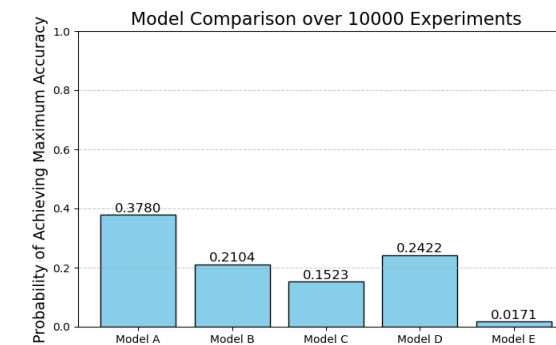
Average Rank

For each dataset, all methods are sorted, and the average ranking of all methods across all datasets is calculated. Generally, this is presented in conjunction with tests such as Wilcoxon-Holm



PAMA (Probability of Achieving the Maximum Accuracy)

The probability that a certain model becomes the one with the highest accuracy among R experimental comparisons of M models and algorithms



$$\text{PAMA}(m) = \frac{1}{R} \sum_{j=1}^R 1[m = \arg \max_{j \in M} \text{Acc}_{j,r}]$$



TALENT: A Tabular Analytics and Learning Toolbox

[\[Paper\]](#) [\[中文解读\]](#) [\[Docs\]](#)

<https://github.com/LAMDA-Tabular/TALENT>

It includes over 35 deep learning methods (including new ones such as NeurIPS'25/ICLR'25/ICML'25).

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort et al. *Scikit-learn: Machine Learning in Python*. JMLR 2011.

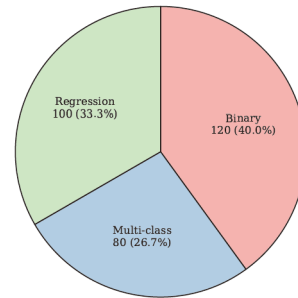
Si-Yang Liu, Hao-Run Cai, Qi-Le Zhou, Han-Jia Ye. *TALENT: A Tabular Analytics and Learning Toolbox*. JMLR 2025.

Han-Jia Ye, Si-Yang Liu, Hao-Run Cai, Qi-Le Zhou, De-Chuan Zhan. *A Closer Look at Deep Learning Methods on Tabular Datasets*. CoRR 2024.

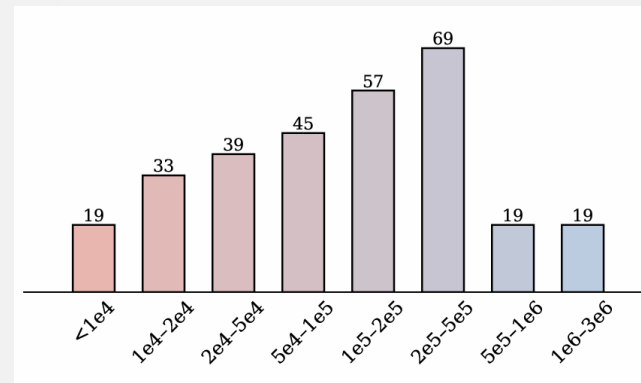
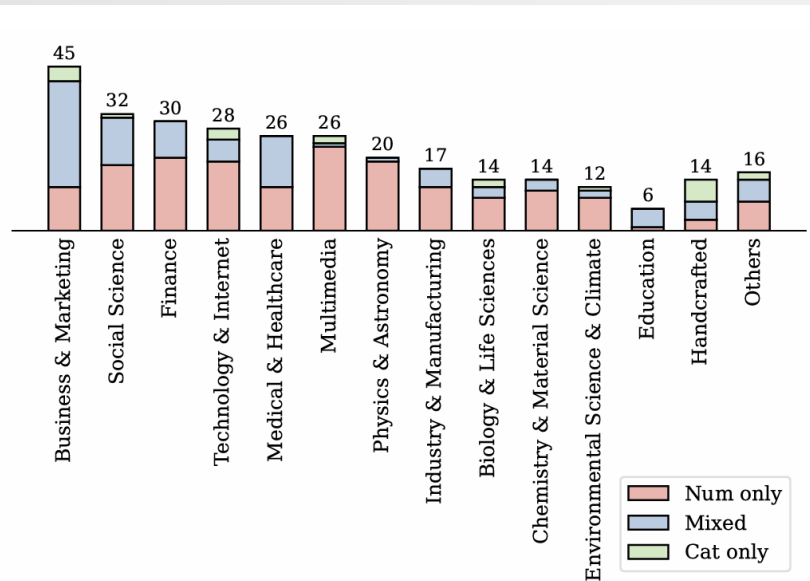
TALENT Benchmark

Evaluations on 300 tabular datasets

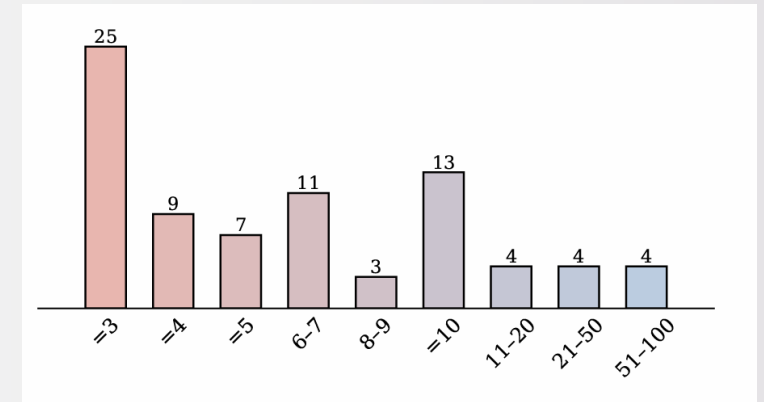
- Binary Classification: 120
- Multi-Class Classification: 80
- Regression: 100



Statistics over domains



Statistics over data sizes



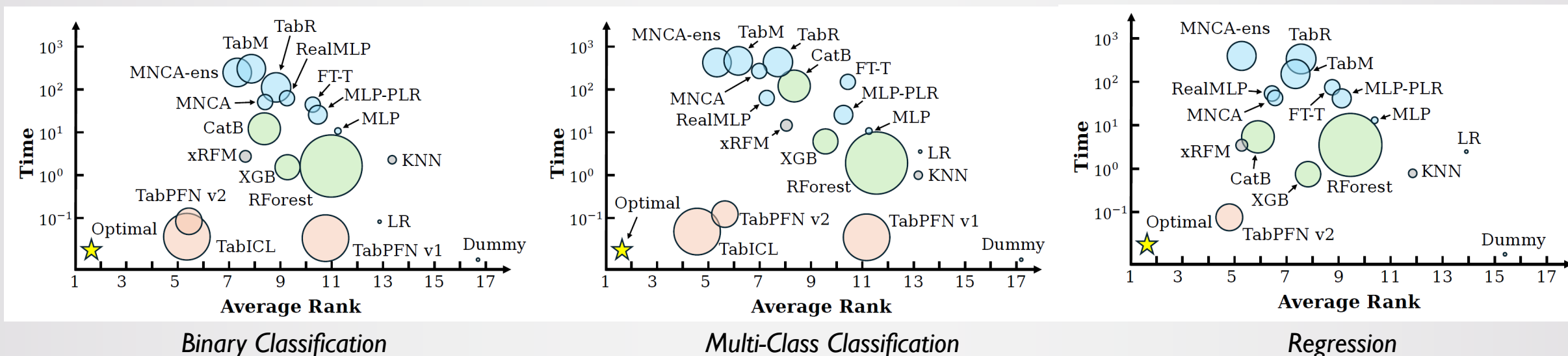
Statistics of classes in multi-class tasks

Tabular Benchmarks

In different benchmarks, there are generally different ways of data splitting and selection of hyperparameters

Data Split		
Split Method	TabArena	TALENT
Split the fixed data set and conduct multiple evaluations	-	64% train 16% validate 20% test - Take the average value after running 15 times with different seeds
Cross validation	- For datasets with a sample size of less than 2,500, repeat the 3-fold cross-validation 10 times. - Repeat 3 times for other datasets	-
Hyperparameters		
TabArena	- Except for some models, conduct a default parameter evaluation for the remaining models and then perform 200 random parameter searches - Limit the evaluation time for a group of parameters to no more than one hour	
TALENT	Without considering the time limit, search for 100 times in the parameter space and take the optimal parameter	

Evaluation



- Deep learning methods can achieve performance comparable to or even better than tree-based models;
- Compared to training separate models for each task, pre-trained models achieve stronger performance with lower computational cost.

Q&A



Presenter:

Dr. Xingxuan Zhang

Assistant Researcher

Department of Computer Science

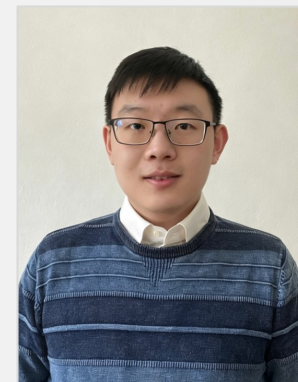
Tsinghua University

Email: xingxuanzhang@hotmail.com

Advances of Tabular Foundation Models



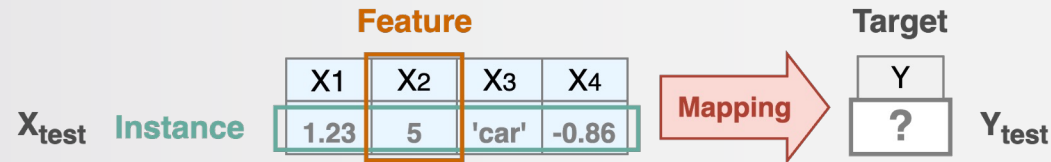
Presenter:
Dr. Xingxuan Zhang
Assistant Researcher
Department of Computer Science
Tsinghua University
Email: xingxuanzhang@hotmail.com



Presenter:
Dr. Shuyang Li
Research Scientist
Stable AI Technology Co., Ltd.
Email: lsy5215212@outlook.com

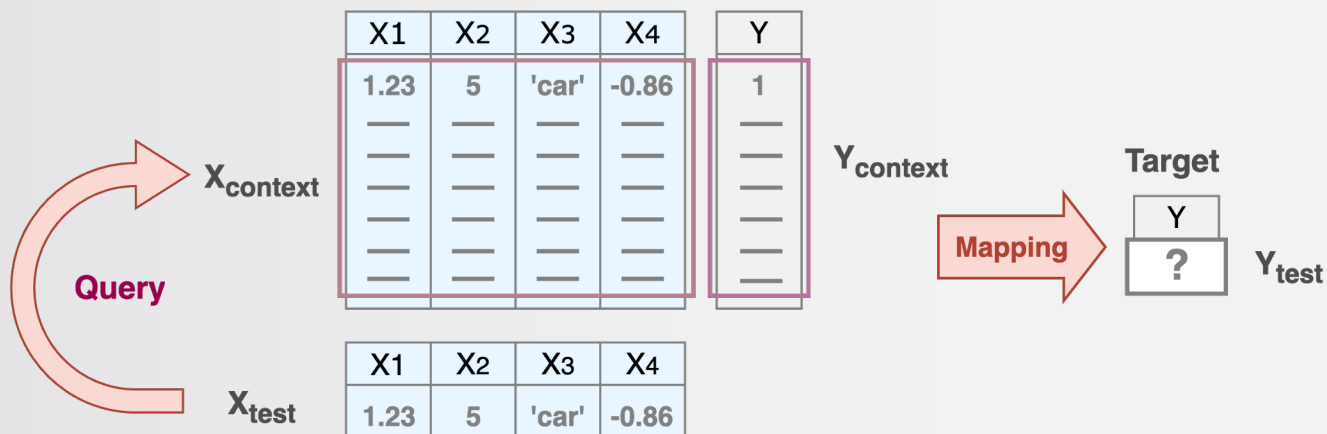
Starting with Tabular ICL

Conventional Tabular Modeling



$$y_{test} = f(x_{test})$$

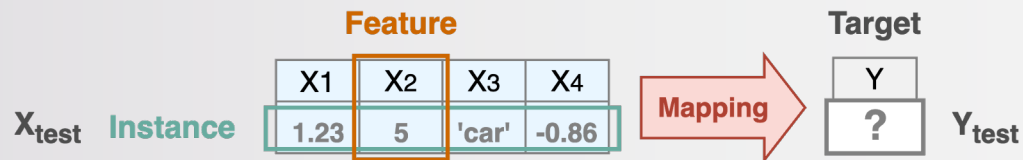
ICL-Based Tabular Modeling



$$y_{test} = f(x_{test} | x_{context}, y_{context})$$

Starting with Tabular ICL

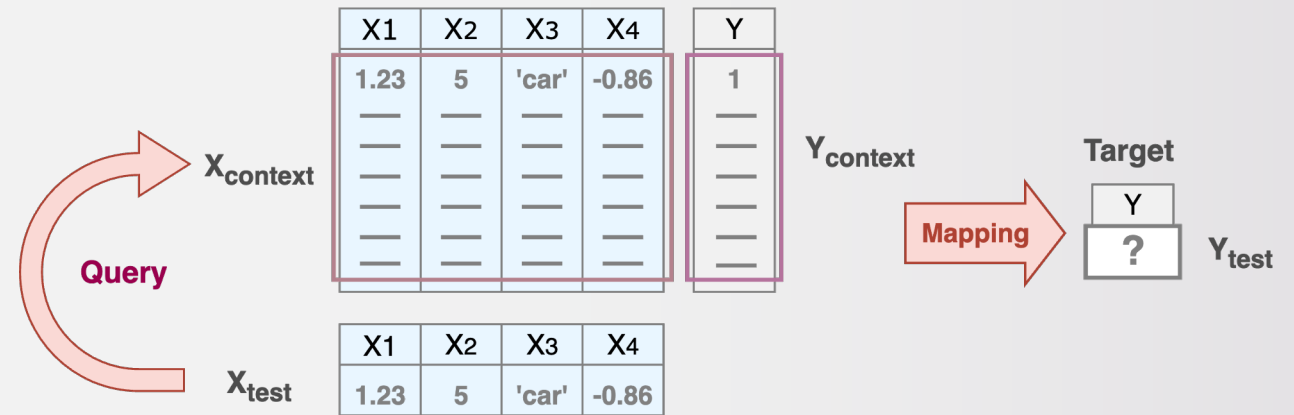
Conventional Tabular Modeling



Train: $y_{\text{train}} = f(x_{\text{train}})$

Test: $y_{\text{test}} = f(x_{\text{test}})$

ICL-Based Tabular Modeling



Train: $y_{\text{train}} = f(x_{\text{train}} \mid x_{\text{context}}^{\text{train}}, y_{\text{context}}^{\text{train}})$

Test: $y_{\text{test}} = f(x_{\text{test}} \mid x_{\text{context}}^{\text{test}}, y_{\text{context}}^{\text{test}})$

How Can We Define ICL Problems

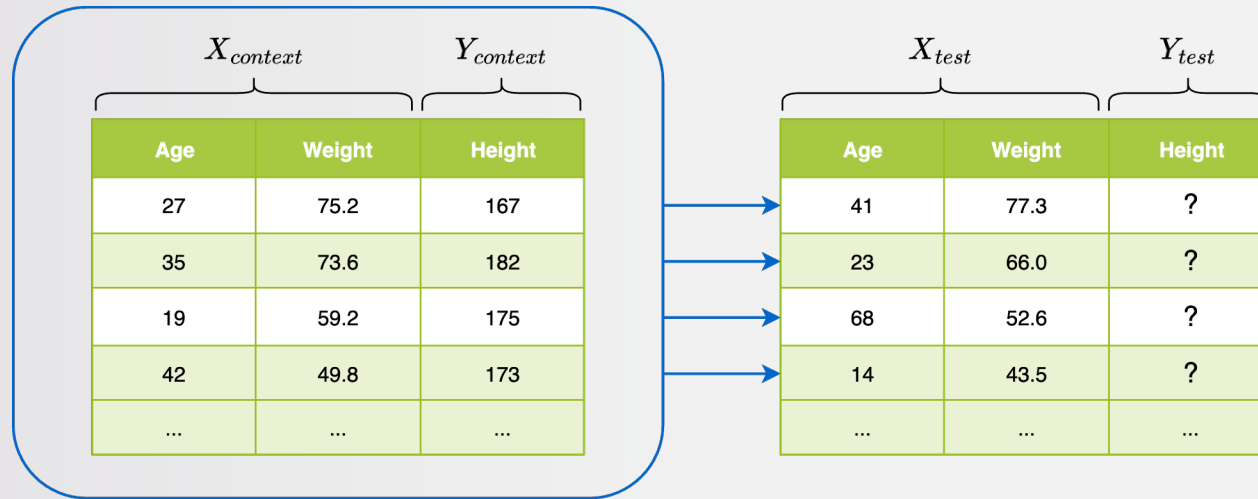
$$y_{test} = f(x_{test} | x_{context}, y_{context})$$

y_{test} and $y_{context}$ can either be integers (for classification) or floating-point numbers (for regression).

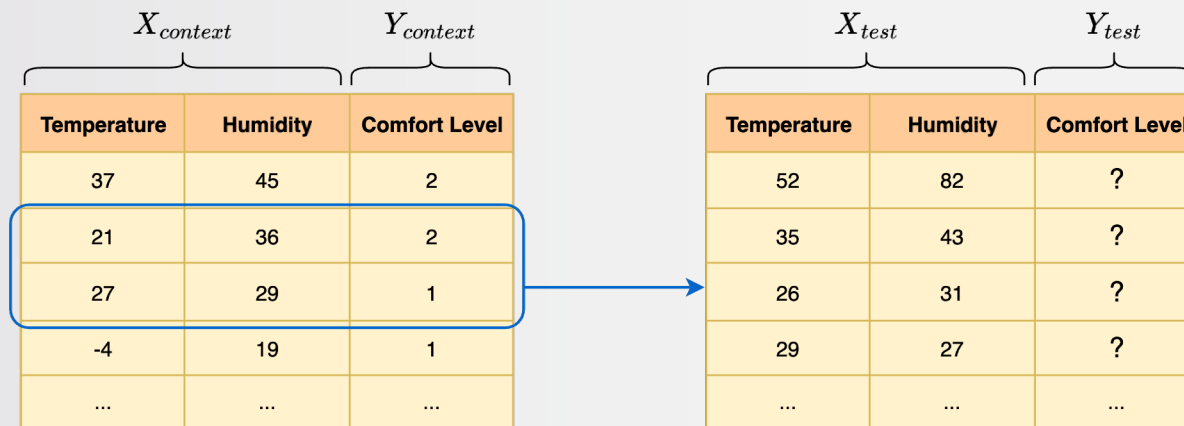
x_{test} and $x_{context}$ can be:

- Raw features (one or more rows of tabular data);
- Features extracted through a feature engineering pipeline (manually defined or ML-driven);
- Representations learned by deep learning models.

How Can We Define ICL Problems



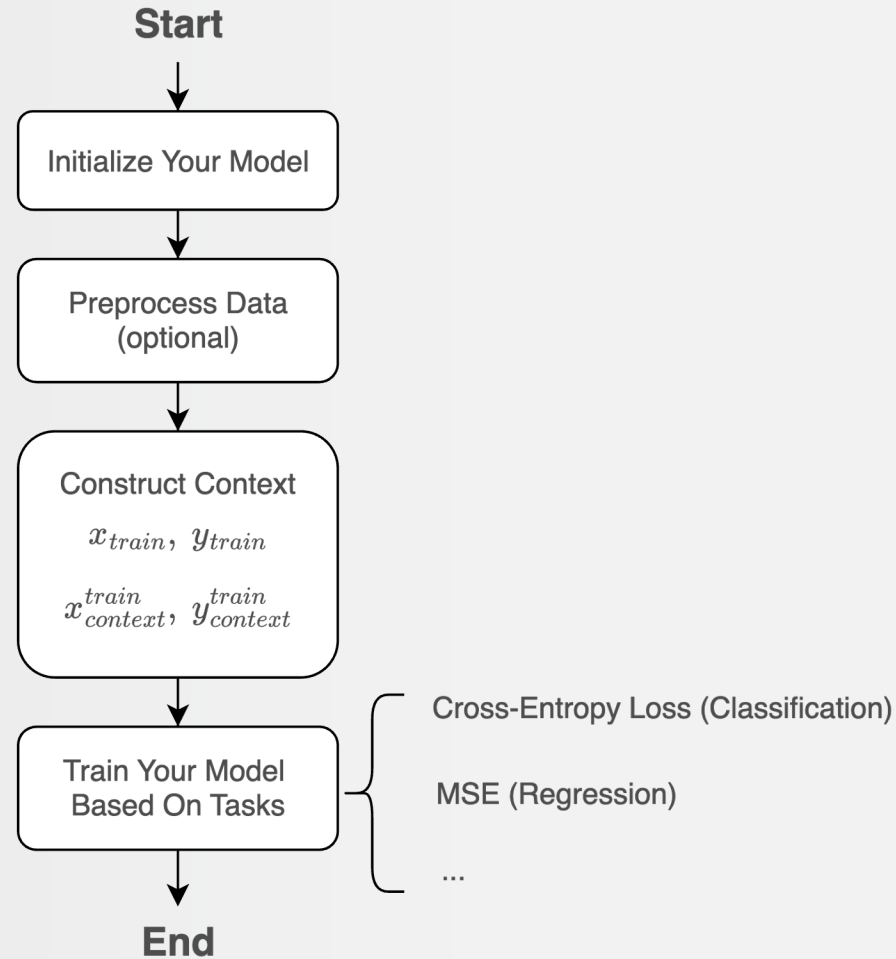
Test samples share the same context



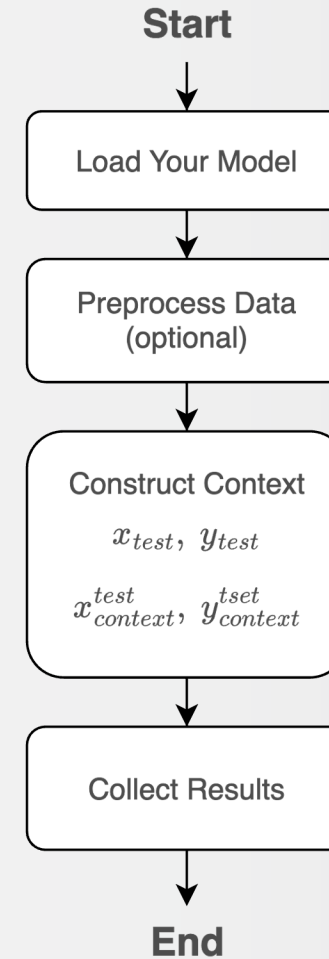
Test samples have different context

A Very Brief Guide to Building Your Own ICL Models

Training Roadmap



Test Roadmap



Compared to conventional modeling methods, ICL-based methods brings some benefits:

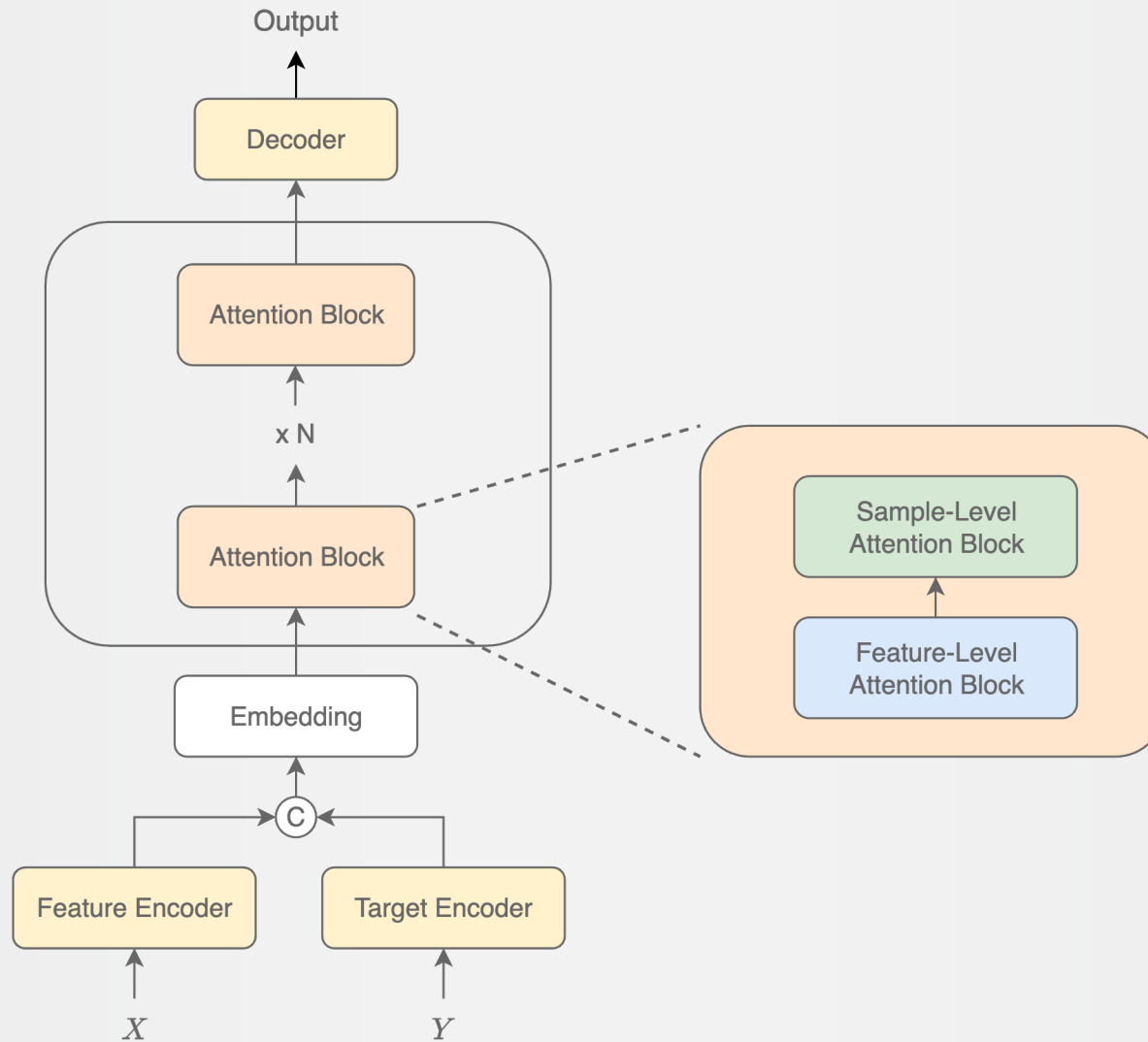
- **Improved Generalization:** Instead of learning the specific dependencies of the training data, ICL-based methods focus on developing the ability to find similar instances from a knowledge base given a specific instance. This approach enables these models to have better generalization capabilities;
- **No Retraining Required:** We can easily define and modify the behavior of ICL-based approaches by adjusting the target or the provided context, all without the need for retraining;
- **Robust OOD Adaptation:** ICL-based methods can better adapt to OOD cases, as shifts in distribution and dependencies can be encoded through the updated context, making them highly beneficial for evolving online systems or similar scenarios.

Benefits and Important Notes

For ICL-based methods, there are also some important notes:

- **Contextual Representation Complexity:** While ICL can use the provided context, it's important that the context is well-defined. Ambiguous or insufficient context can lead to poor performance;
- **Scalability:** For large-scale tabular datasets with millions of rows and features, generating the right context or efficiently processing it within ICL frameworks could become computationally intensive;
- **Training Difficulty:** ICL-based methods are more challenging to train compared to non-ICL-based methods, requiring careful tuning of hyperparameters during the training process.

A Simple Design of Tabular Foundation Models



Task-Specific Training:

- Strong performance on individual tasks;
- Limited generalization across tasks.

Multitask Learning:

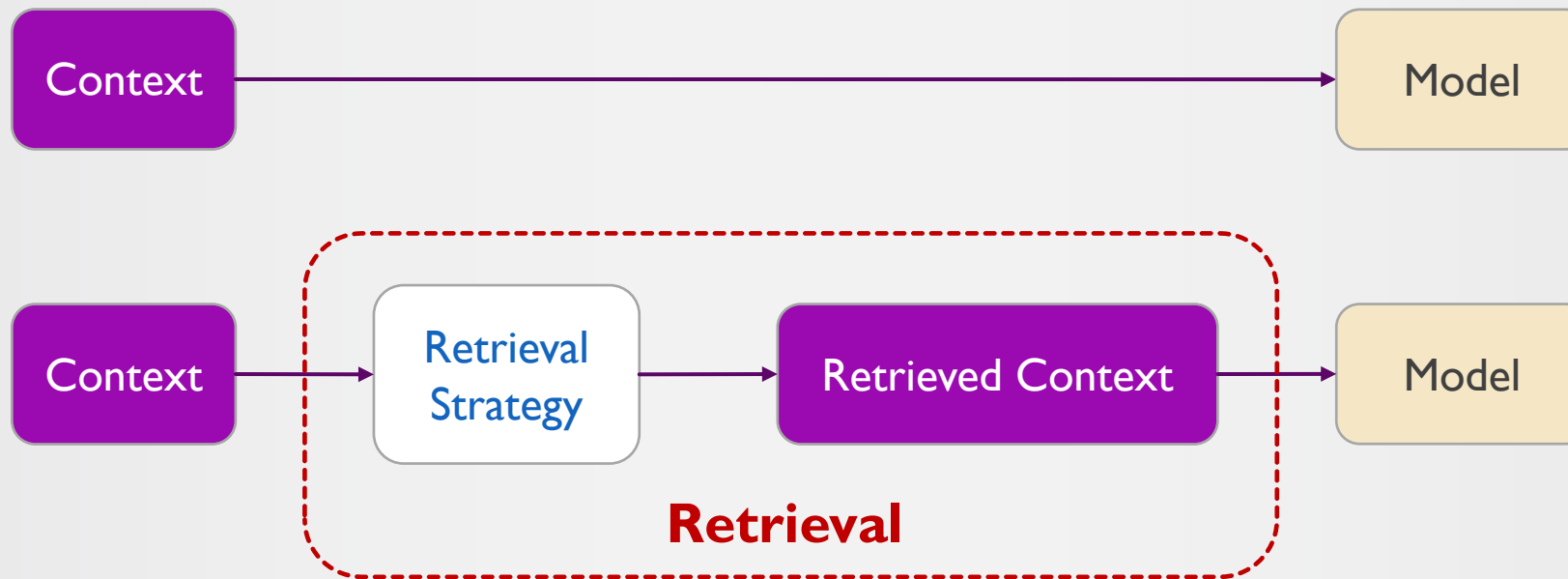
- Learns shared representations across tasks;
- Improves data efficiency and transferability.

Task-specific models are highly optimized but limited in scope, often failing to generalize beyond the task they were trained on.

Multitask learning enables a model to learn from multiple tasks simultaneously, leading to improved representation learning, better generalization, and more efficient use of data. However, multitask learning also introduces new challenges, as different tasks may have conflicting objectives.

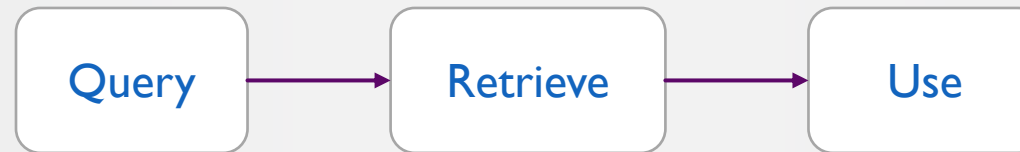
What is Retrieval

Retrieval is the process of selecting relevant samples from a dataset given a query, and the goal is to find similar instances based on feature similarity or learned representations.



Retrieved samples act as external knowledge during inference.

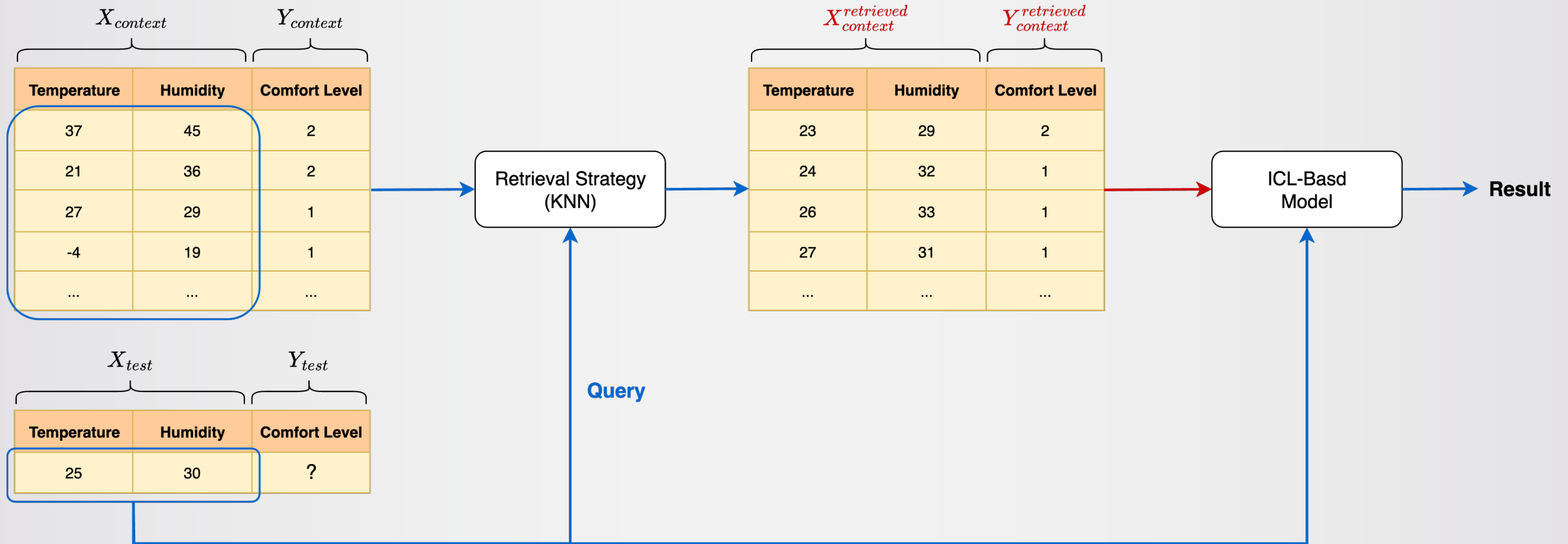
A basic retrieval pipeline can be described as follows:



Taking KNN as an example:

- **Step 1:** Given a query, compute similarity scores with dataset samples;
- **Step 2:** Select top-k relevant instances;
- **Step 3:** Use retrieved samples to assist prediction.

An Illustrative Example



Challenges in Data Collection

For ICL-based foundation models, diverse and high-quality pretraining data is the key to have a good model. For tabular foundation models, high quality data is difficult to obtain:

- **Limited Availability:** Tabular data is often proprietary or domain-specific;
- **Data Labeling:** Requires expert knowledge for accurate labeling;
- **Data Consistency and Cleanliness:** Missing values, errors, and outliers are common;
- **Privacy Concerns:** Sensitive data in sectors like healthcare and finance.

Why Synthetic Data Is the Solution

- **Data Availability:** Real-world tabular data are limited by privacy, accessibility, and heterogeneity constraints;
- **Controlled Diversity:** Synthetic data enables controlled exploration of diverse statistical and causal structures;
- **Scalable Pretraining:** Large-scale generation facilitates pretraining without reliance on labeled data;
- **Bias Mitigation:** Synthetic data helps reduce overfitting to specific domains or data distributions;
- **Causal & Structural Richness:** Allows embedding known generative mechanisms, correlations, and causal dependencies.

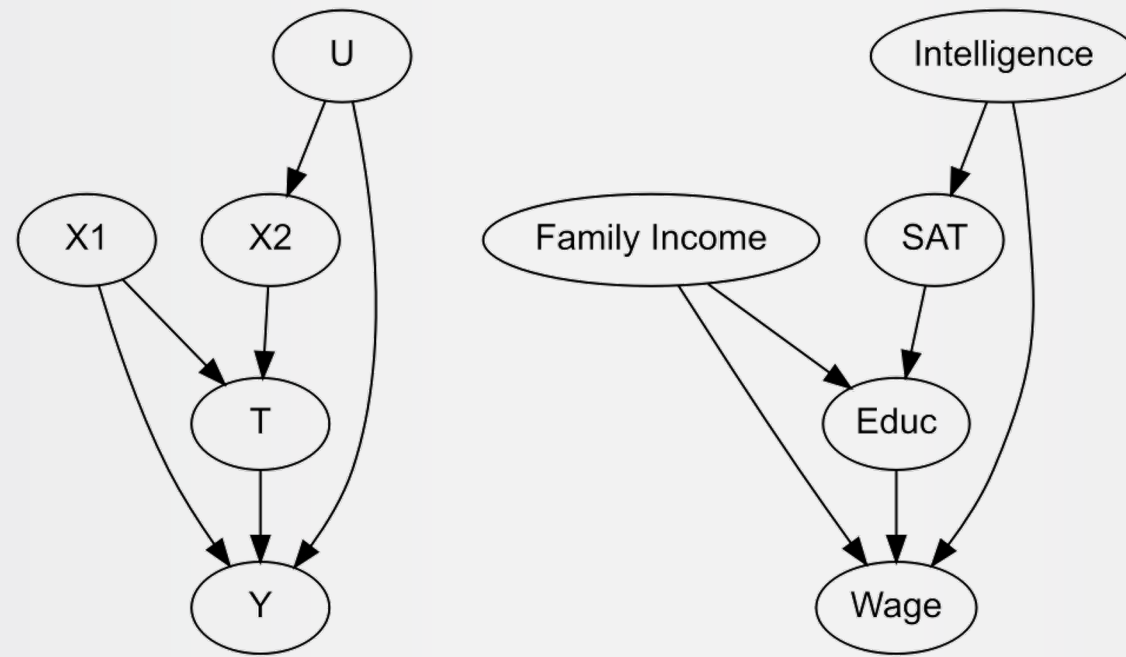
How to Generate Synthetic Data

In conclusion, there are several popular ways to generate synthetic data:

- **Generative Models:** Use machine learning models like generative adversarial networks or variational autoencoders to learn from real data and generate synthetic samples that preserve complex relationships and distributions;
- ➔ • **Rule-Based Generation:** Use predefined rules or logic to generate data that follows a specific pattern or relationship. This method is commonly used for simple or structured data;
- **Data Augmentation:** Instead of generating data from scratch, data augmentation techniques can be used to create new data by applying transformations to existing data.

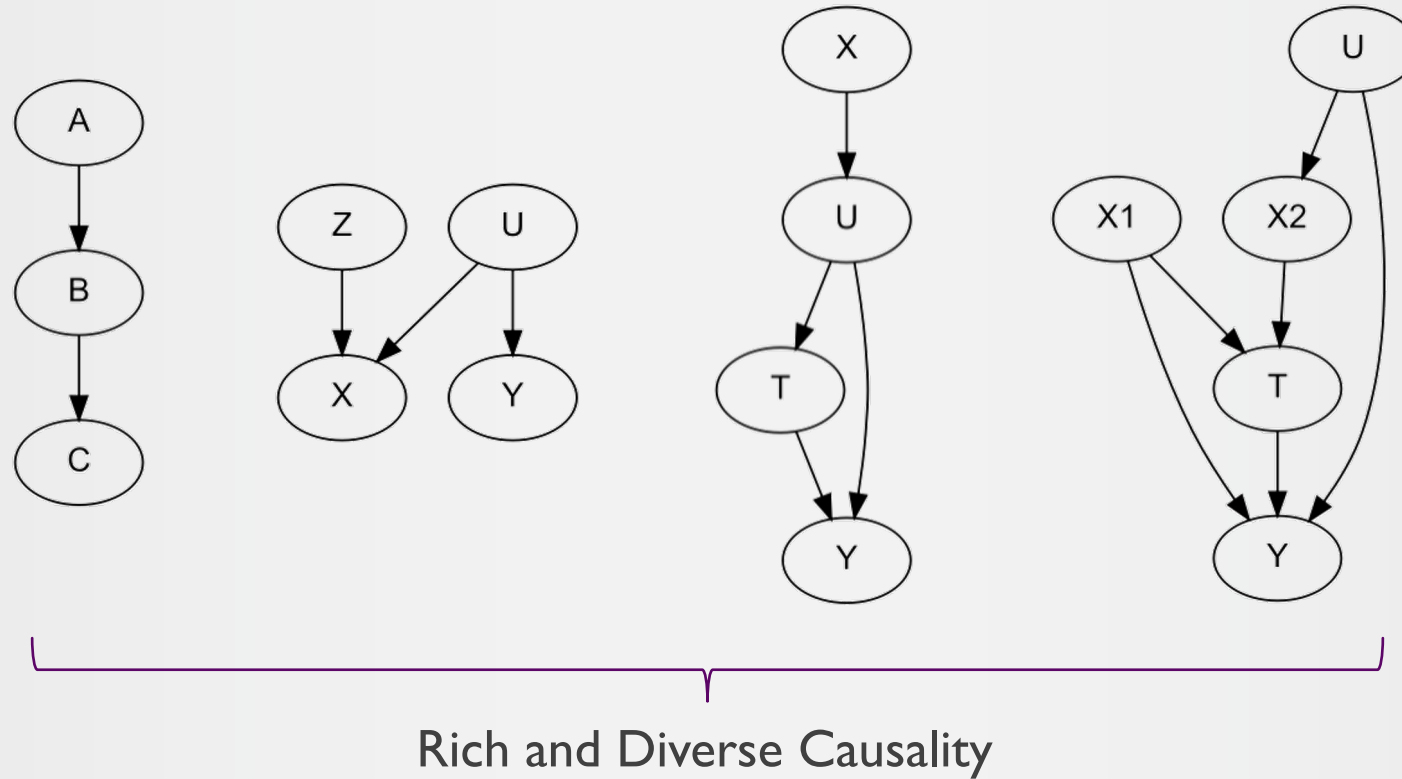
Causality: An Overlooked Pillar of Real-World Data

An example of causality:



Causality is common in the real world — yet rarely captured in tabular models

Causality: An Overlooked Pillar of Real-World Data



Universal causal representation is central to advancing synthetic datasets

Structural Causal Models (SCMs)

A SCM is a data structure is a data structure composed of a [Directed Acyclic Graph \(DAG\)](#) and [variables](#). The DAG defines the relationships between variables, while the variables themselves are the data stored at each node.

From a mathematical point of view, a SCM is a triple (N, X, F) where:

- N is a set of exogenous random variables $\{N_1, N_2, \dots\}$;
- X is a set of endogenous random variables $\{X_1, X_2, \dots\}$;
- F is a set of functions $F = \{f_{X_1}, f_{X_2}, \dots\}$ defining an endogenous variable X_i in terms of N_i and other endogenous variables X_j .

Structural Causal Models (SCMs)

DAG is an ideal tool for encoding causal relationships

Examples:

Let $N_1, N_2, N_3 \sim \mathcal{N}(0, 1)$ iid.

SCM:

$$X_1 = N_1$$

$$X_2 = N_2$$

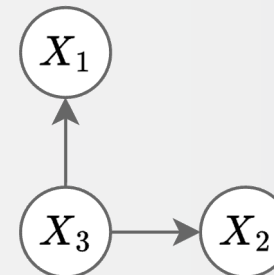
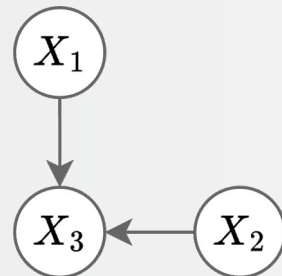
$$X_3 = X_1 + X_2 + N_3$$

$$X_1 = X_3 + N_1$$

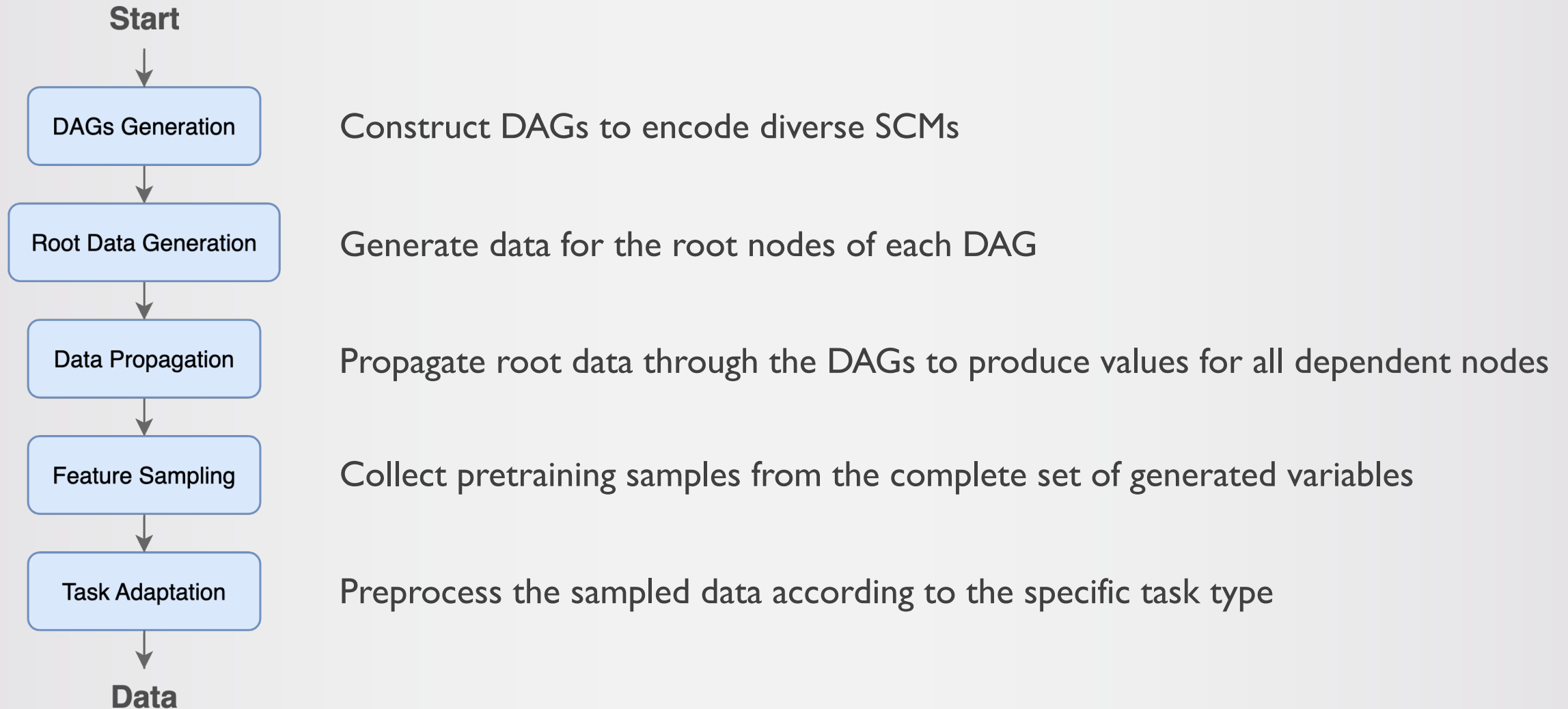
$$X_2 = X_3 + N_2$$

$$X_3 = N_3$$

DAG:



SCM-Based Data Generation



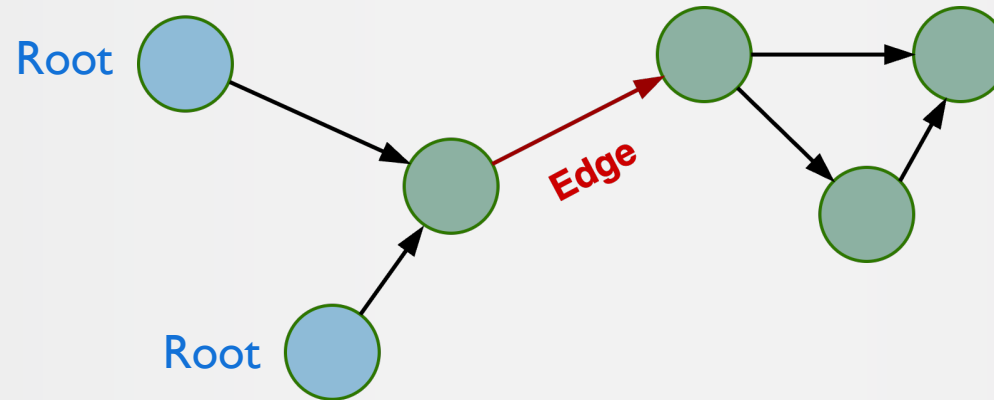
Step I: DAGs Generation – Graph

The generation of DAGs forms the foundation for defining dependencies between features, and various graph types can be constructed, including:

- Random DAG (Erdős–Rényi DAG);
- Tree-based DAG;
- BFS / DFS Trees;
- Growing Network with Redirection (GNR);
- ...

The selection of DAG architecture is an important hyperparameter of data generation.

Step I: DAGs Generation – Edge Functions

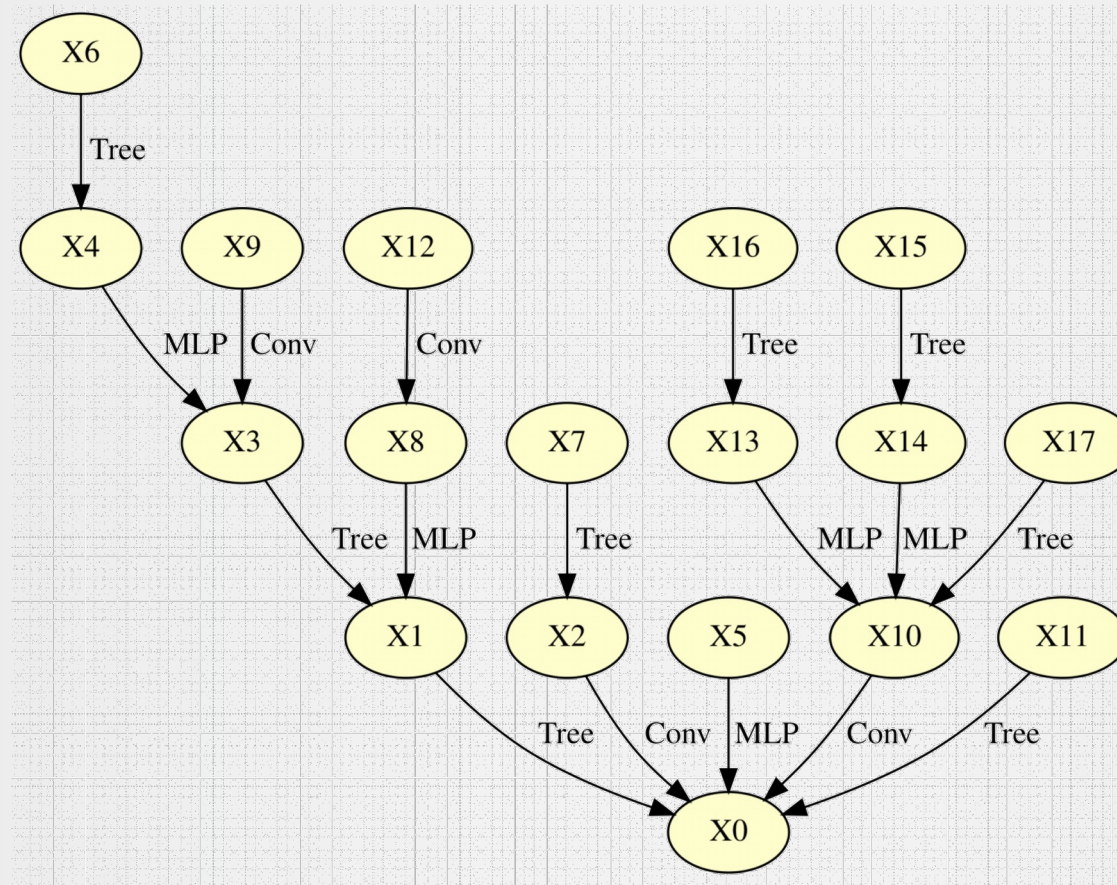


The DAG encodes dependencies among nodes (features), while the mathematical relationships are specified through **edge functions**. The widely used edge functions include:

- MLP (Multilayer Perceptron);
- Decision Tree.

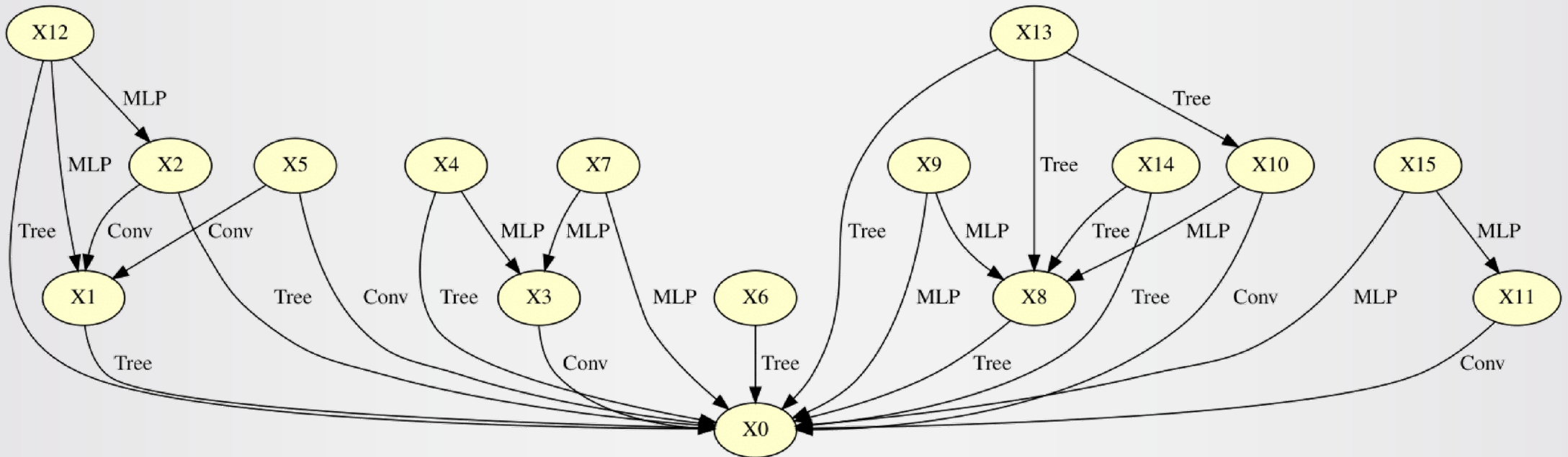
The activation functions are sampled from a predefined pool, which includes common choices such as ReLU, Tanh and others.

Examples: Generated DAGs



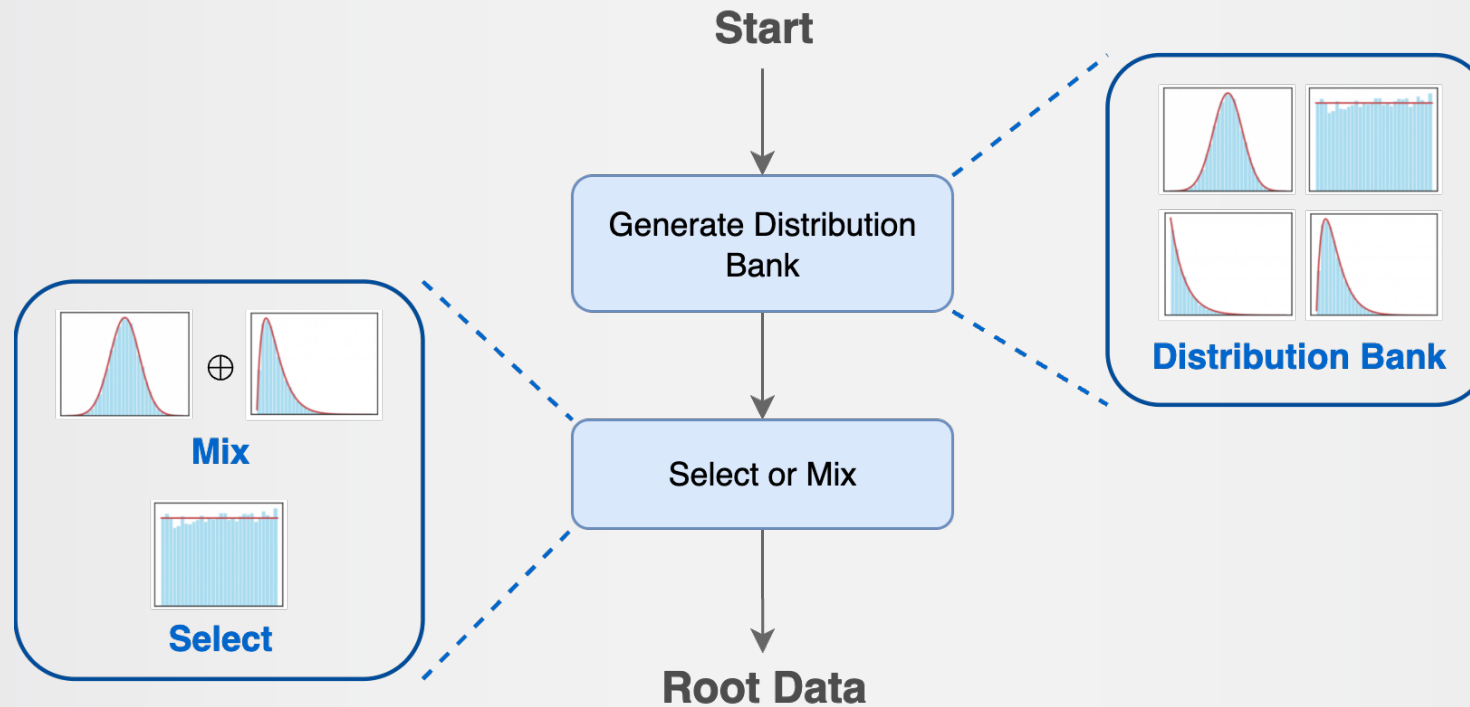
Tree-Like DAG

Examples: Generated DAGs



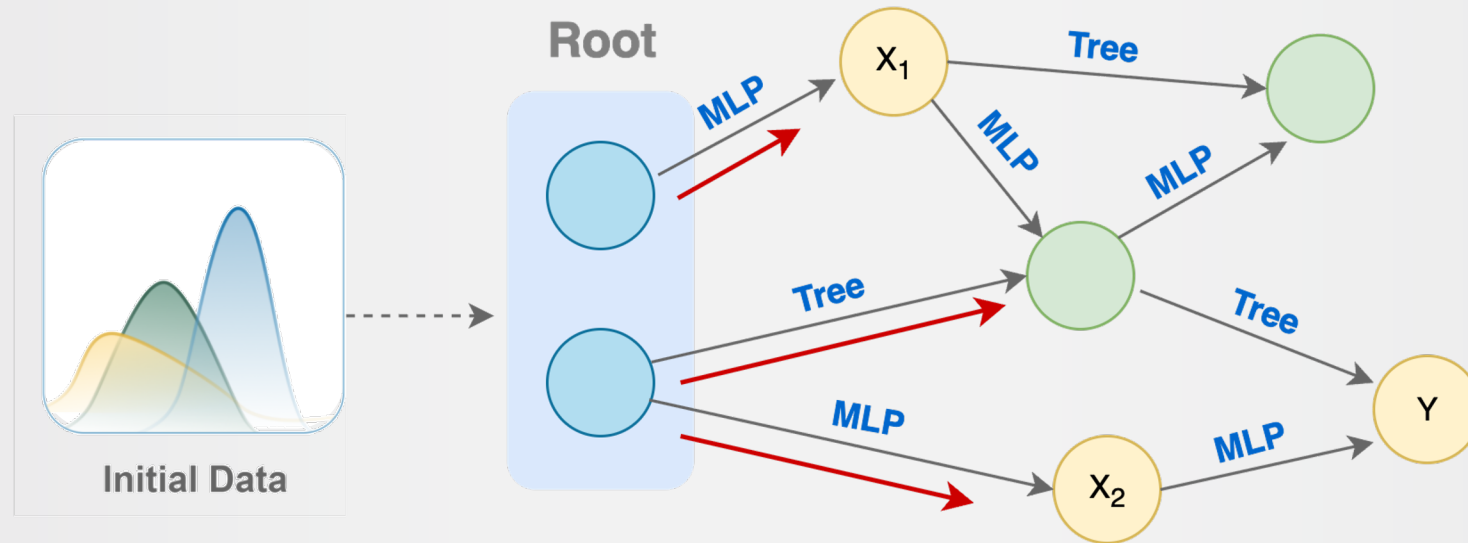
Ring-Like DAG

Step 2: Root Data Generation



The root data can be generated through sampling a specific distribution or mix several distributions

Step 3: Data Propagation



Initial data are assigned to each root node, and data for subsequent nodes are derived by propagating the root data along the DAG.

Step 4: Data Sampling

Once each node is assigned its data, the next step is to sample the input features X and the target Y . Depending on the design, different sampling strategies can be applied; some examples are as follows:

- **Random Sampling**: Randomly sample X and Y ;
- **DAG-Constrained Sampling**: Sample X and Y only if they are reachable via a directed edge;
- **Neighborhood Sampling** : Sample X and Y only if they are directly connected.

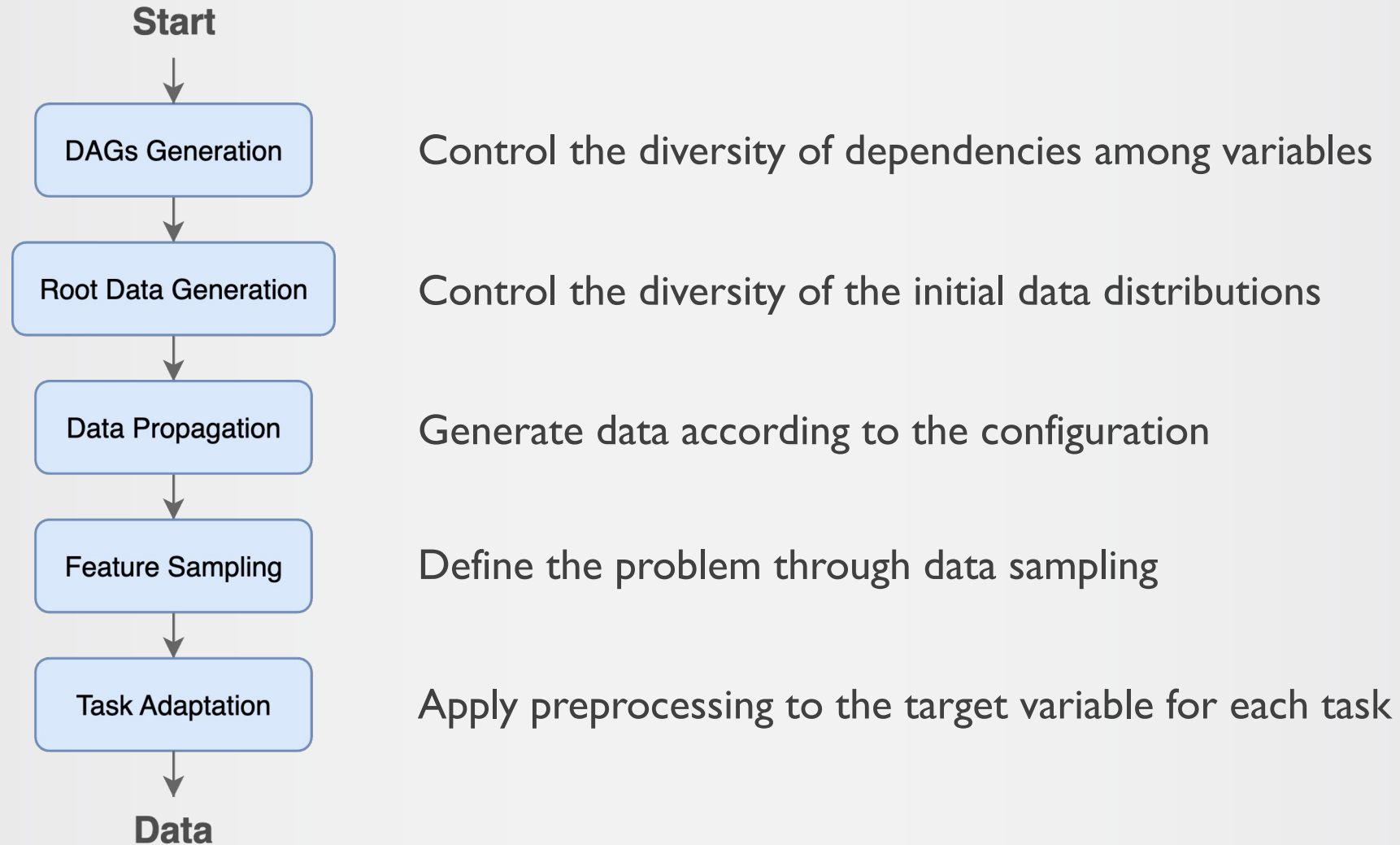
The sampling strategy is crucial for the solvability of the defined problem.

Final Step: Task Adaptation

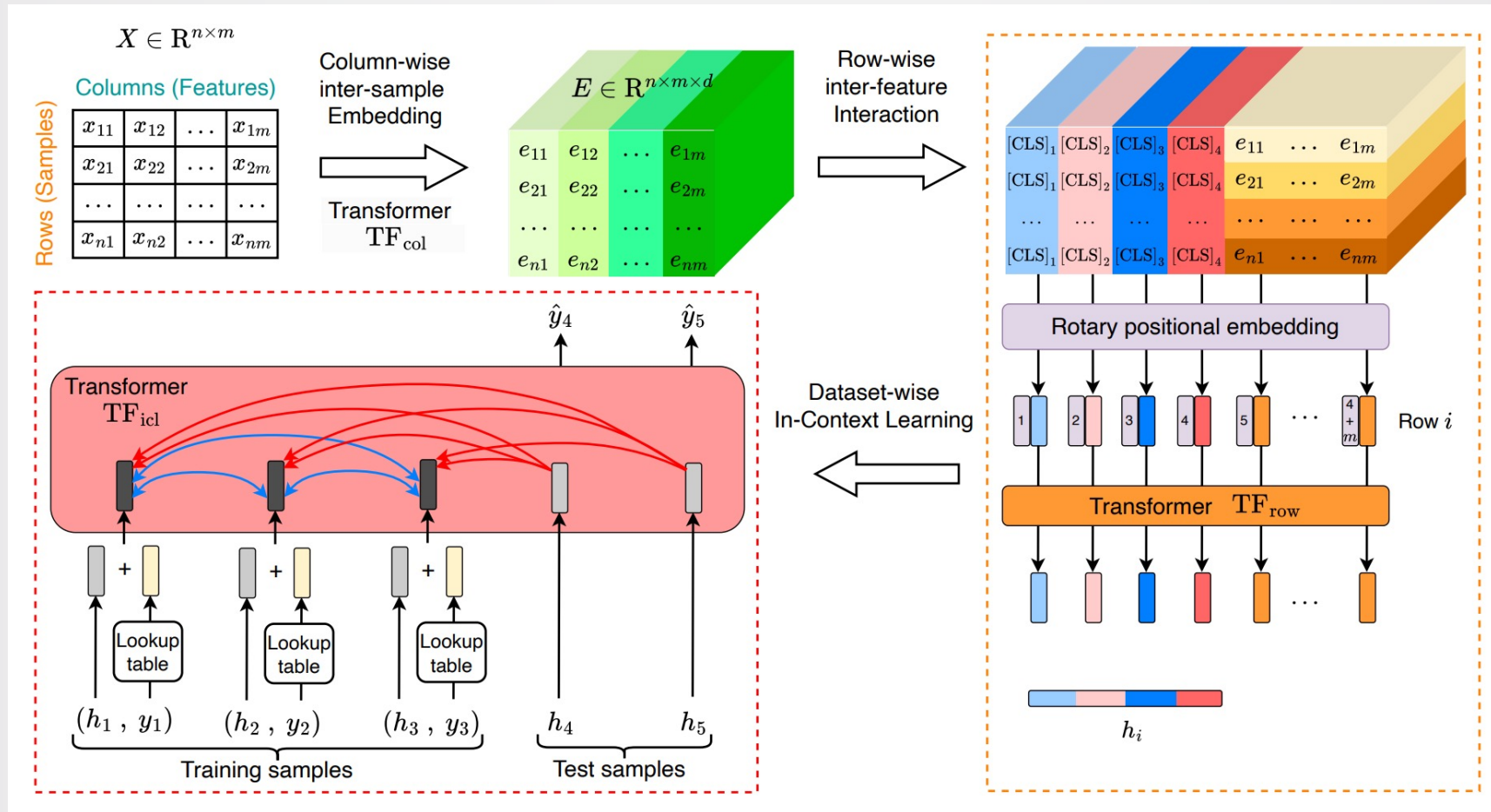
After defining the problem through data sampling, the final step is to adapt the target to the specific task. For example:

- **Classification**: the target variable Y is subsequently discretized into categorical values;
- **Regression**: If the target variable y is a general continuous variable, no mandatory processing is required. However, it could be beneficial to scale its magnitude or apply appropriate transformations.

Recap: SCM-Based Data Generation



TabICL: Architecture



Qu, Jingang, et al. "Tabicl: A tabular foundation model for in-context learning on large data." arXiv preprint arXiv:2502.05564 (2025).

TabICL extends TabPFN by adopting a tree-based SCM for data generation and introducing a richer set of activation functions inspired by TabPFN's design, including:

- ReLU
- ReLU6 (Krizhevsky)
- SELU (Klambauer et al., 2017)
- SiLU (Hendrycks & Gimpel, 2016)
- Softplus
- Hardtanh(x) = $\max(-1, \min(1, x))$
- Signum function
- Sine
- RBF(x) = $\exp(-x^2)$
- Exponential function
- $f(x) = \sqrt{|x|}$
- $f(x) = 1_{|x| \leq 1}$
- $f(x) = x^2$
- $f(x) = |x|$
- A random function $f(x) = \phi(x)^\top \mathbf{z}$, where $\mathbf{z} \sim \mathcal{N}(0, 1)$ and the feature map ϕ is defined randomly as
$$\phi(x) := \frac{\mathbf{w}}{\|\mathbf{w}\|_2} \odot \sin(\mathbf{a}x + \mathbf{b}) \in \mathbb{R}^N,$$
$$N := 256,$$
$$b_i \sim \mathcal{U}[0, 2\pi],$$
$$a_i \sim \mathcal{U}[0, N],$$
$$w_i := a_i^{-\exp(u)},$$
$$u \sim \mathcal{U}[0.7, 3.0].$$

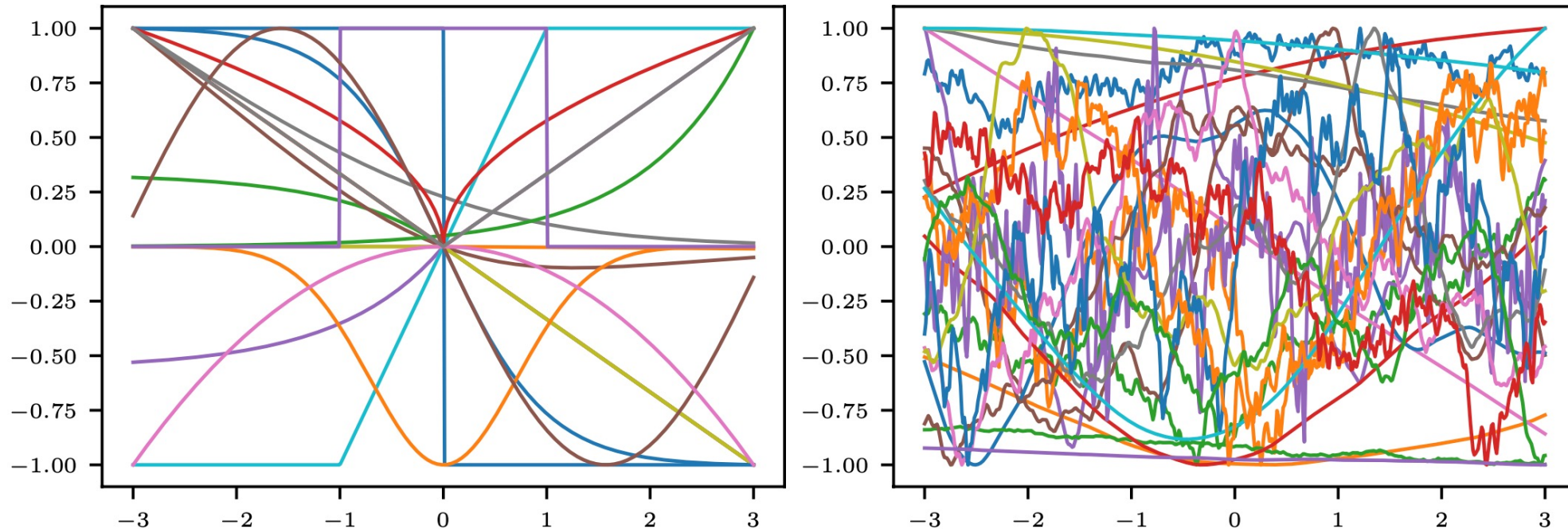
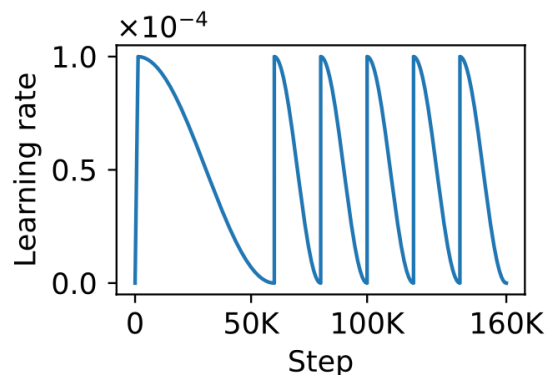
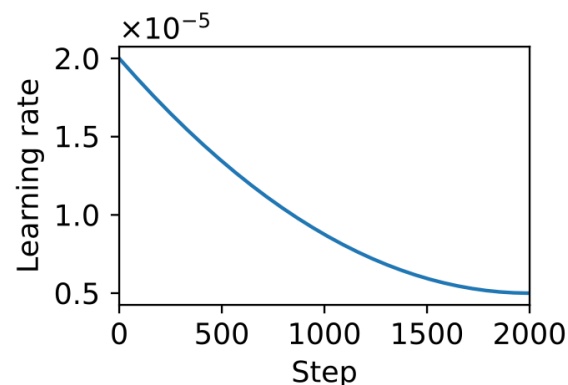


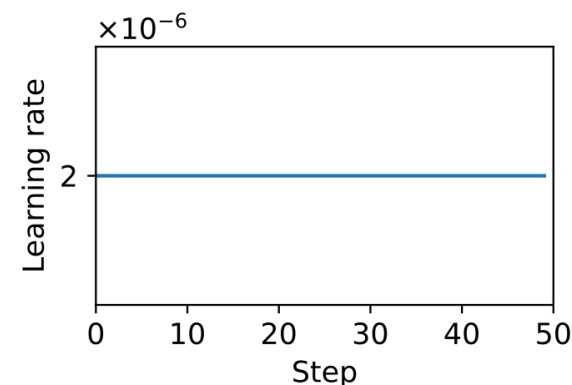
Figure C.1. Activation functions from the SCM prior. Left: Non-random activation functions from the SCM prior without standardization and random rescaling. Activation functions are randomly flipped along the x - and/or y -axis to fully utilize the space in the plot. Right: Random instantiations of the random activation function from the SCM prior, including the automatic standardization of the inputs.



(a) Cosine decay with restarts for stage 1



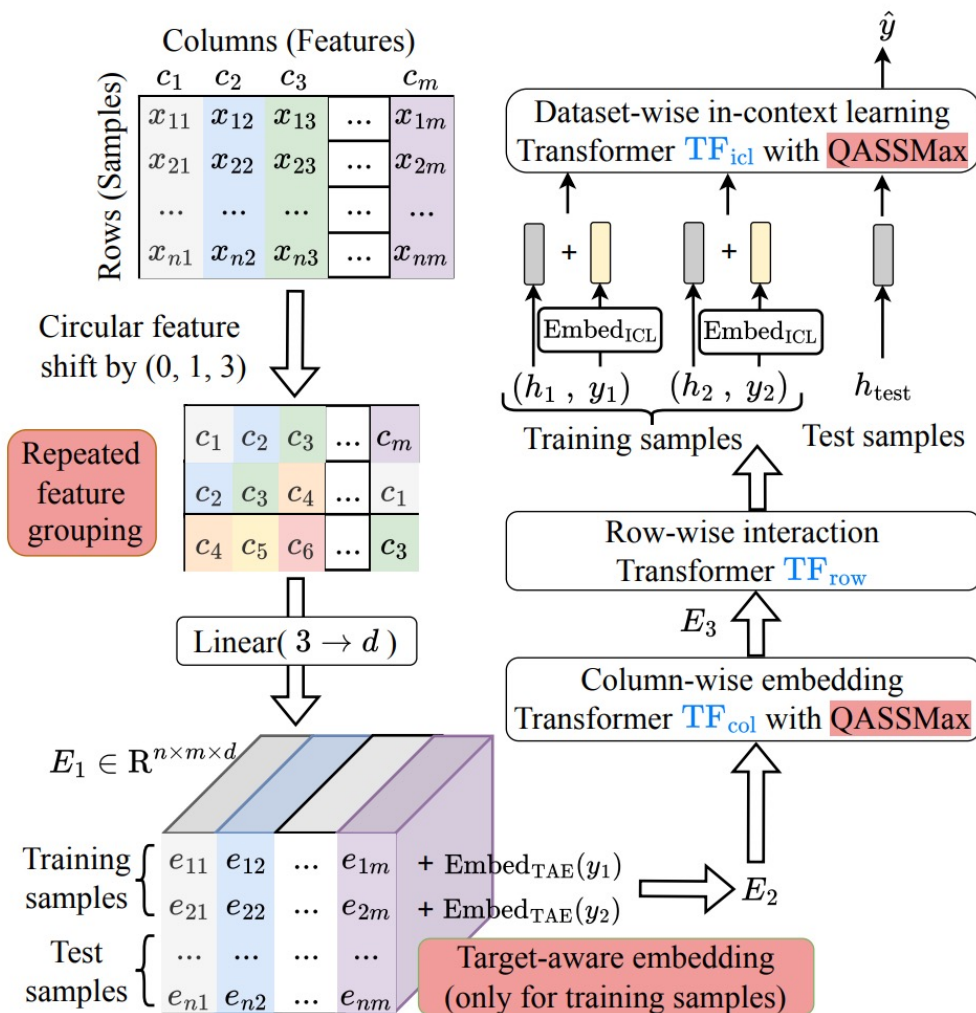
(b) Polynomial decay for stage 2



(c) Constant learning rate for stage 3

Figure E.1. Learning rate schedules for 3 pretraining stages.

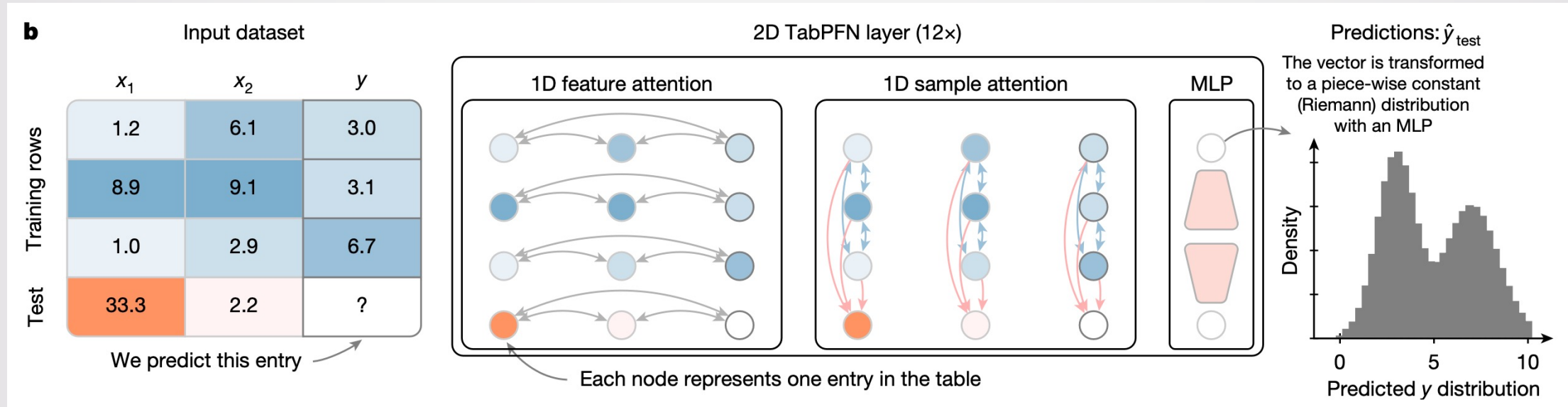
1. $N_B = 4$ with a fixed size of 1,024 the first 160K steps.
2. $N_B = 1$ with the size randomly drawn from a log-uniform distribution between 1K and 40K over 2K steps. Activation checkpointing is enabled for datasets exceeding 10K samples, and we accordingly reduce the number of features to avoid out-of-memory issues.
3. $N_B = 1$ with the size uniformly sampled between 40K and 60K for 50 steps, training only TF_{icl} while all other components remain frozen.



On the top of TabICL, TabICLv2 makes a couple of modifications:

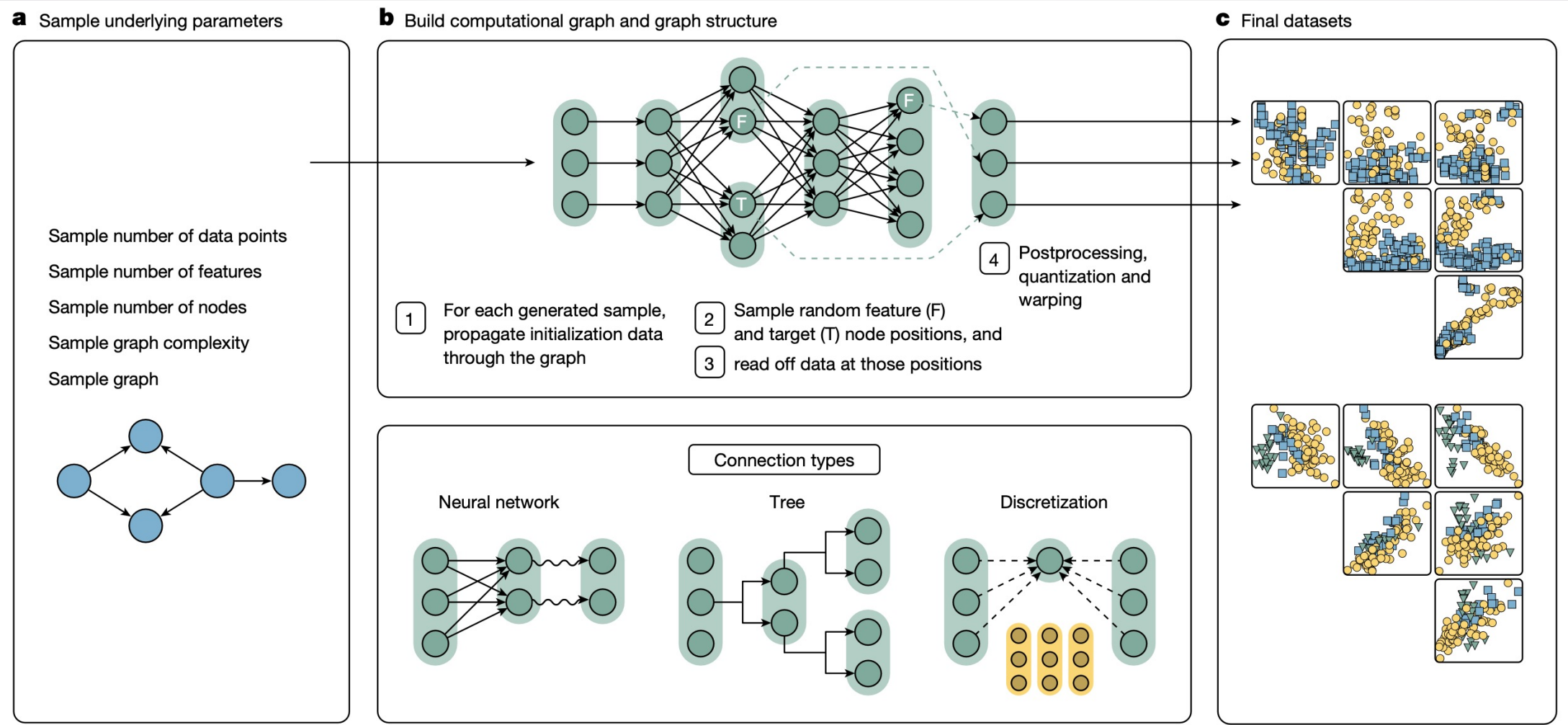
- It introduces a new, more diverse synthetic dataset generator for pretraining. Focuses on covering a wider range of tabular “data regimes” (different feature types, noise patterns, dataset sizes) by introducing richer edge functions and mapping functions;
- It adds a new scalable softmax variant in the attention module (QASSMax), which improves stability and performance on larger tables and longer sequences. This design reduces the need for prohibitively expensive long-context pretraining;

TabPFNv2: Architecture



TabPFN v2 is an encoder-only Transformer for tabular in-context learning that processes preprocessed tabular data as sequences of feature-label tokens, embeds them with feature, label, and positional embeddings, passes them through stacked self-attention and feedforward Transformer encoder layers, and outputs predictions for test tokens.

TabPFNv2: Data Generation



Regarding the edge functions, TabPFNv2 used three different types, including:

- **Small Neural Networks**: it can be either a fully-connected layer or a MLP;
- **Categorical Feature Discretization**: generates categorical features from numerical vectors at each node using a KNN-like approach;
- **Decision Trees**: to incorporate structured, rule-based dependencies.

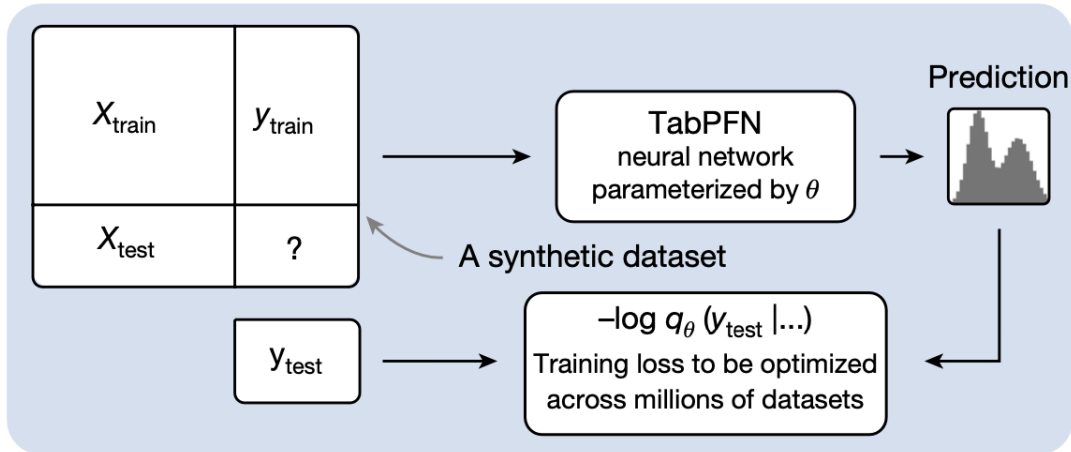
The activation functions are uniformly sampled from **identity, logarithm, sigmoid, absolute value, sine, hyperbolic tangent, rank operation, squaring, power functions, smooth ReLU, step function** and **modulo operation**.

Target generation:

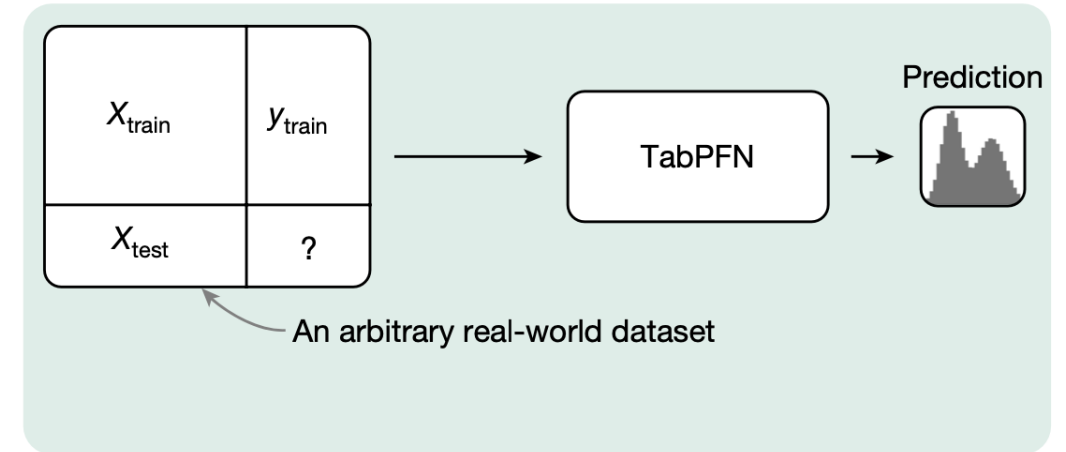
- **Regression:** To generate target labels for regression tasks, the authors select a randomly chosen continuous feature without post-processing;
- **Classification:** For classification labels, the authors select a random categorical feature that contains up to 10 classes. Thus, natively TabPFNv2 is limited to predicting at most 10 classes.

TabPFNv2: Pretraining

TabPFN is trained on synthetic data to take entire datasets as inputs and predict in a forward pass



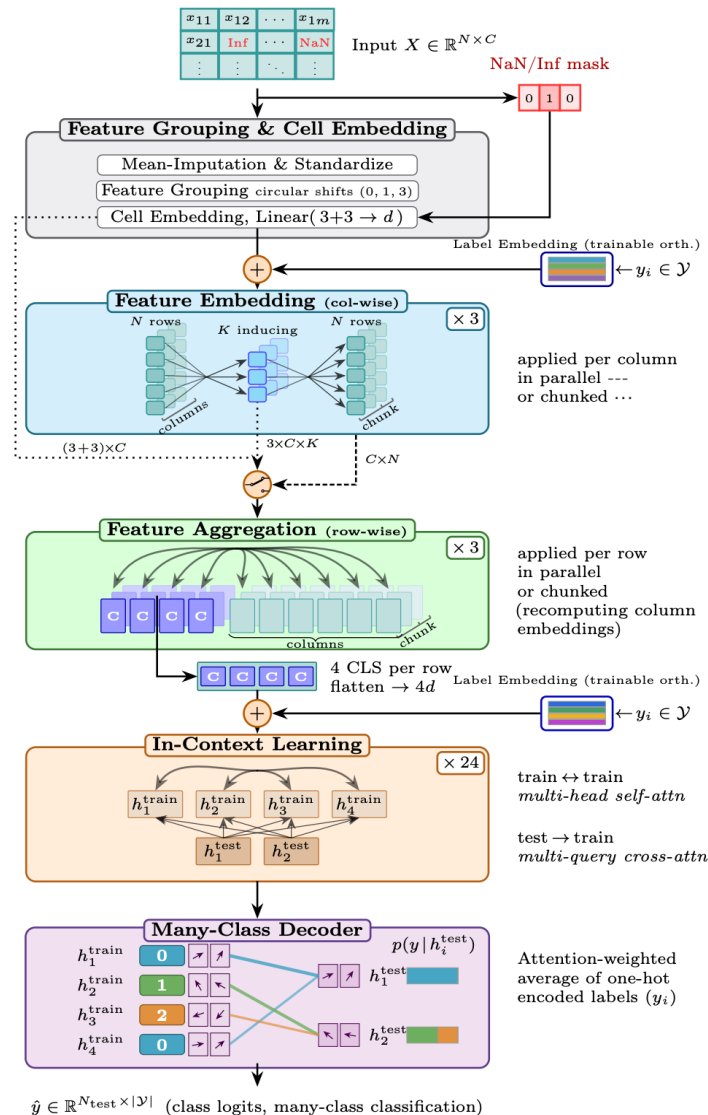
TabPFN can now be applied to arbitrary unseen real-world datasets



TabPFNv2 is trained using the cross-entropy loss between the model's predictions and the targets of held-out samples from synthetic datasets. **Note that the TabPFN series handles regression as a classification problem:** the target range is divided into thousands of bins, and the model predicts the corresponding bin.

TabPFN-2.5 introduces a few minor updates compared to TabPFNV2, including:

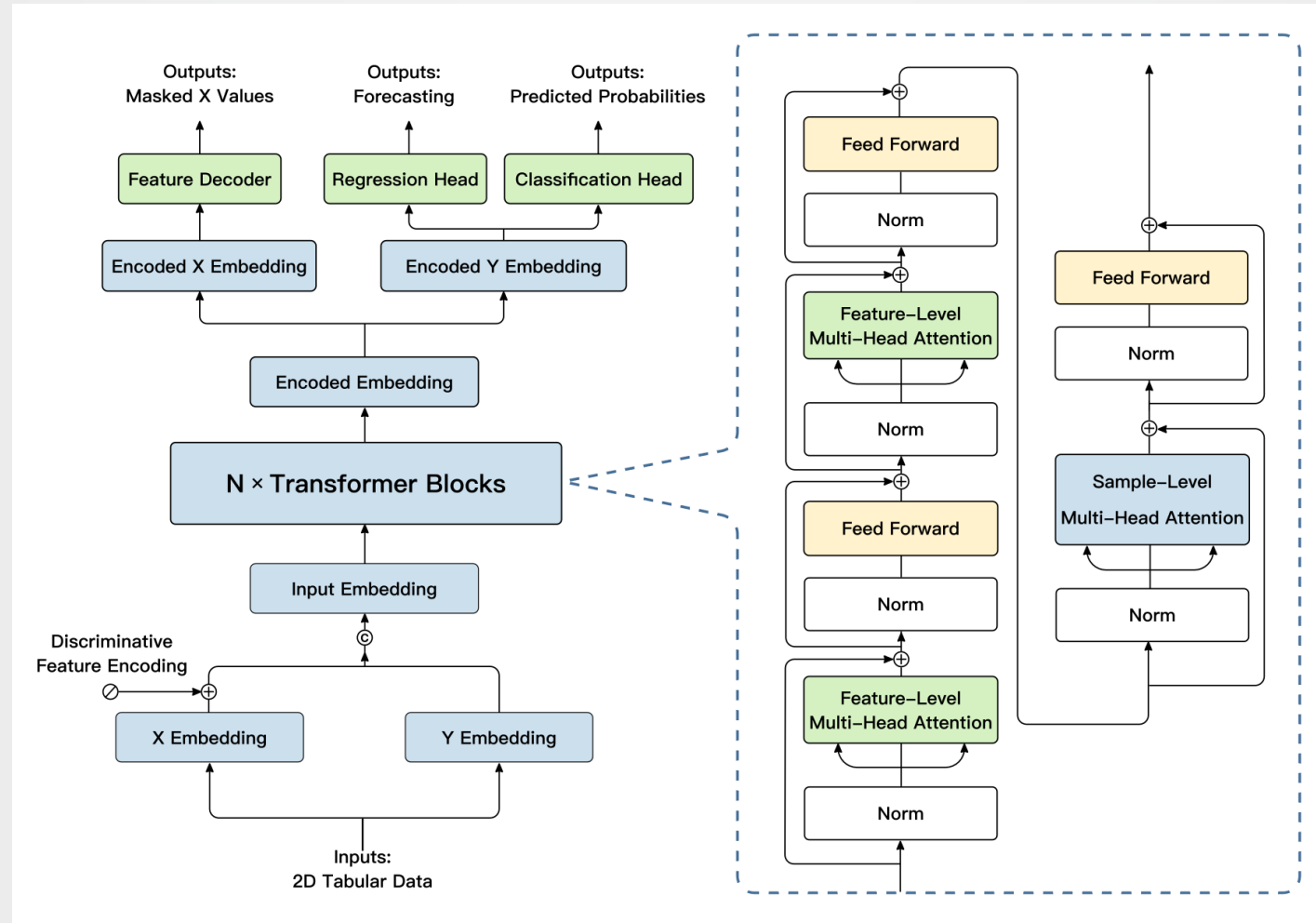
- TabPFN-2.5 increases the network depth from **12** to **18** layers for the regression model and **24** layers for the classification model;
- The authors use a group size of **3** for TabPFN-2.5, compared to **2** for TabPFNV2;
- For the regression models, TabPFN-2.5 replace the **linear encoder** used in TabPFNV2 by a **2-layer MLP**.
- TabPFN-2.5 adds **64 additional “thinking” rows** to the input dataset of TabPFN-2.5, which are learned during pretraining; these rows give additional computational capacity to the model and can also act as attention sinks to help the model ignore other rows.



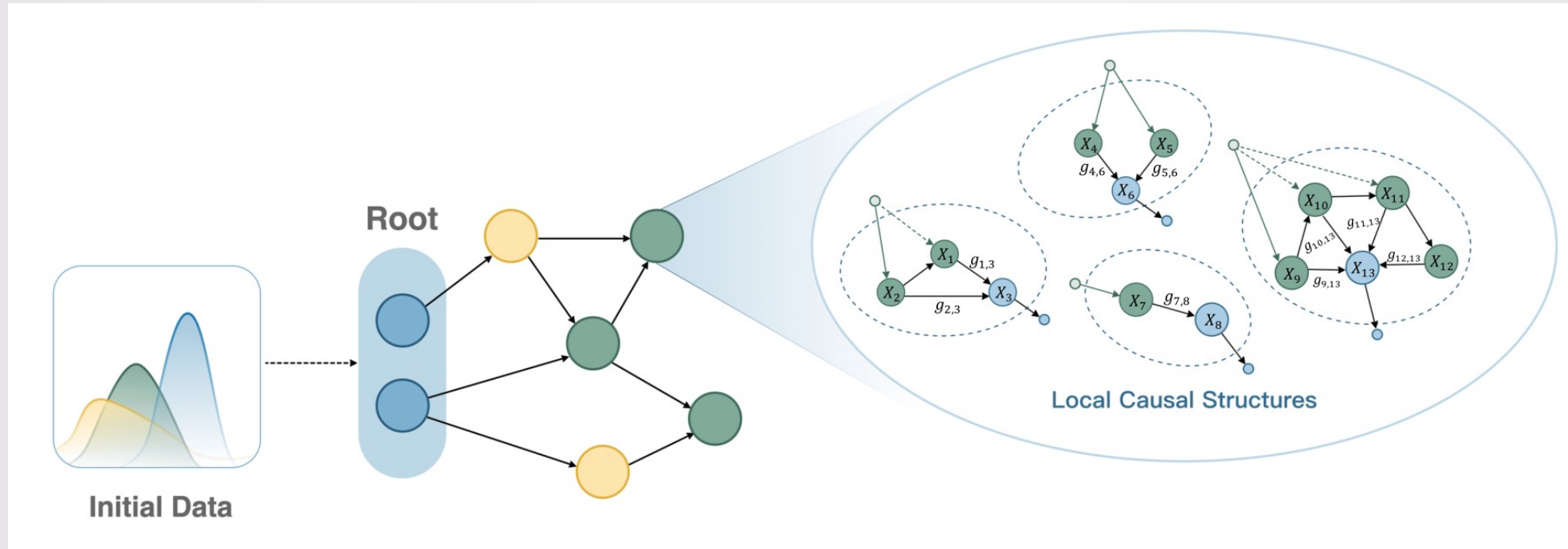
TabPFN-3 introduces a major architectural update and adopts some design elements from TabCLv2:

- It builds on the two-stage row-compression design introduced by TabCL, which uses a column-wise feature embedding layer followed by row-wise feature aggregation to obtain the row representation;
- For feature grouping, it borrows the triplet-based strategy from TabCLv2, forming feature triplets by pairing each feature with its two cyclically shifted neighbors;
- Additional modifications include changes to the decoder design, the replacement of LayerNorm with RMSNorm, and several other refinements.

LimiX: Architecture



Zhang, Xingxuan, et al. "Limix: Unleashing structured-data modeling capability for generalist intelligence." *arXiv preprint arXiv:2509.03505* (2025).



For the DAGs generation, LimiX follows [a hierarchical approach](#): the authors generate diverse local causal structures (LCSs), each encoding distinct causal dependencies. These LCSs are then interconnected following a prescribed procedure to form a complete DAG.

Regarding the edge functions, LimiX used three different types, including:

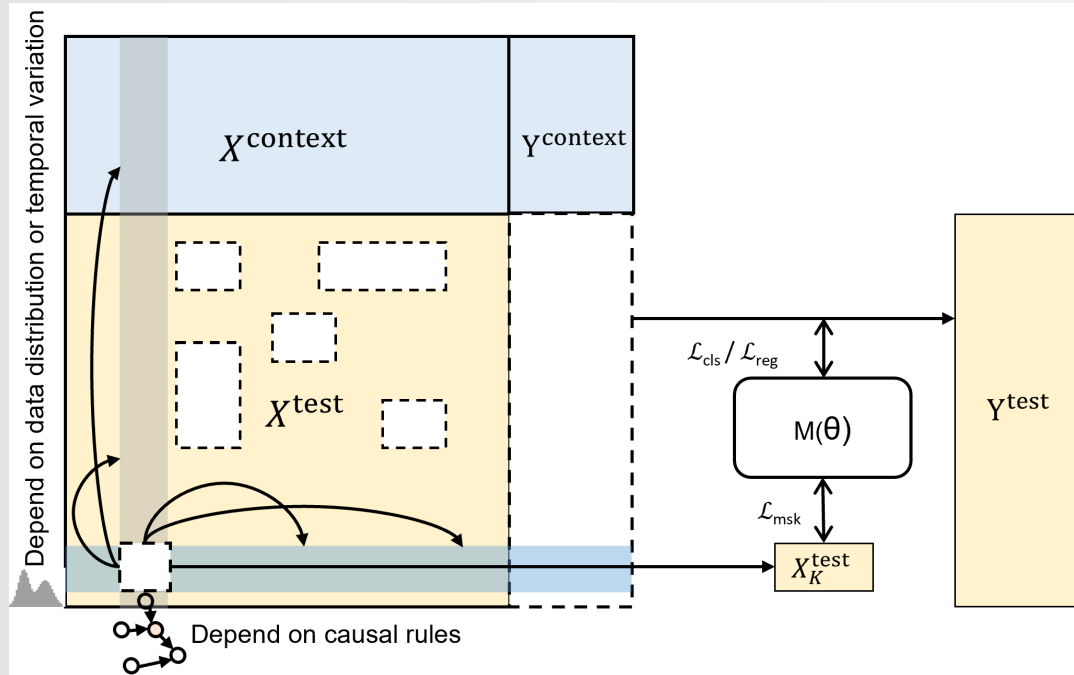
- **MLP**: it is used to introduces complex nonlinear dependencies along the edge;
- **Convolutional layers**: the convolution operation is applied along the sample dimension, serving as a local information mixer;
- **Decision Trees**: these are employed to introduce rule-based mappings and can take the form of either classification or regression trees.

The activation functions are uniformly sampled from **identity mapping**, **hyperbolic tangent**, **sigmoid**, **logarithm**, **absolute value**, **sine**, **squaring**, **modulo operation**, **heaviside step function**, and **GELU**.

The data sampling approach of LimiX consists of two strategies:

- **Graph-Aware Sampling:** perform training data sampling based on the graphical distribution of the generated data.
- **Solvability-Aware Sampling:** provide training data with varying degrees of solvability, thereby encouraging the model to achieve better generalization; the sampled problems can be divided into three classes: high-solvability, moderate-solvability and low-solvability.

LimiX: Pretraining



LimiX is pretrained by randomly masking cells in each row and learning to recover them from the observed context.

Exposure to diverse masking patterns enables the model to capture rich conditional relationships, which together form a unified joint model of the data.

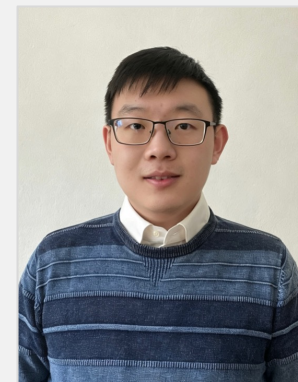
This joint model supports multiple tasks through a single mechanism: selecting a target column enables regression or classification, predicting masked entries allows imputation, and iteratively masking and refilling features enables data generation.

Zhang, Xingxuan, et al. "Limix: Unleashing structured-data modeling capability for generalist intelligence." *arXiv preprint arXiv:2509.03505* (2025).

Q&A



Presenter:
Dr. Xingxuan Zhang
Assistant Researcher
Department of Computer Science
Tsinghua University
Email: xingxuanzhang@hotmail.com

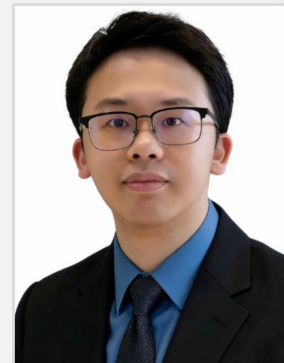


Presenter:
Dr. Shuyang Li
Research Scientist
Stable AI Technology Co., Ltd.
Email: lsy5215212@outlook.com

Cross-Modality Modeling



Presenter:
Prof. Han-Jia Ye
Associate Professor
School of Artificial Intelligence
Nanjing University
Email: yehj@nju.edu.cn



Presenter:
Prof. Jintai Chen
Assistant Professor
AI Thrust, Information Hub
The Hong Kong University of Science
and Technology (Guangzhou)
Email: jintaiCHEN@hkust-gz.edu.cn

Foundation Model for Tabular Prediction

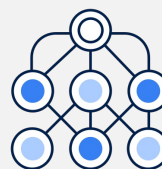
Tabular foundation model (TFM) predicts the label of a test instance based on a given downstream training set in a tuning-free manner.

Mileage	Year	Brand	Condition	Price
45000	2018	BMW	Excellent	18500
82000	2015	Ford	Good	12000
120000	2012	Honda	Fair	7500
25000	2020	BMW	Excellent	?

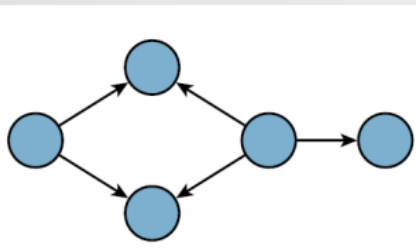
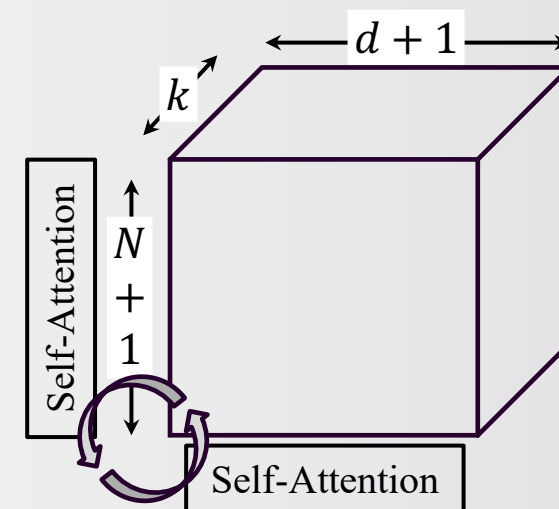
Price=
32,000

Income	Credit	Status
25	Low	Denied
51	High	Approved
38	Medium	?

Status=
Approved



TFMs usually utilize row-wise attention or row-wise and column-wise attention to mix information among cells.



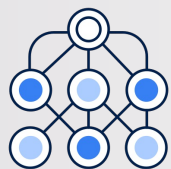
Pre-train on datasets generated by SCMs.

[Hollmann et al., ICLR'23/Nature'25]

[Qu et al., ICML'25/26] [Zhang et al., CoRR'25]

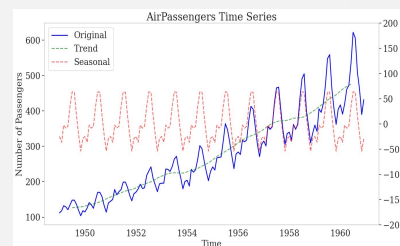
Tabular Foundation Model for Other Modalities

We can transform prediction tasks in other modalities as a tabular task. Then we can apply the pre-trained tabular foundation model to solve the problem, or train a foundation model *in a similar manner*.

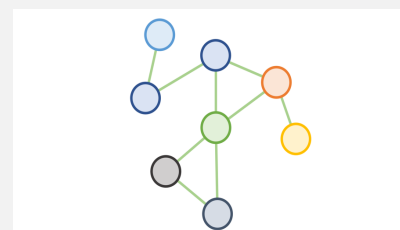


Features & Label Samples	F1	F2	F3	F4	Label
Sample_1	$v_{1,1}$	$v_{1,2}$	$v_{1,3}$	$v_{1,4}$	L_1
Sample_2	$v_{2,1}$	$v_{2,2}$	$v_{2,3}$	$v_{2,4}$	L_2
Sample_3	$v_{3,1}$	$v_{3,2}$	$v_{3,3}$	$v_{3,4}$	L_3
.	.	.	.	.	.
.	.	.	.	.	.
.	.	.	.	.	.
Sample_n	$v_{n,1}$	$v_{n,2}$	$v_{n,3}$	$v_{n,4}$	L_n

Table

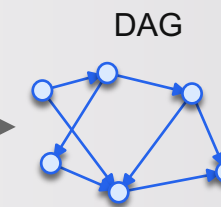
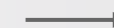


Time Series



Graph

sample	X1	X2	X3	...
obs	1.2	-0.3	2.1	...
int	1.0	1.5	-1.0	...
int	0.8	1.4	-2.0	...
obs	...	...	...	...



Causal Discovery

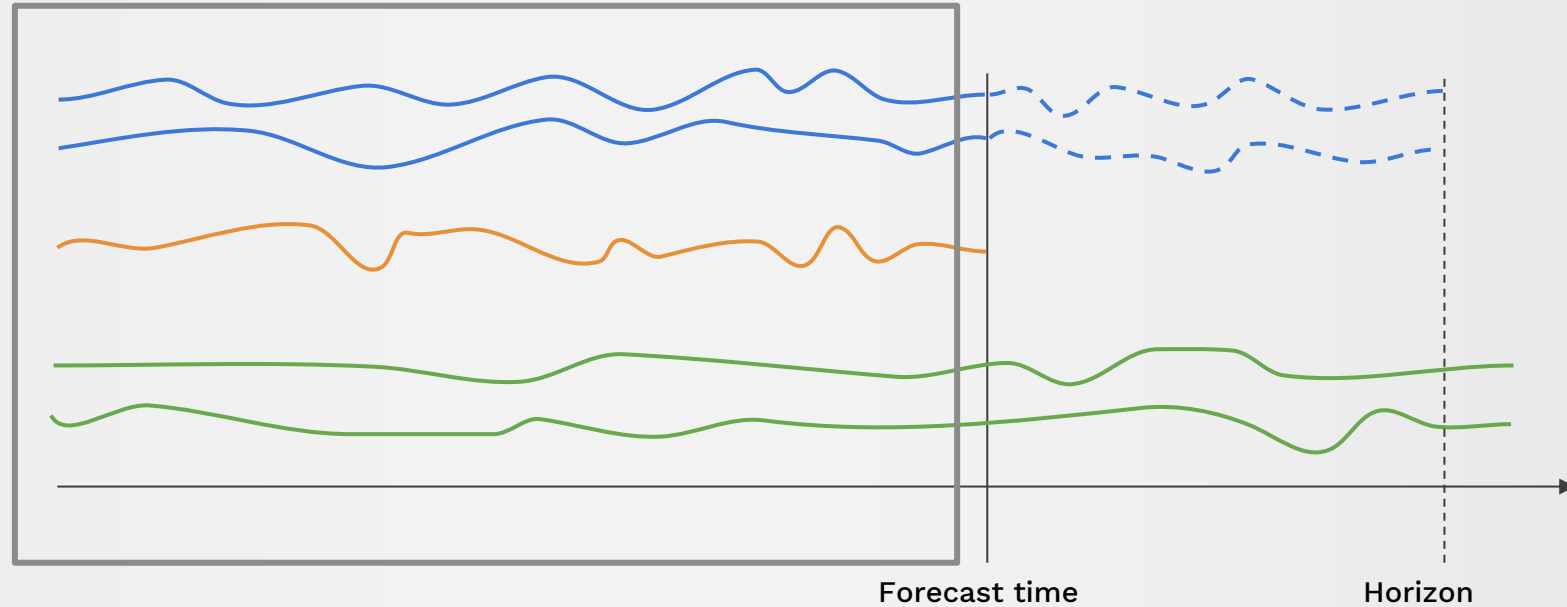
BBO, RL, Feature Engineering...

Time Series Forecasting

Target series: what we want to forecast

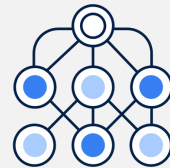
Past covariates:
Unknown into the future
e.g. measurements

Future covariates:
Known into the future
e.g. calendar, weather forecasts, actions taken



```
from nixtlats import TimeGPT

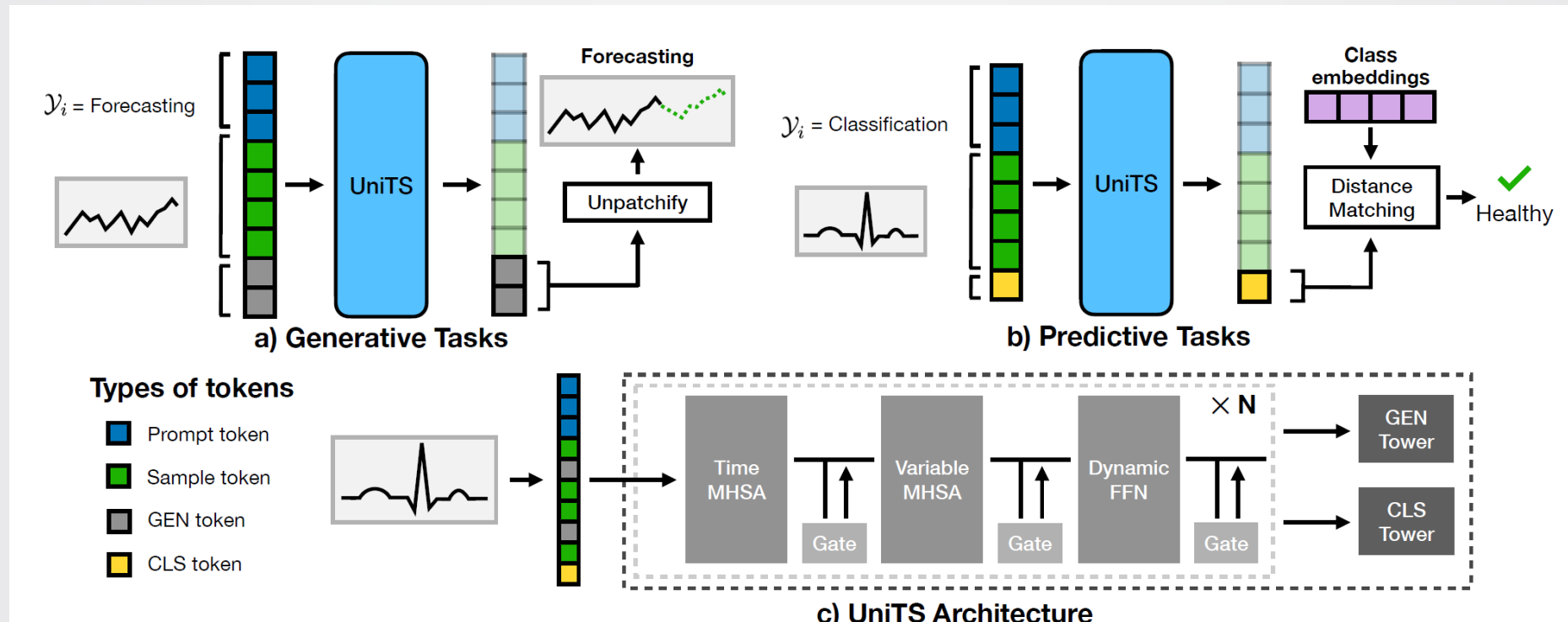
timegpt = TimeGPT(token='User_token')
future = timegpt.forecast(your_data)
```



Fine-Tune or Zero-Shot

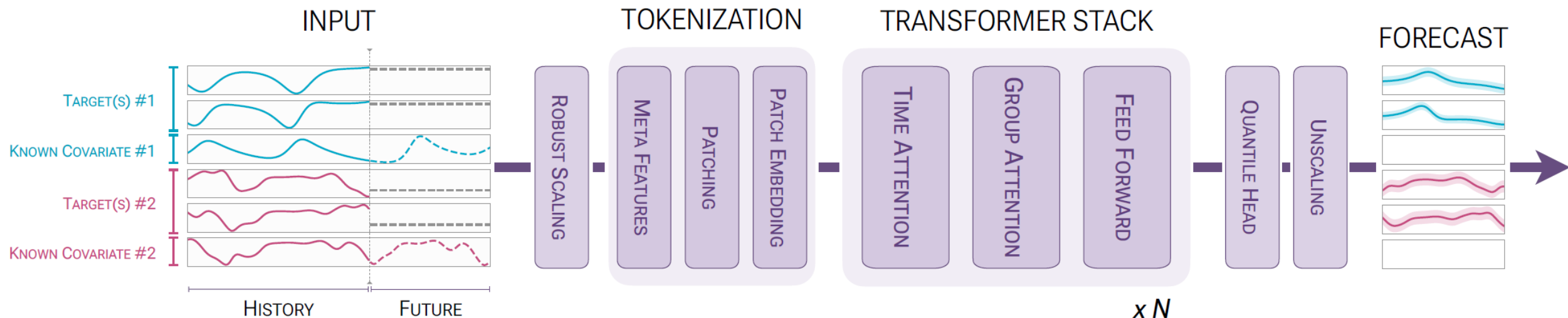
Time-Series Model with Similar Design

- A pre-trained model for time-series forecasting, imputation, classification, and analogy detection.
- Time $\times$ Variable dual axis Self-Attention + DyLinear (Dynamic Linear Operator)+ Gating Module
- The alternative attention in TabPFN v2 is very similar to the design in this paper.



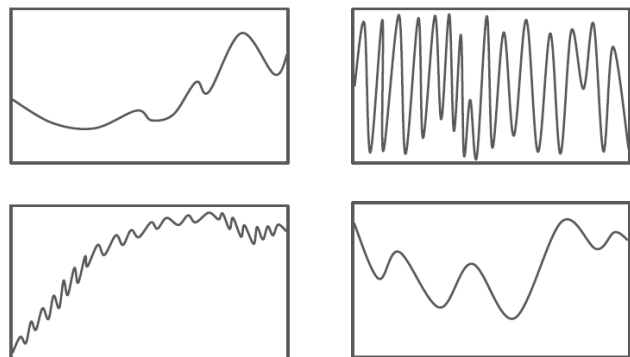
Time-Series Foundation Model with Similar Design

- A pre-trained foundation model to deal with univariate / multivariate / covariate-informed time series forecasting.
- Two types of attentions with alternative blocks (similar to TabPFN v2)
 - Time Attention: along the time axis, which aggregate the information across different patches in a series. (the position encoding is set as RoPE)
 - Group Attention: based on the same patch index, aggregate the information across the same group series/variates/covariates

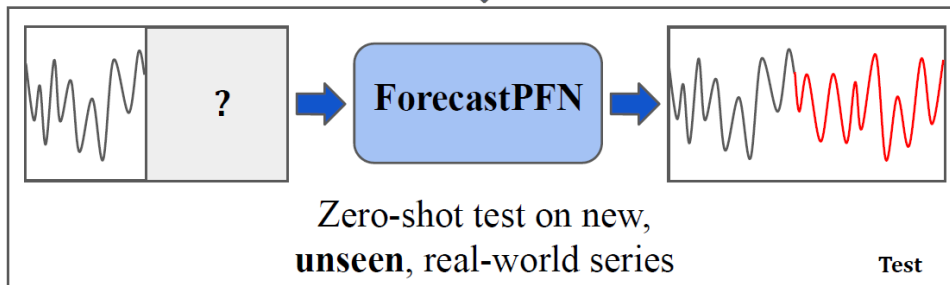


ForecastPFN setting

Train once, offline
with **synthetic**, diverse series



Train



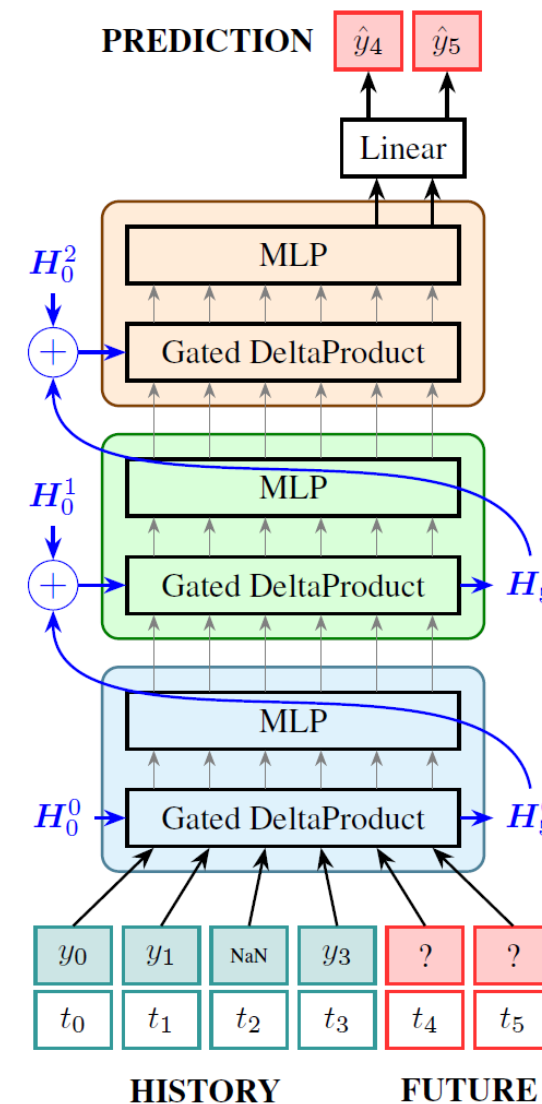
- token = continuous pair (t, y)
- The time index t is indexed as numerical features, such as year / month / day / week / day-of-year
- The value is encoded with robust scaling

- The prior design

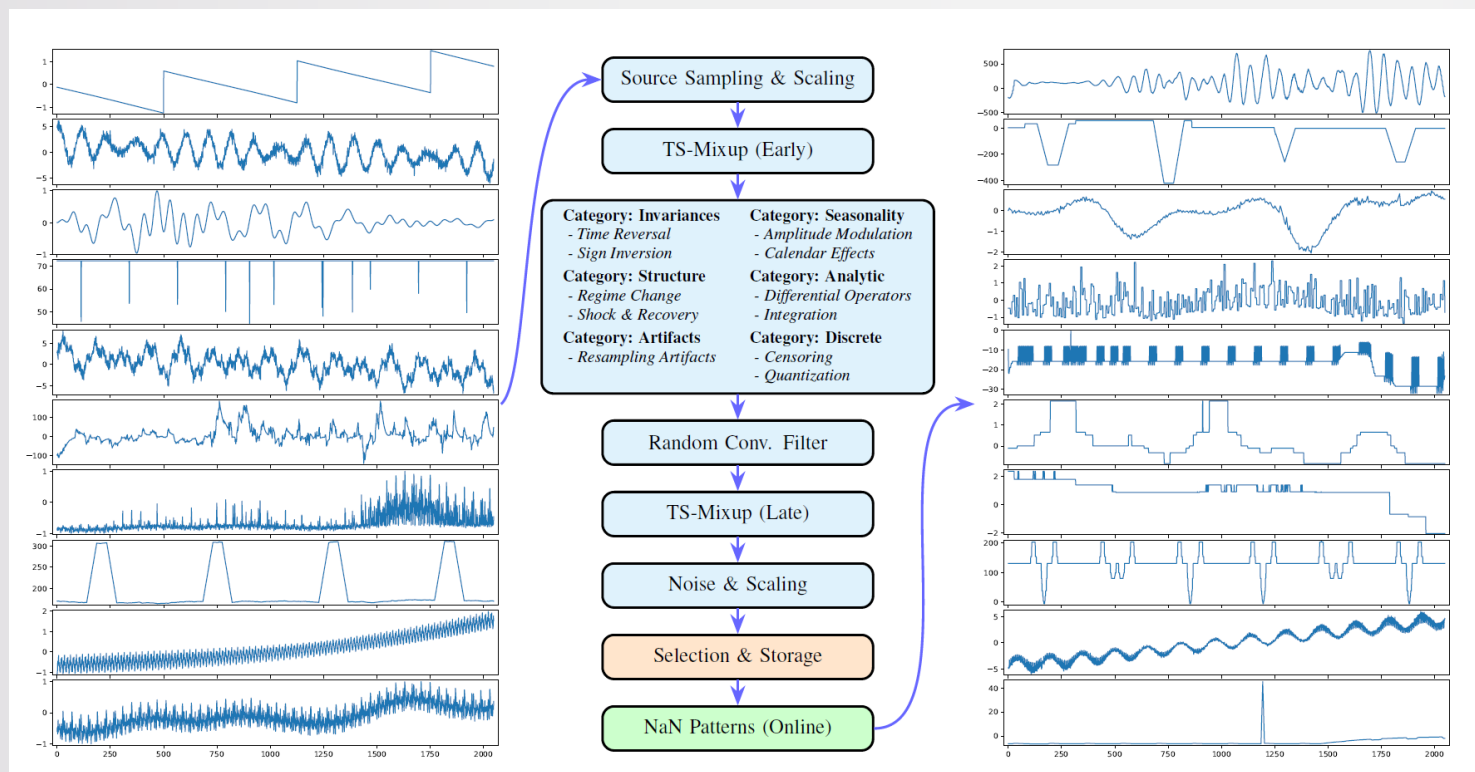
$$\begin{aligned}
 y_t &= \psi(t) \cdot z_t = \text{trend}(t) \cdot \text{seasonal}(t) \cdot z_t, \text{ where} \\
 z_t &= 1 + m_{\text{noise}} (z - \bar{z}), \text{ where } z \sim \text{Weibull}(1, k), \bar{z} = (\ln 2)^{1/k} \\
 \text{trend}(t) &= (1 + m_{\text{lin}} \cdot t + c_{\text{lin}})(m_{\text{exp}} \cdot c_{\text{exp}}^t) \\
 \text{seasonal}(t) &= \text{seasonal}_{\text{week}}(t) \cdot \text{seasonal}_{\text{month}}(t) \cdot \text{seasonal}_{\text{year}}(t), \text{ where} \\
 \text{seasonal}_v(t) &= 1 + m_v \sum_{f=1}^{\lfloor v/2 \rfloor} \left[c_{f,v} \sin\left(2\pi f \frac{t}{p_v}\right) + d_{f,v} \cos\left(2\pi f \frac{t}{p_v}\right) \right]
 \end{aligned}$$

- TempoPFN represents the observed history and future time stamps as a single token sequence.
- Historical tokens contain both time features and observed values, while future tokens contain only time features.
- A Linear RNN backbone based on GatedDeltaProduct recurrence processes the whole sequence efficiently, and a state-weaving mechanism allows information to flow across the full prediction horizon.
- The model predicts all future time steps in one forward pass and outputs quantile forecasts.

Vladyslav Moroshan et al., TempoPFN: Synthetic Pre-training of Linear RNNs for Zero-shot Time Series Forecasting. ICLR 2026

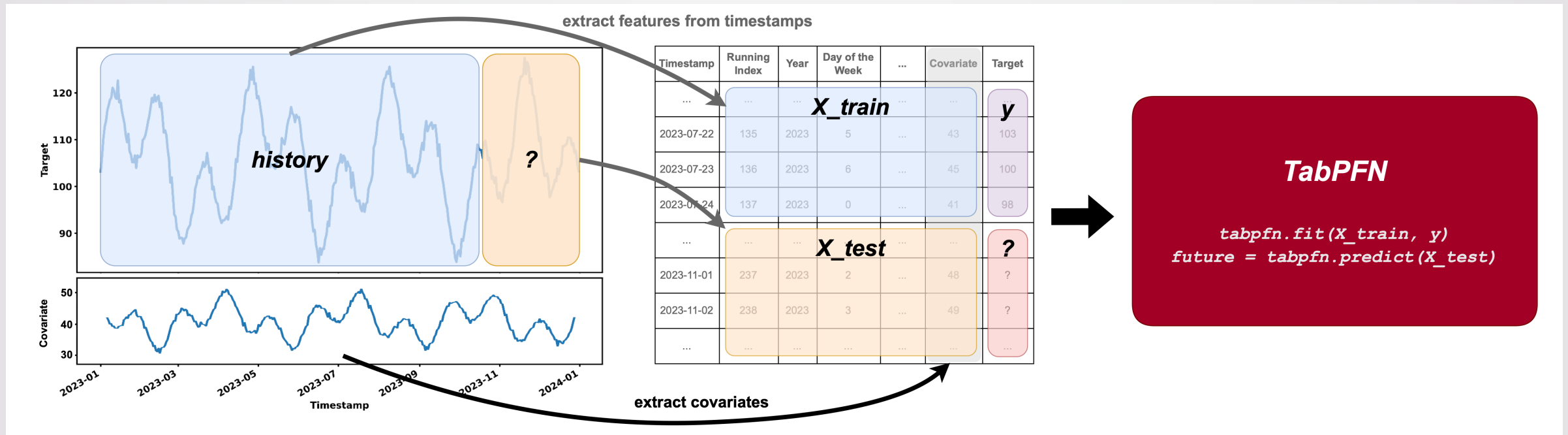


- The pre-training corpus is generated entirely from synthetic time-series priors, including Gaussian processes, kernel-based generators, causal-kernel generators, ForecastPFN-style trend/seasonality priors, sawtooth/step/spike/anomaly patterns, audio-inspired rhythms, and regime-switching SDEs.



These generators are combined with augmentations such as time reversal, sign inversion, regime changes, shock recovery, amplitude modulation, calendar effects, censoring, quantization, convolutional filtering, TS-mixup, scaling, noise injection, and missing-value patterns.

How Can Tabular Foundation Models Solve Time Series Forecasting



Hoo, Shi Bin, et al. "The tabular foundation model tabpfn outperforms specialized time series forecasting models based on simple features." NeurIPS workshop on time series in the age of large models. 2024.

How Can Tabular Foundation Models Solve Time Series Forecasting

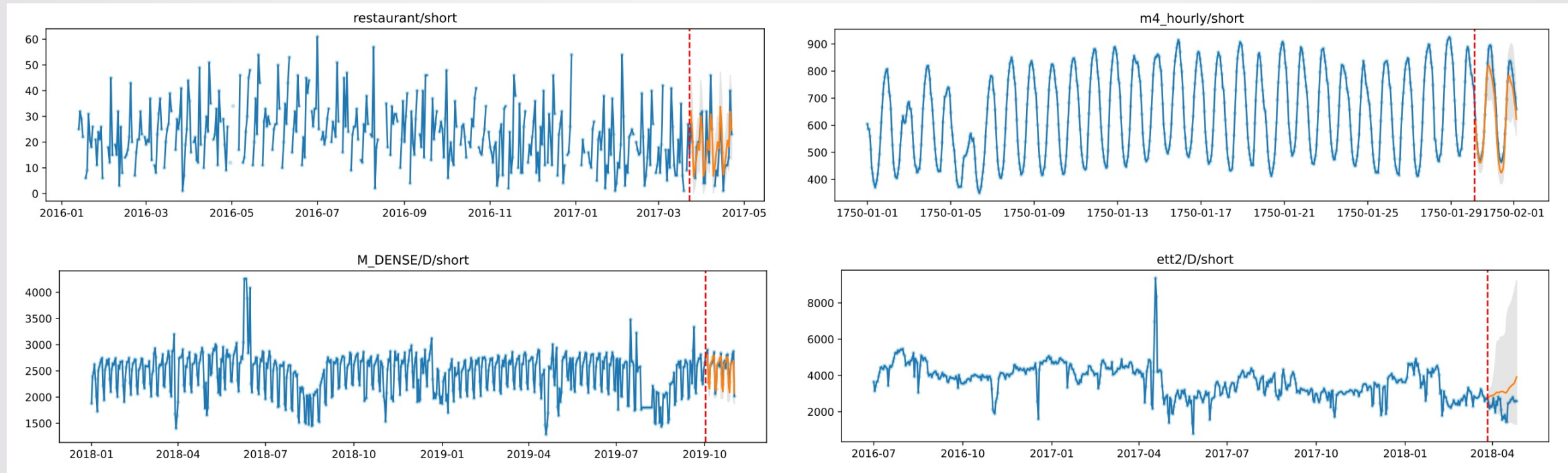


TabPFN-TS leverages standard time-series feature engineering techniques to generate features, including :

- **Running Index**: this feature includes the index of time steps;
- **Calendar Features**: these features are the sine and cosine encodings of periodic calendar components (e.g., second, minute, hour, etc.).
- **FFT-based Periods Modeling**: sine and cosine encodings of the top-k periodic components identified via FFT.

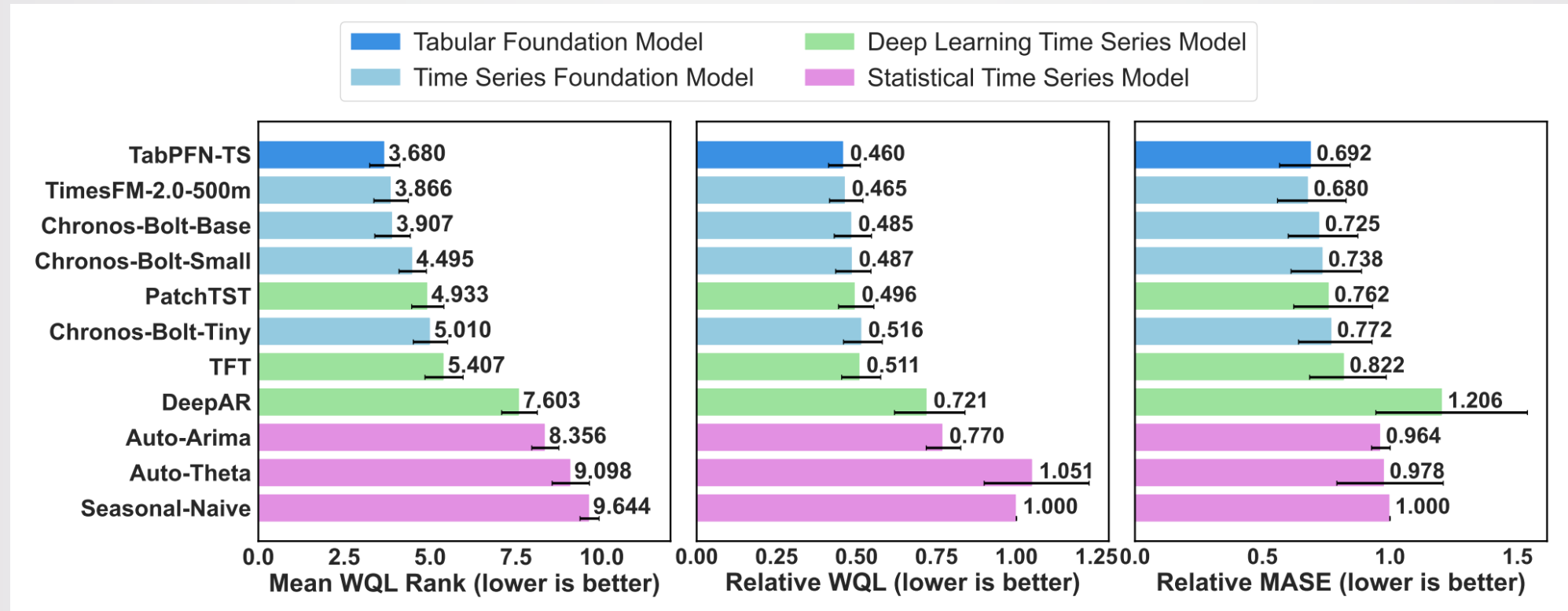
Hoo, Shi Bin, et al. "The tabular foundation model tabpfm outperforms specialized time series forecasting models based on simple features." NeurIPS workshop on time series in the age of large models. 2024.

How Can Tabular Foundation Models Solve Time Series Forecasting



Hoo, Shi Bin, et al. "The tabular foundation model tabpfm outperforms specialized time series forecasting models based on simple features." NeurIPS workshop on time series in the age of large models. 2024.

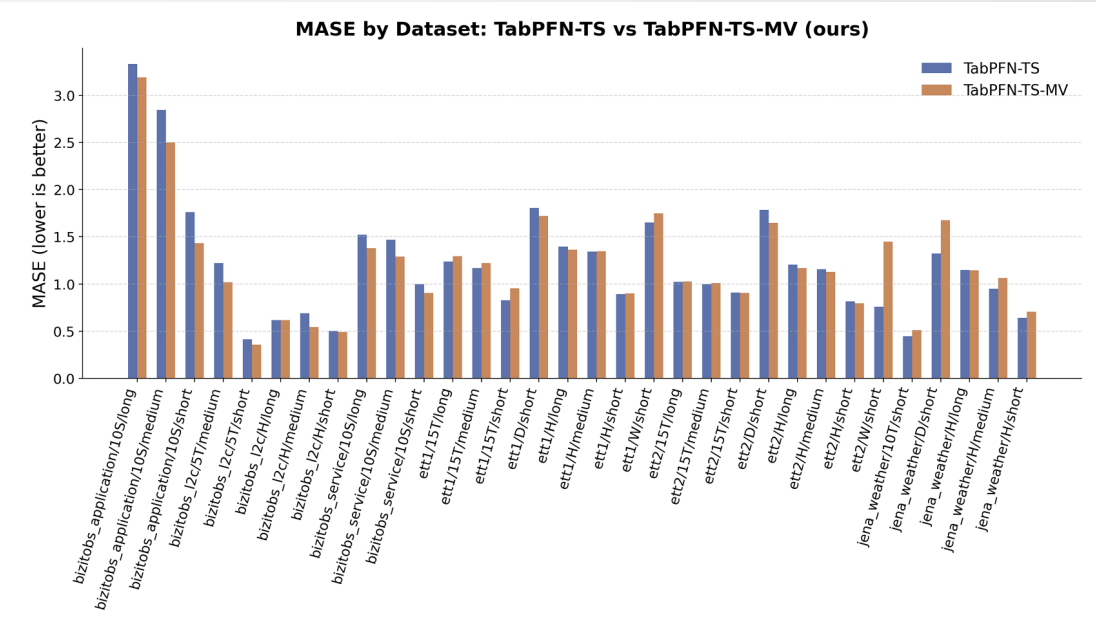
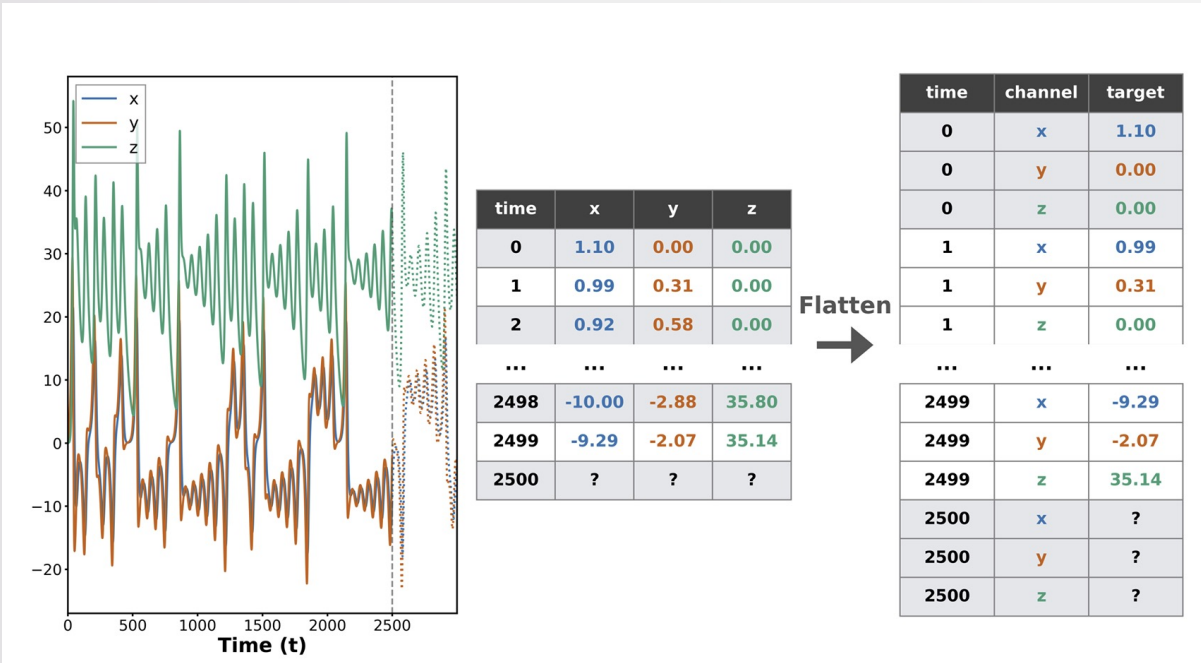
How Can Tabular Foundation Models Solve Time Series Forecasting



Hoo, Shi Bin, et al. "The tabular foundation model tabpfm outperforms specialized time series forecasting models based on simple features." NeurIPS workshop on time series in the age of large models. 2024.

Zero-shot Multivariate Time Series Forecasting Using TabPFN

It converts multivariate forecasting into tabular scalar regression by rolling out a multivariate series into rows indexed by time and channel, using features such as timestamp, channel indicator and observed value, with the next scalar value as the target.



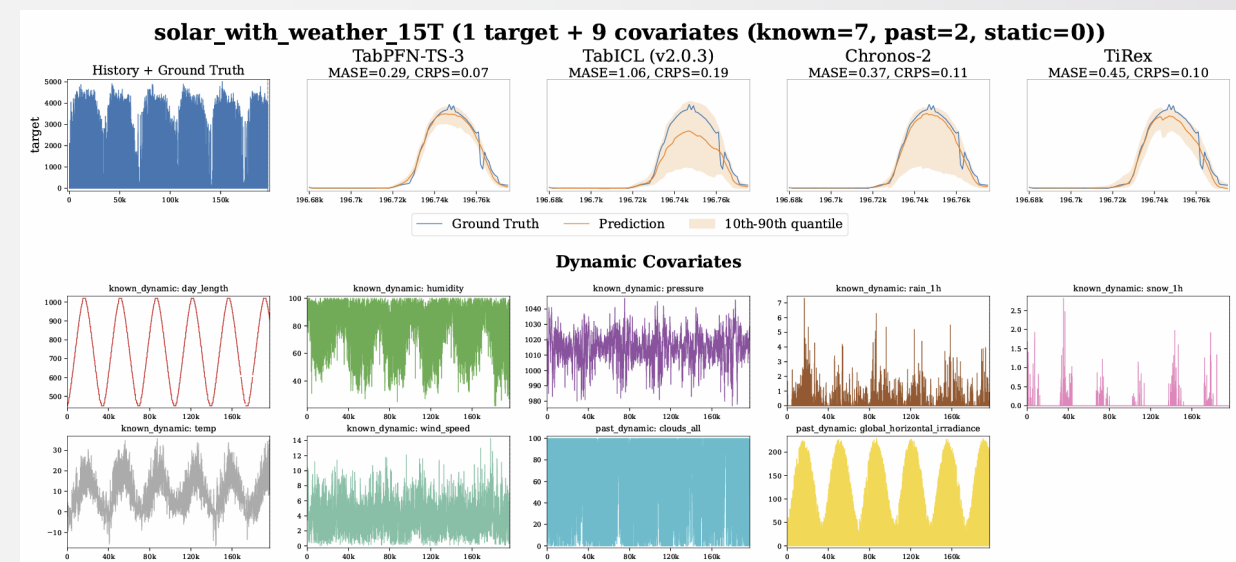
TabPFN-TS-MV improves MASE over standard TabPFN-TS on about 60% of multivariate GIFT-Eval datasets and slightly improves average MASE, although its probabilistic WQL/calibration remains weaker.

Time-Series Forecasting with TabPFN v3

TabPFN-TS-3 uses a specialized TabPFN-3 regression checkpoint fine-tuned on synthetic time-series data. The report states that this checkpoint ranks 2nd on fev-bench, suggesting that synthetic time-series fine-tuning and model scaling can further improve the TFM-based forecasting route.

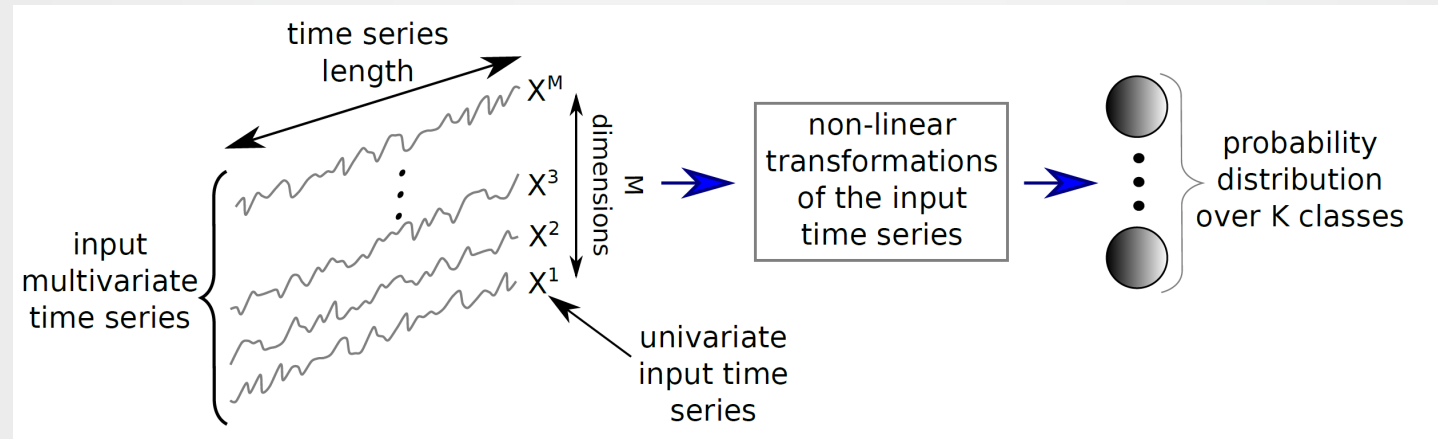
Model	Win (%)	Skill (%)	Runtime (s)	Leak. (%)	# fails
Chronos-2	86.9	35.5	0.8	0	0
TabPFN-TS-3	69.8	30.6	234.6	0	0
TimesFM-2.5	74.9	30.2	1.9	10	0
TiRex	76.9	30.0	0.2	1	0
Toto-1.0	66.3	28.2	22.1	8	0
TabPFN-v2-TS	58.5	27.6	88.9	0	2
Moirai-2.0	61.4	27.3	0.3	28	0
Chronos-Bolt	60.7	26.5	0.2	0	0
Sundial-Base	53.4	24.7	8.0	1	0
CatBoost (Recursive)	54.0	23.7	0.3	0	0
LightGBM (Recursive)	50.3	22.4	0.3	0	0
Stat. Ensemble	46.7	15.7	148.6	0	11
AutoARIMA	36.0	11.2	19.5	0	10
AutoTheta	34.2	11.0	3.3	0	0
TabICL-v2 [†]	33.2	7.0	64.7	0	0
AutoETS	33.5	2.3	3.5	0	3
Seasonal Naive	20.0	0.0	0.5	0	0
Naive	18.0	-16.7	0.5	0	0
Drift	15.3	-18.1	0.5	0	0

Full marginal forecasting MASE (point)
performance on fevbench (100tasks).

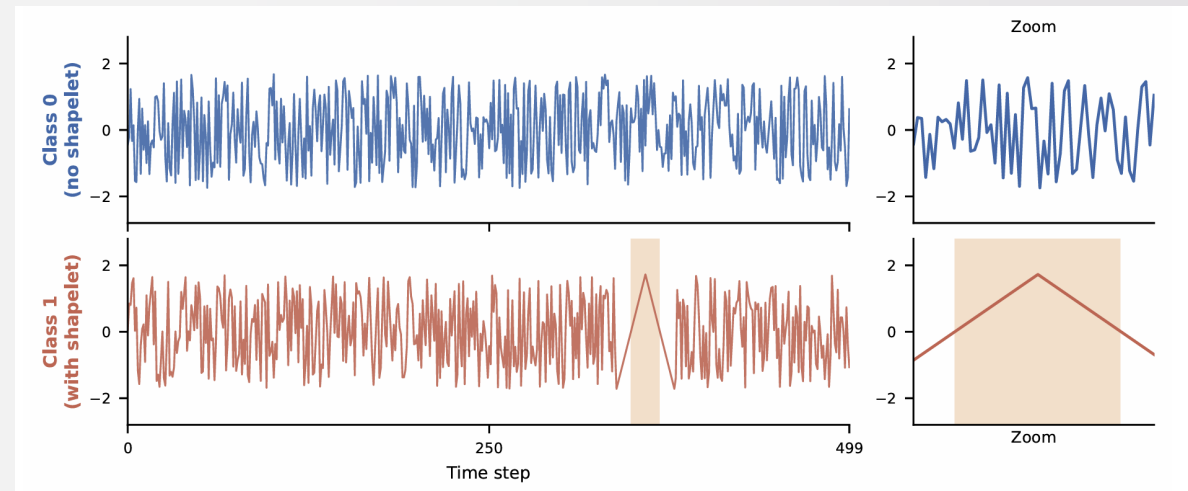


Qualitative forecast examples

RocketPFN: Accurate Time Series Classification via In-Context Learning



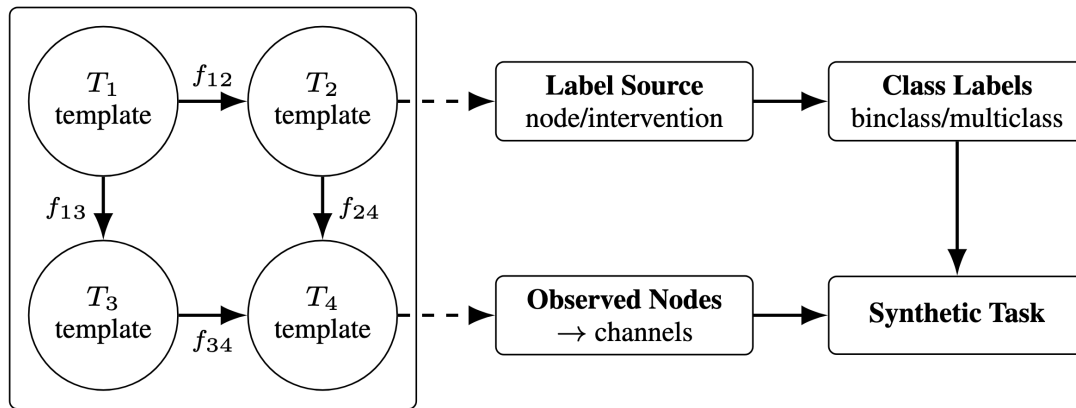
- It converts time-series classification into tabular classification by applying ROCKET random convolutional kernels to each series and feeding the resulting fixed-length feature vectors into TabPFN v2.5, with probability averaging across feature groups.
- It matches HC2 on 92 UCR datasets with mean accuracy 0.900 while learning no target-data parameters, and it also shows that simply flattening raw time series into TabPFN is not enough.



Synthetic Causal Priors for In-Context Time-Series Classification

- Frames time-series classification as support/query in-context classification episodes and studies what synthetic prior is needed to make PFN/TabPFN-style learning work for TSC.
- The prior combines KernelSynth-style temporal templates, SCM/DAG mechanisms and causal label rules, so the focus is less on handcrafted feature conversion and more on designing a synthetic task distribution that induces time-series classification ability.

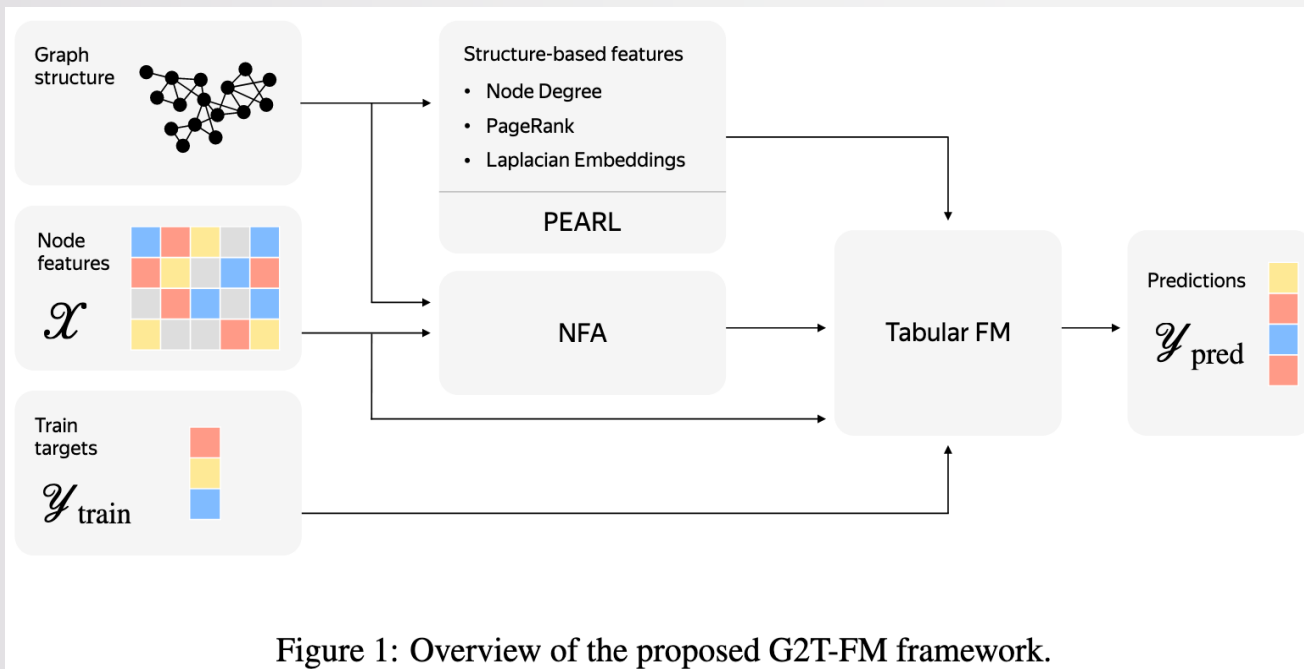
CauKer-style SCM over KernelSynth



Pretraining source	Inference	Acc.	ROC-AUC
<i>In-domain reference</i>			
Real UCR mixup reference	full-support	0.823	0.852
<i>Synthetic non-causal priors</i>			
TiCT-style synthetic	full-support	0.760	0.799
TiCT-style synthetic	retrieval-64	0.788	0.828
Mechanism-id prior	full-support	0.715	0.746
<i>Synthetic causal priors</i>			
SCM hidden-label with enforced path	full-support	0.776	0.815
SCM hidden-label with enforced path	retrieval-64	0.802	0.846
Unconstrained hidden-label prior	full-support	0.708	0.742

Results on the 42-task binary UCR subset used by TimEE under official split. Real-data mixup is an in-domain reference because it constructs pretraining tasks from target benchmark training distributions. Synthetic priors do not use target benchmark series during pretraining.

How Can Tabular Foundation Models Solve Graph Tasks



- **Neighborhood Aggregation Features:** Aggregate neighborhood information into node features to capture local relational patterns;
- **Handcrafted Structural Features:** Use explicit structural descriptors, such as centrality and spectral/positional features, to characterize each node's role in the graph;
- **Learned Structure Encodings:** Leverage structure encoders to extract higher-order topological patterns beyond manually designed structural descriptors.

How Can Tabular Foundation Models Solve Graph Tasks

Table 2: Evaluation results on datasets with tabular features (datasets from the GraphLand benchmark under the RL (random low) data split). For each column, we highlight **first**, **second**, and **third** best results with a color.

	tolokers-2	city-reviews	artnet-exp	hm-prices	avazu-ctr	city-roads-M	twitch-views	artnet-views	AR
LightGBM+NFA	56.34 ± 0.06	78.53 ± 0.01	46.13 ± 0.03	70.84 ± 0.04	31.71 ± 0.01	61.18 ± 0.03	60.14 ± 0.01	56.10 ± 0.02	5.38
GCN	56.27 ± 0.29	77.81 ± 0.14	44.86 ± 0.34	68.02 ± 0.40	32.00 ± 0.15	58.82 ± 0.24	75.51 ± 0.05	56.03 ± 0.24	6.25
GraphSAGE	54.43 ± 0.32	78.17 ± 0.09	45.14 ± 0.34	70.00 ± 0.70	31.44 ± 0.15	59.44 ± 0.26	66.29 ± 0.31	49.32 ± 0.86	7.12
GAT	57.41 ± 0.80	77.74 ± 0.20	45.06 ± 0.49	72.07 ± 1.16	32.63 ± 0.16	59.86 ± 0.19	72.89 ± 0.25	53.60 ± 0.23	5.12
GT	56.98 ± 0.53	77.34 ± 0.20	46.41 ± 0.68	69.44 ± 0.89	31.11 ± 0.47	59.55 ± 0.27	72.13 ± 0.13	53.37 ± 0.43	6.62
OpenGraph (ICL)	40.38 ± 1.13	59.09 ± 0.72	15.16 ± 0.83	N/A	N/A	N/A	N/A	N/A	11.00
AnyGraph (ICL)	28.75 ± 3.56	63.71 ± 1.45	12.84 ± 0.93	N/A	N/A	N/A	N/A	N/A	12.33
TS-GNN (ICL)	38.54 ± 0.94	43.46 ± 5.17	20.44 ± 1.05	N/A	N/A	N/A	N/A	N/A	11.33
GCOPE (FT)	28.81 ± 1.28	67.16 ± 0.98	14.92 ± 1.56	N/A	N/A	N/A	N/A	N/A	11.33
G2T-TabPFNv2 (ICL)	60.42 ± 0.27	77.46 ± 0.10	45.84 ± 0.03	66.68 ± 0.09	26.38 ± 0.07	60.47 ± 0.04	70.00 ± 0.06	58.75 ± 0.15	6.38
G2T-TabPFNv2 (FT)	57.65 ± 1.92	79.12 ± 0.21	47.31 ± 0.59	71.05 ± 0.91	28.52 ± 0.43	63.08 ± 0.28	74.06 ± 0.16	60.29 ± 0.13	3.62
G2T-LimiX (ICL)	61.48 ± 0.30	77.72 ± 0.54	48.43 ± 0.18	74.96 ± 0.06	32.39 ± 0.14	64.53 ± 0.07	71.08 ± 0.07	60.95 ± 0.10	3.12
G2T-LimiX (FT)	61.17 ± 0.49	80.13 ± 0.05	49.88 ± 0.13	76.32 ± 0.17	33.94 ± 0.34	65.87 ± 0.10	73.16 ± 0.40	62.12 ± 0.10	1.38

Strengths

- Enables zero-shot or training-free adaptation on unseen graphs.
- Can be empirically competitive with specialized GNNs.

Limitations

- Graph structure is only modeled indirectly through engineered features.
- Performance depends heavily on the handcrafted structural encodings.

How Can Tabular Foundation Models Solve Graph Tasks

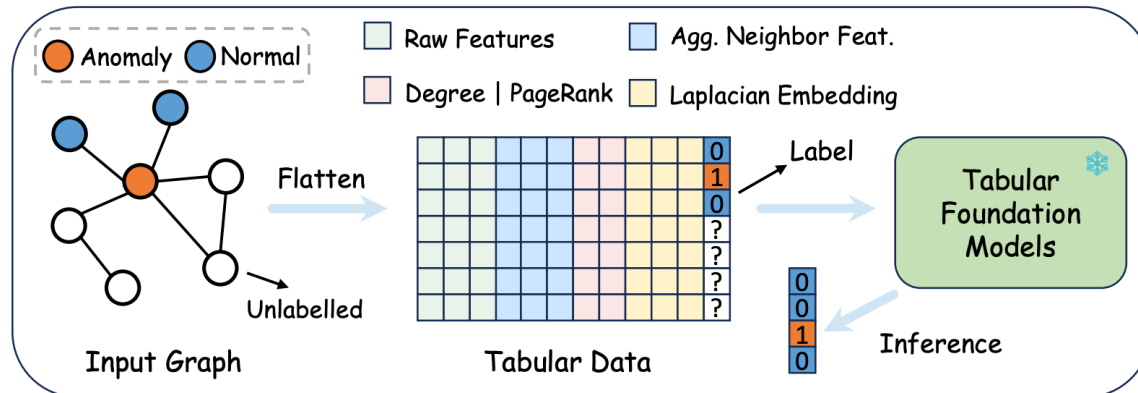


Table 2: Performance comparison. Baseline results are from GADBench except ConsisGAD and SpaceGNN.

	Method	Amazon		YelpChi		T-Finance		T-Social		Average	
		AUROC	AUPRC	AUROC	AUPRC	AUROC	AUPRC	AUROC	AUPRC	AUROC	AUPRC
Traditional	GCN	82.0±0.3	32.8±1.2	51.2±3.7	16.4±2.6	88.3±2.5	60.5±10.8	71.6±10.4	8.4±3.8	73.28	29.52
	AMNet	92.8±2.1	82.4±2.2	64.8±5.2	23.9±3.5	92.6±0.9	60.2±8.2	53.7±3.4	3.1±0.3	75.97	42.40
	BWGNN	91.8±2.3	81.7±2.2	64.3±3.4	23.7±2.9	92.1±2.7	60.9±13.8	77.5±4.3	15.9±6.2	81.42	45.55
	GHRN	90.9±1.9	80.7±1.7	64.5±3.1	23.8±2.8	92.6±0.7	63.4±10.4	78.7±3.0	16.2±4.6	81.68	46.02
	RFGGraph	92.5±1.3	70.7±5.1	61.6±2.7	23.6±2.5	95.0±0.7	81.1±2.7	88.6±1.6	51.3±6.2	84.42	56.68
	XGBGraph	94.7±0.9	84.4±1.1	64.0±3.5	24.8±3.1	94.8±0.6	78.3±3.1	85.2±1.8	40.6±7.6	84.68	57.02
	ConsisGAD	92.31±2.16	78.68±5.65	66.08±3.80	25.85±2.93	94.31±0.76	79.68±4.73	93.07±1.90	41.29±4.97	86.44	56.38
	SpaceGNN	91.08±2.50	81.14±2.29	66.81±2.75	25.73±2.35	93.41±0.96	81.03±3.46	94.72±0.65	58.99±5.69	86.50	61.72
TFM4GAD	Limix-2M	95.65±1.25	83.67±5.19	70.83±4.36	31.96±4.88	95.69±0.40	83.49±2.66	96.44±1.01	83.26±4.33	89.65	70.60
	Limix-16M	96.14±1.02	85.50±4.24	70.54±4.93	31.57±5.16	96.01±0.40	84.92±1.63	96.98±0.63	87.01±2.18	89.92	72.25
	TabPFNV2	95.94±1.03	86.76±1.91	70.91±4.92	30.72±4.89	95.38±0.66	82.75±2.75	96.93±0.60	86.14±3.08	89.79	71.59
	TabPFNV2.5	96.12±0.99	87.28±1.61	71.64±3.62	31.69±4.13	94.52±1.04	79.55±3.37	97.16±0.56	87.19±2.48	89.86	71.43

Best Graph-Based Method

Best Method

Tabular Foundation Models do not “create causality” by themselves. They solve causal tasks by becoming highly accurate base learners inside causal estimators.

- **Causal setup**
 - Observed table: features X , treatment T , outcome Y
 - Goal: estimate individual treatment effect
 - CATE: $\tau(x) = E[Y(1) - Y(0) \mid X = x]$
- **How TabPFN is used**
 - Estimate outcome models: $\mu_1(x), \mu_0(x)$
 - Estimate propensity: $e(x) = P(T = 1 \mid X = x)$
 - Plug into causal meta-learners: T-Learner, S-Learner, X-Learner
 - Output: CATE / uplift / policy ranking

Evidence from TabPFN-2.5

TabPFN-2.5: strong PEHE on RealCause

RealCause uses real RCT data and adds synthetic observable confounding. TabPFN-2.5 as a **T-Learner** achieves the strongest overall performance.

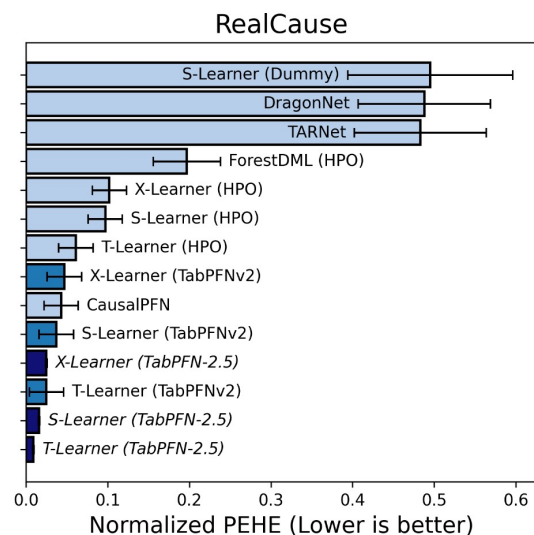


Figure 9: PFN-based CATE estimators dominate RealCause, outperforming specialized tree- and deep-learning-based methods for causal inference. Choice of propensity and outcome model is important for CATE estimation.

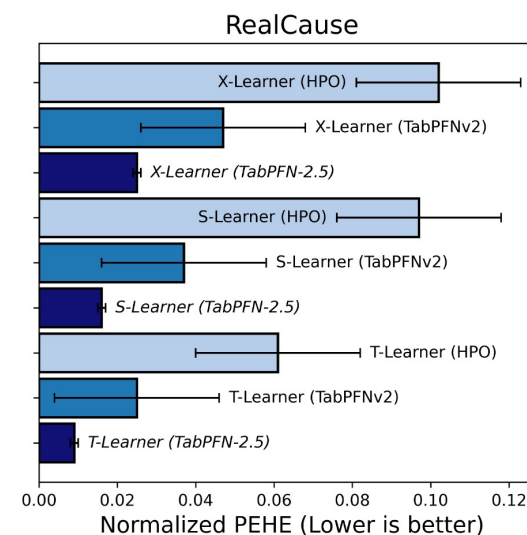
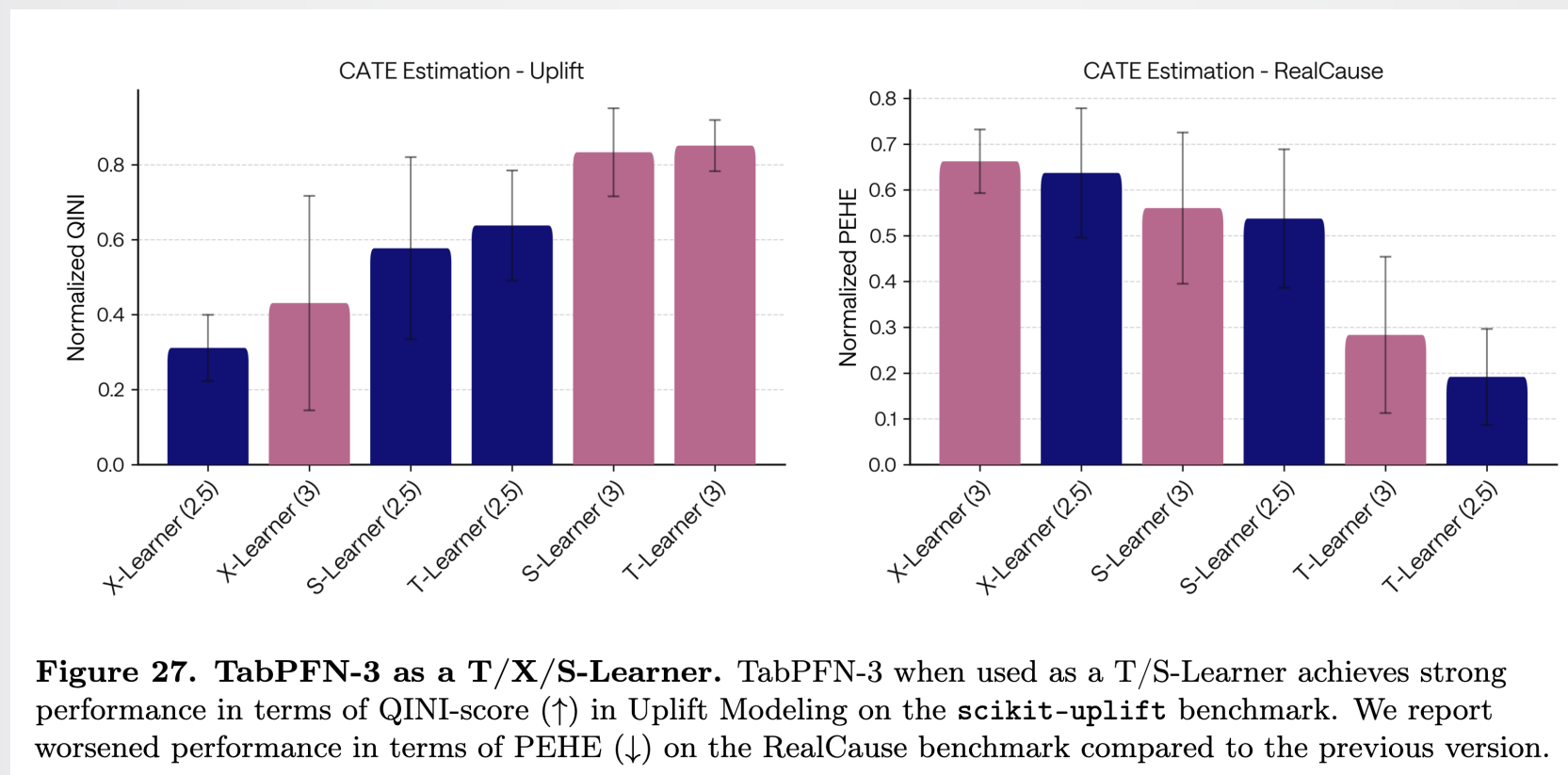


Figure 10: Improvements in base model predictive performance transfer to improved performance in CATE estimation. Our new model, TabPFN-2.5, is the strongest choice of base model for all meta-learners.

TabPFN-3: stronger uplift ranking, mixed RealCause result

TabPFN-3 improves over TabPFN-2.5 on the scikit-uplift benchmark using QINI-score, with T- and S-Learners taking the top positions.

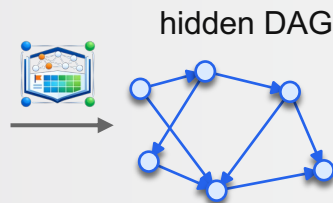


TabCausal: A TFM for Causal Discovery

Causal discovery foundation models (CDFMs) map a dataset directly to a causal graph in a single forward pass, avoiding per-dataset testing, search, or optimization.

CDFM: table $\rightarrow$ causal graph

sample	X1	X2	X3	...
obs	1.2	-0.3	2.1	...
int	1.0	1.5	-1.0	...
int	0.8	1.4	-2.0	...
obs	...	...	...	...

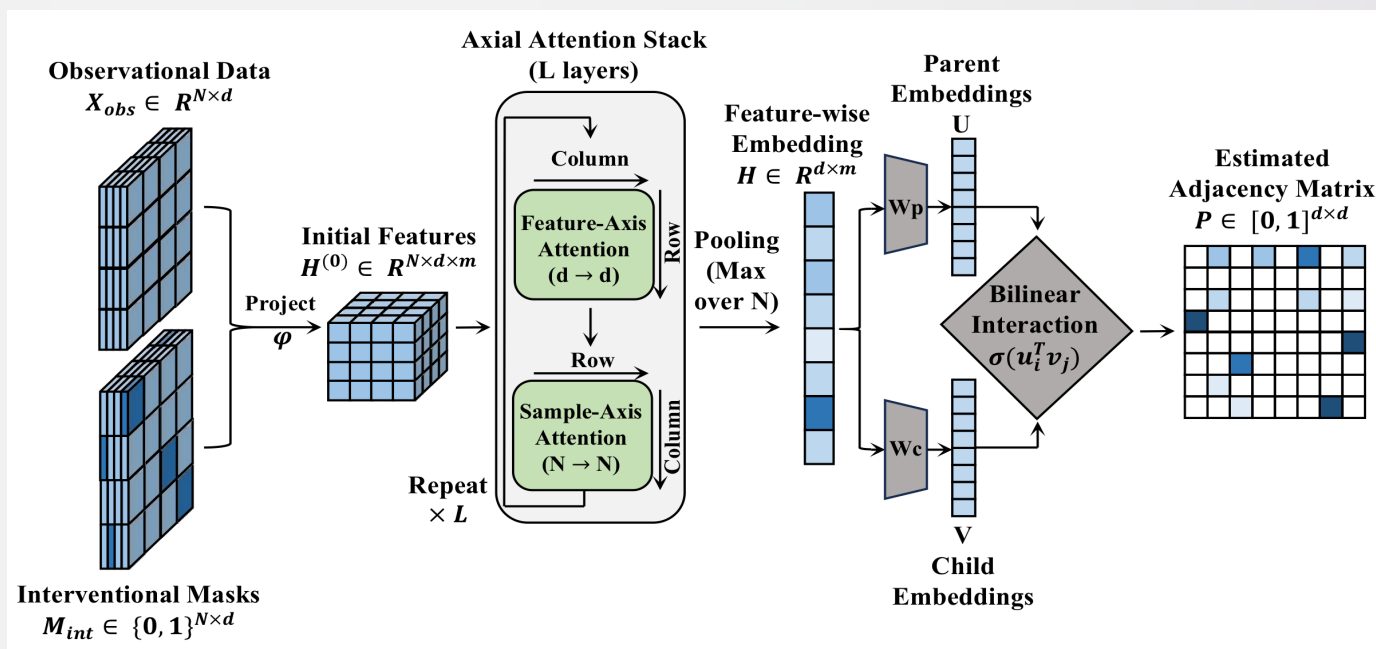


Goal: Recover the causal structure, i.e., hidden directed acyclic graph (DAG) behind the table.

Output: directed edge probabilities between variables.

Interventions test what changes when a variable is manipulated.

TabCausal is pre-trained over synthetic datasets. It predicts causal graphs directly from observational/interventional tables.

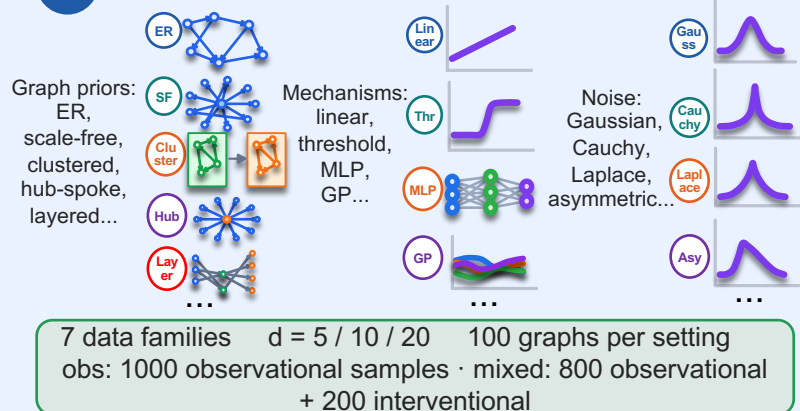


Axial attention over samples/features $\rightarrow$ parent/child embeddings $\rightarrow$ bilinear edge scores

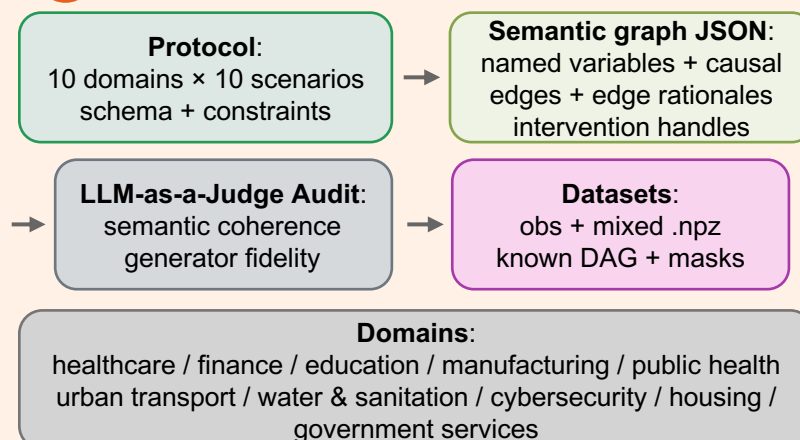
Evaluation of TabCausal

Benchmark construction

1 Synthetic benchmark

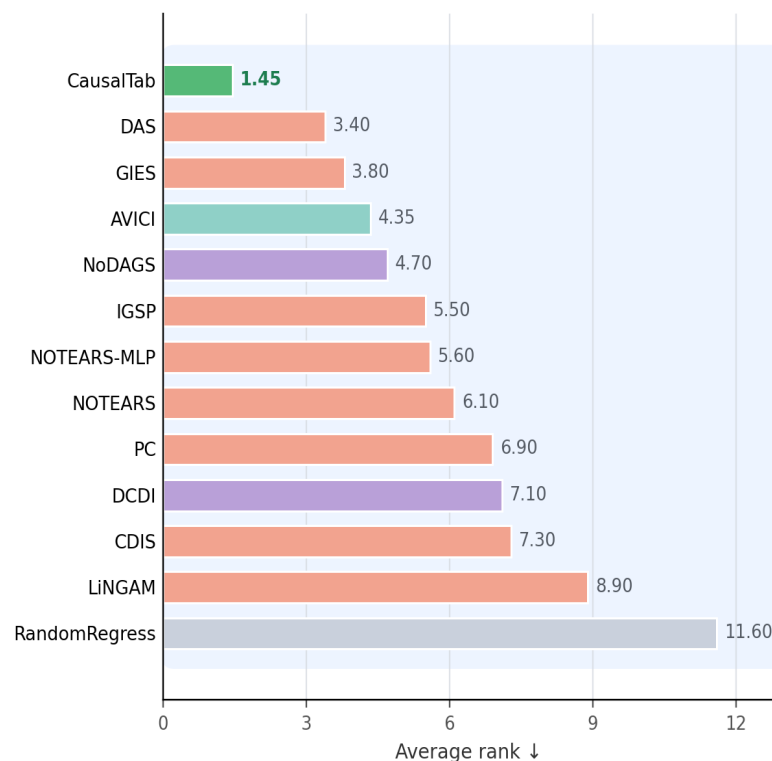


2 Semantic benchmark

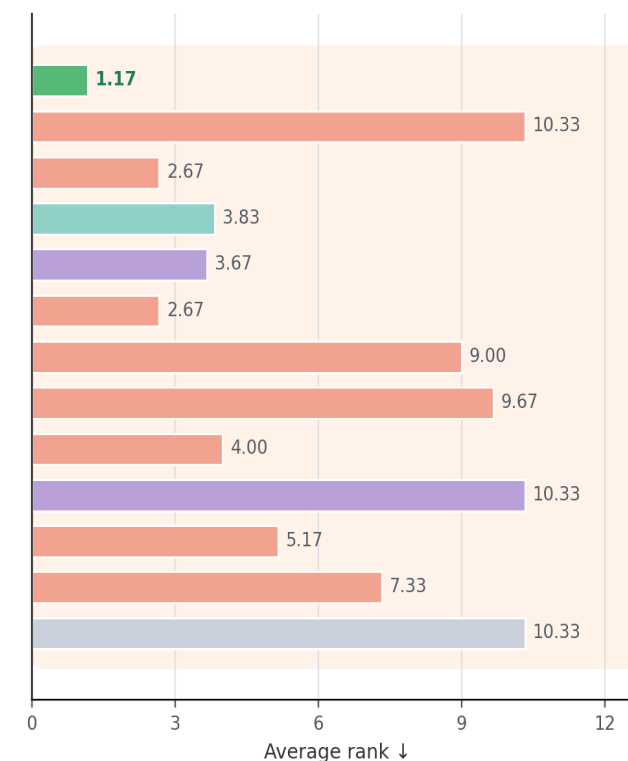


TabCausal demonstrates robust causal structure recovery on both benchmarks, especially when interventional evidence is available.

Main benchmark



Semantic benchmark



<https://github.com/LAMDA-Tabular/TabCausal>

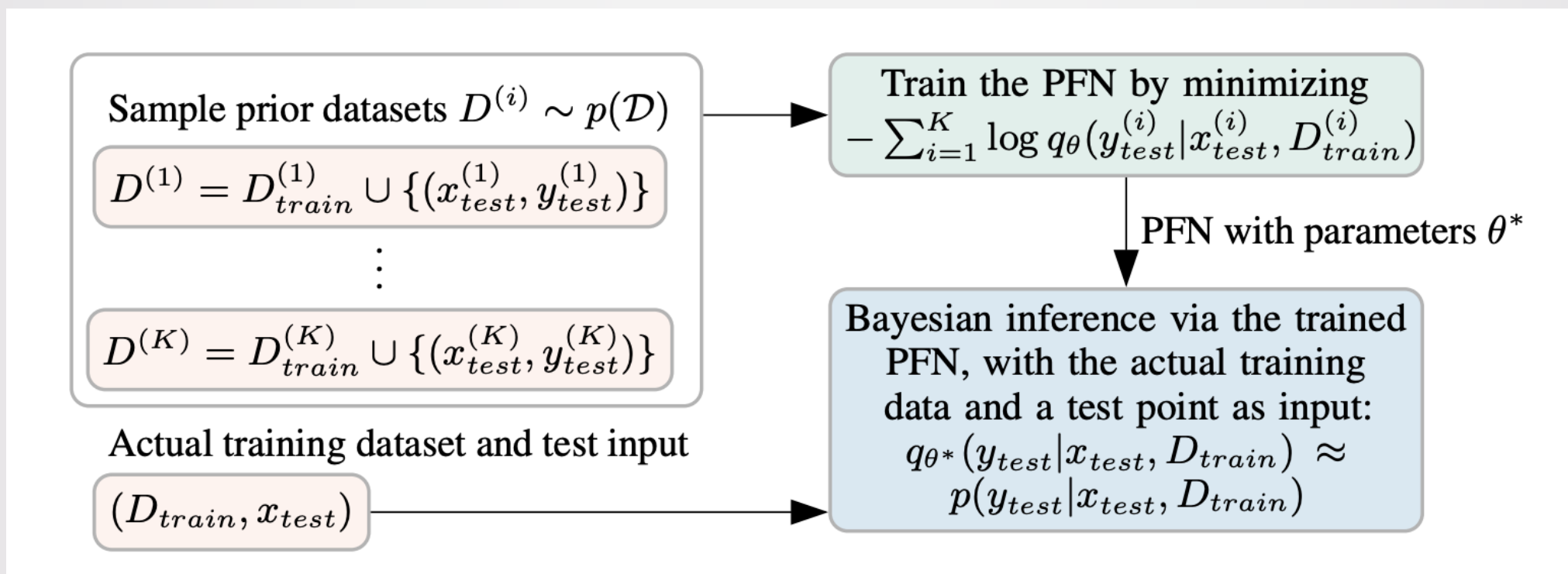
Why PFNs for Black-Box Optimization?

Black-box optimization (BBO) seeks the best input x using only expensive function evaluations $y=f(x)$.

- Each optimization run naturally forms a small table: configuration / design variables $\rightarrow$ measured objective value
- Classical Bayesian Optimization fits a surrogate model online, usually a Gaussian Process.
- PFNs replace repeated online fitting with a pretrained Transformer that performs Bayesian-style prediction in context.

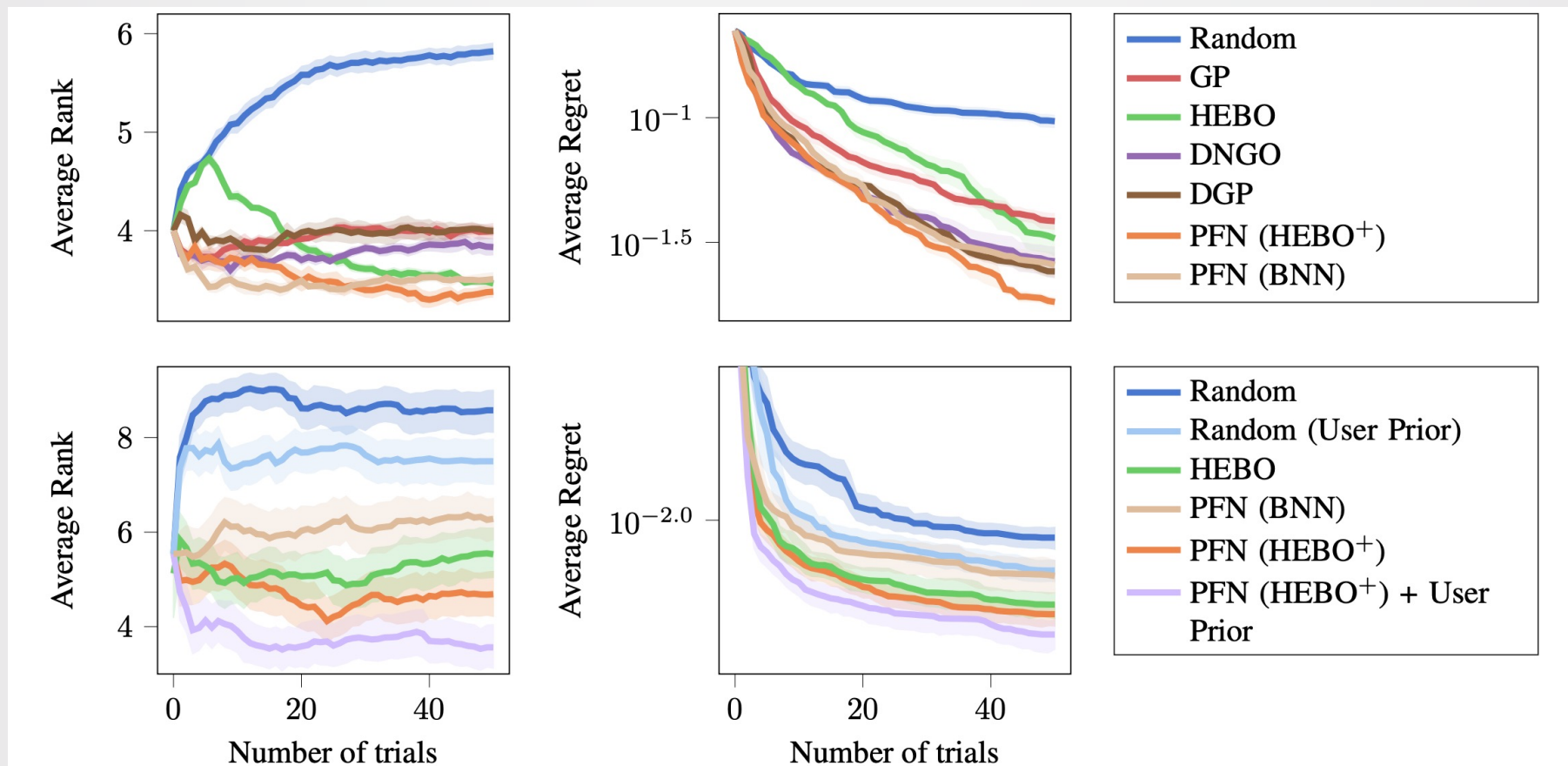
Tabular Foundation Models for Black Box Optimization

Transformers pretrained on synthetic datasets sampled from a prior over functions. Given a small context table $D = \{(x_i, y_i)\}$ and a new query point x_{test} , the PFN directly outputs a predictive distribution $q_{\theta}(y_{test} | x_{test}, D)$ in one forward pass.



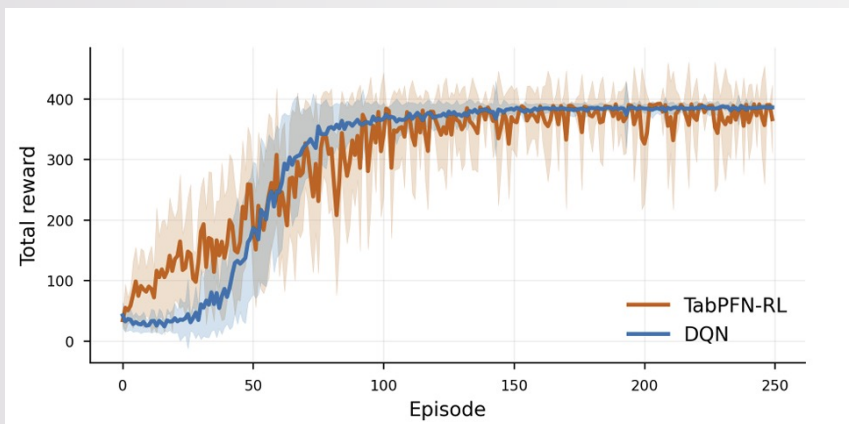
Tabular Foundation Models for Black Box Optimization

Instead of refitting a Gaussian Process after every new evaluation, PFNs4BO conditions on the current optimization history and predicts the posterior distribution for candidate points.



Tabular Foundation Models for RL

- Reformulates fitted Q-iteration as a small tabular regression problem: each transition is converted into an input–label pair where the input is a state–action pair and the label is a bootstrapped Bellman target.
- TabPFN is then used as an inference-only Q-function approximator, achieving competitive or better performance than DQN on classic-control benchmarks without gradient-based training.



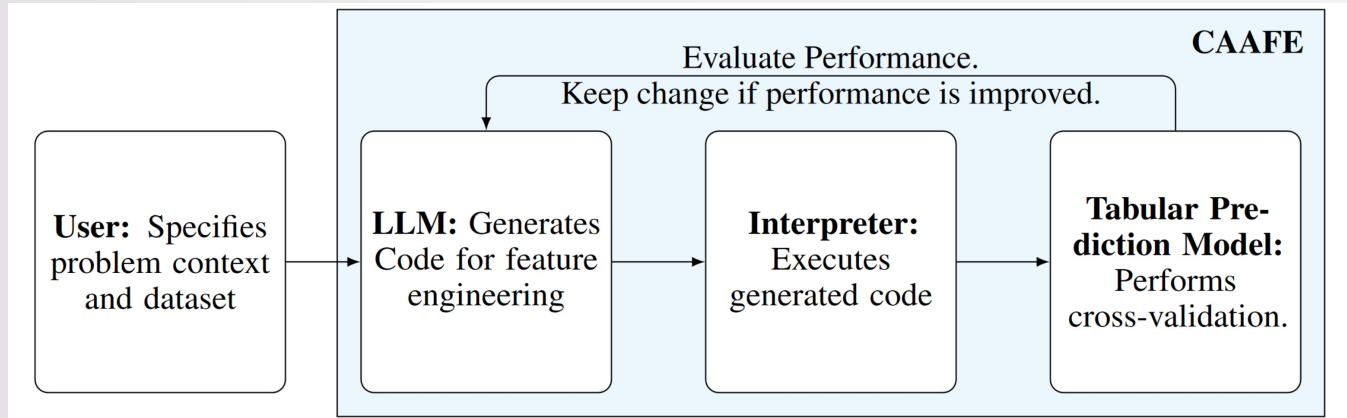
Performance on
reward per-episode
on Cartpole v1.

Algorithm 1: TabPFN-RL (gradient-free fitted Q with TabPFN)

Require: budget B ; initial N ; ε schedule $(\varepsilon_0, \lambda, \varepsilon_{\min})$

```
1:  $\mathcal{S} \leftarrow \emptyset$  /* context buffer */
2: for  $i = 1$  to  $N$  do
3:   Collect random transition  $(s, a, r, s', d)$ 
4:    $\mathcal{S} \leftarrow \mathcal{S} \cup \{(s, a, r, s')\}$ 
5: end for
6:  $Q_\theta \leftarrow \text{TABPFN-FQI}(\mathcal{S})$  /* first fit */
7: for episode  $e = 1, 2, \dots$  do
8:    $\varepsilon \leftarrow \max(\varepsilon_{\min}, \varepsilon_0 \lambda^e)$ 
9:   Run episode  $\mathcal{E}$  with  $\pi(s) = \arg \max_a Q_\theta(s, a)$ 
10:   $R_\mathcal{E} \leftarrow \sum_{(s,a,r) \in \mathcal{E}} r$  /* Get episode reward and compare */
11:  if  $R_\mathcal{E} > \text{Quantile}_{0.95}(\{R_{\text{hist}}\})$  then
12:    if  $|\mathcal{S}| < B$  then
13:       $\mathcal{S} \leftarrow \mathcal{S} \cup \mathcal{E}$ 
14:       $Q_\theta \leftarrow \text{TABPFN-FQI}(\mathcal{S})$ 
15:    end if
16:  end if
17: end if
18: end for
```

Tabular Foundation Models for Feature Engineering



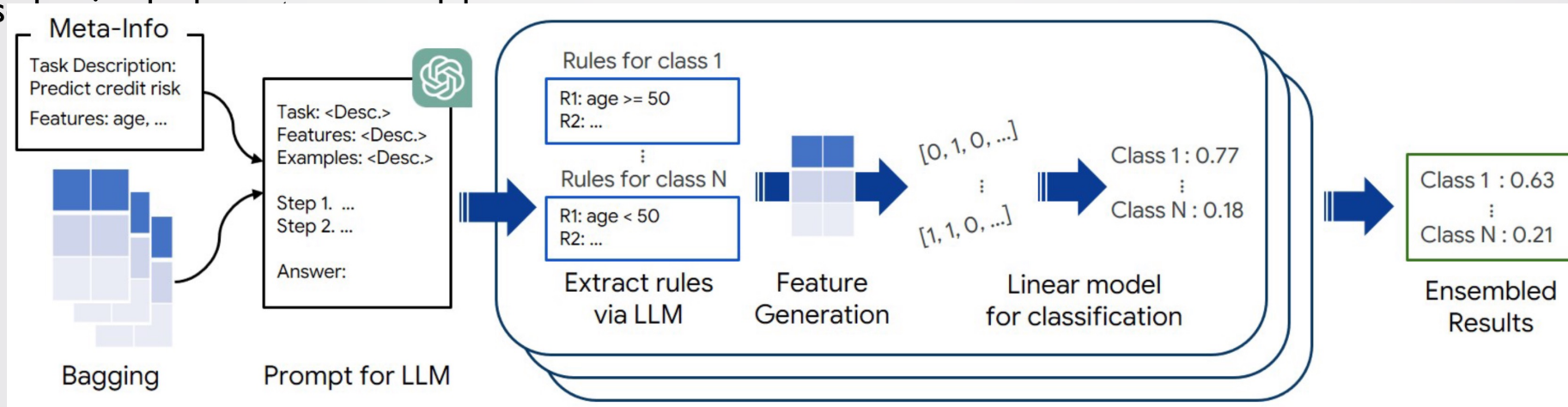
		No FE	DFS	Baselines			CAAFE	
				AutoFeat	FETCH	OpenFE	GPT-3.5	GPT-4
Log. Reg.	Mean	0.749	0.764	0.754	0.76	0.757	0.763	0.769
	Mean Rank	27.4	23.6	26.2	25.2	25	24.8	24.3
Random Forest	Mean	0.782	0.783	0.783	0.785	0.785	0.79	0.803
	Mean Rank	23.4	22.1	21.8	23.5	22.3	23.1	19.9
ASKL2	Mean	0.807	0.801	0.808	0.807	0.806	0.815	0.818
	Mean Rank	12.2	12.9	12.6	13.4	13.5	10.9	11.6
Autogluon	Mean	0.796	0.799	0.797	0.787	0.798	0.803	0.812
	Mean Rank	17.6	15.4	16.4	17.6	16.6	15.8	14.1
TabPFN	Mean	0.798	0.791	0.796	0.796	0.798	0.806	0.822
	Mean Rank	13.9	15	14.8	16.5	13.9	12.9	9.78

- **Prompt Construction:** Combine the dataset's contextual description, feature names, data types, and a few-shot samples into a prompt.
- **LLM Code Generation:** Generate Python code to create new features, accompanied by a text explanation of the purpose of these features.
- **Code Execution and Evaluation:** Integrate the new features into the dataset and evaluate the resulting performance on the validation set using a predictive model.
- **Iterative Optimization:** Retain new features if they improve performance, then direct the LLM to continue creating additional features based on the updated dataset.

Tabular Foundation Models for Feature Engineering

Using LLM to Create Decision Rules

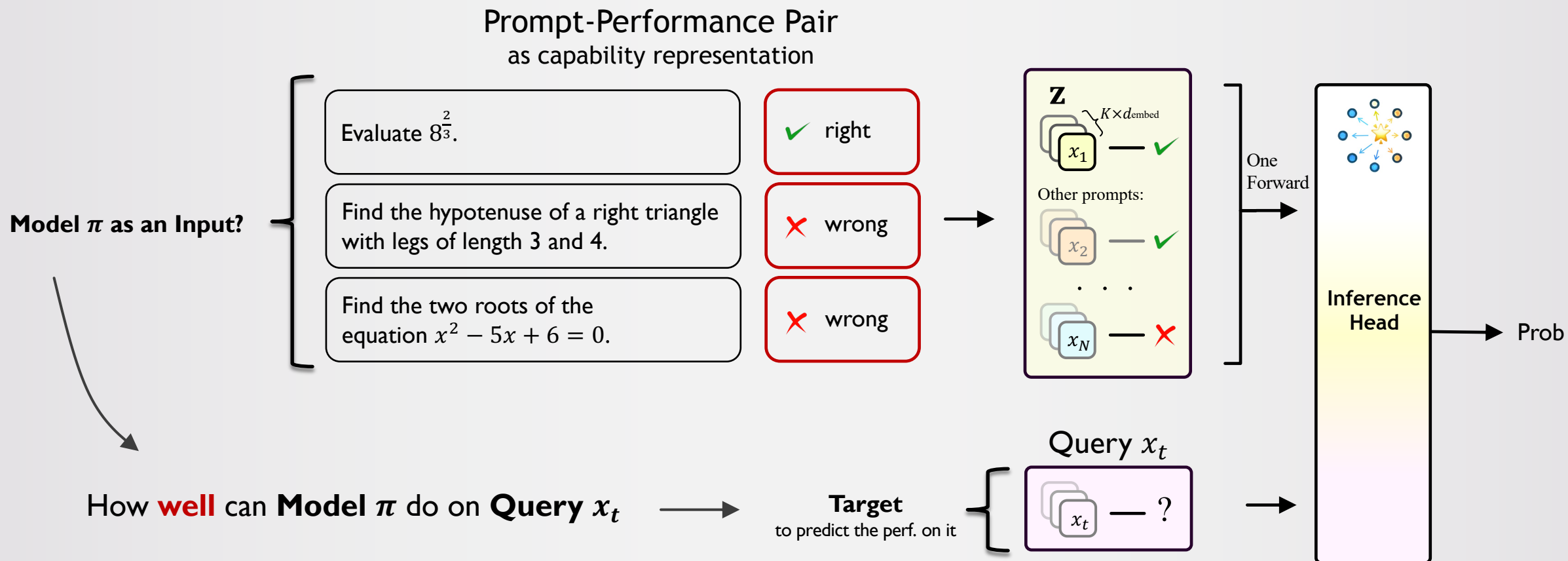
- **Research Problem:** Current LLM methods for few-shot tabular prediction suffer from high inference costs and are limited by context window constraints.
- **Solution:** Enable LLMs to extract classification rules from a small number of samples and generate new features for us



- **Rule Extraction:** Use prompts to guide the LLM in distilling classification rules from a few samples.
- **Feature Generation:** Invoke the LLM to convert these rules into code, which is then used to generate a new binary feature matrix.
- **Prediction and Ensemble:** Train simple models on the new features and enhance robustness through Bagging and Ensemble techniques.

Tabular Foundation Models for Model Routing

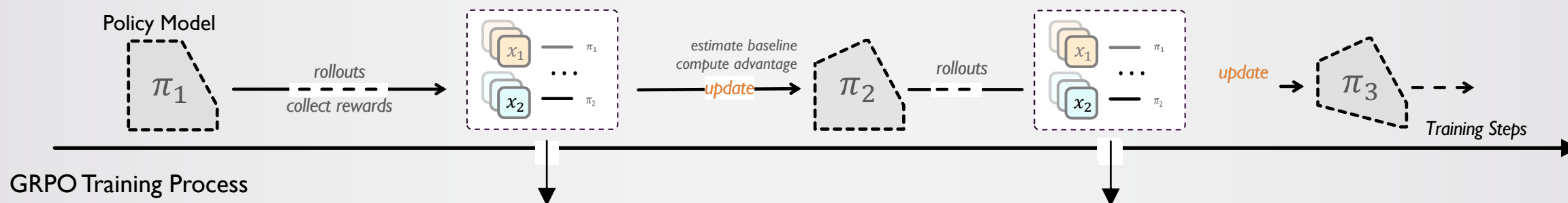
Core Idea: Encode Model Capabilities to the Input Context



Tabular Foundation Models for Model Routing

How to Construct the Pre-trained Data: Requiring Models with Varying Capabilities

We use GRPO training in advance to obtain different checkpoints with its rollouts.



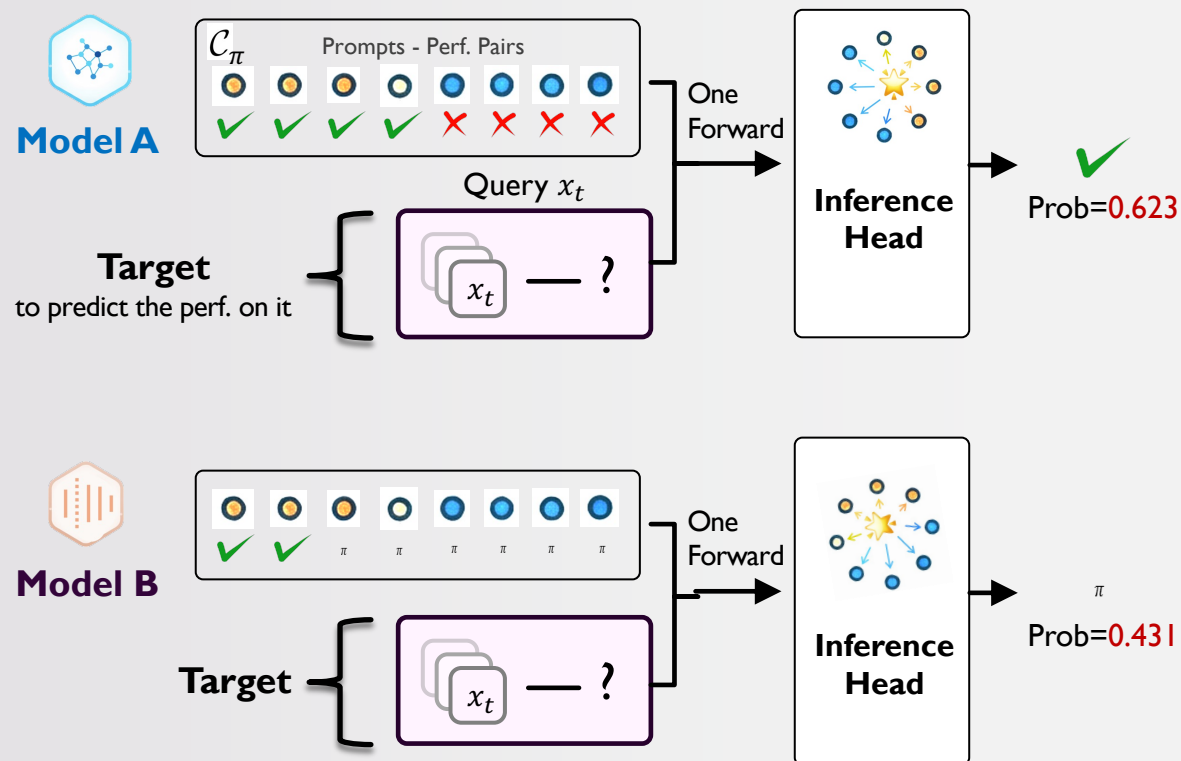
Prompt-Performance Training Pairs

Composition of Pre-Train Data (for more capabilities)

Model		X	GRPO Training Trajectories	
Qwen3-0.6B	Qwen3-32B		Each trajectory contains	with 1536 context data pool
Qwen3-1.7B	Qwen3-4B-Thinking-2507		200+ checkpoints	and ~500 training queries
Qwen3-8B	Qwen3-30B-A3B-Thinking-2507			
Qwen3-14B	Qwen3-235B-A22B-Instruct-2507			

Tabular Foundation Models for Model Routing

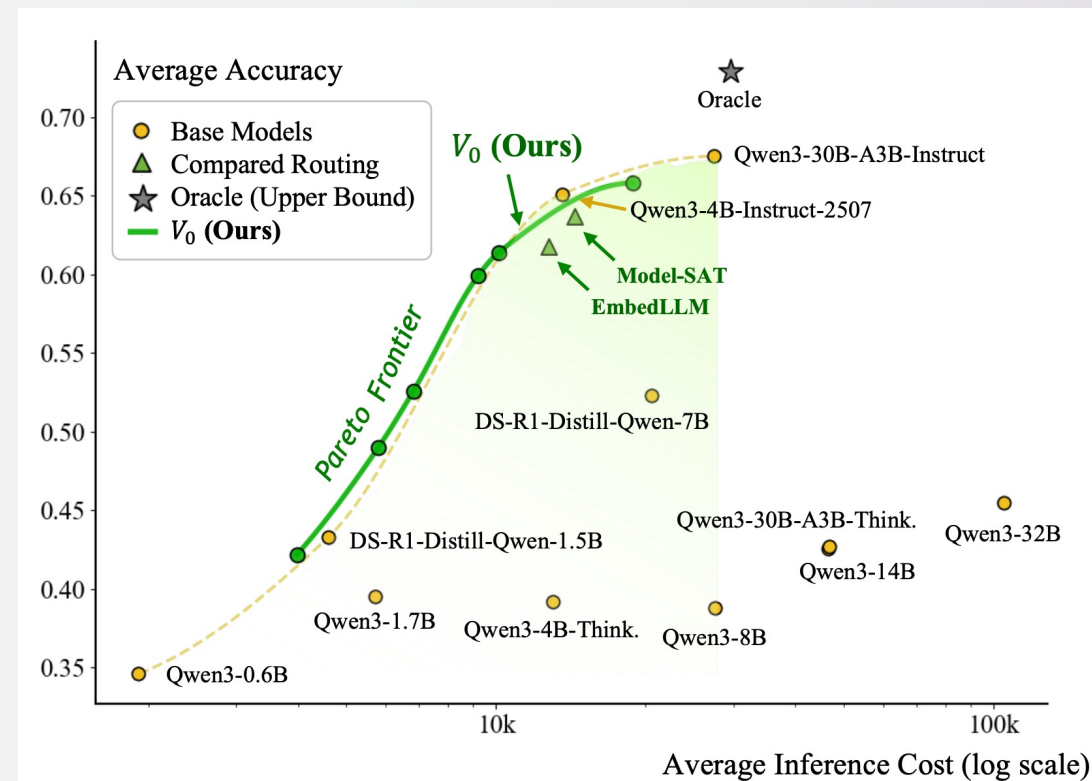
Details of the Routing Process



So, **Model A** more likely to do better than **Model B**

Routing Results

with controlled **Cost-Perf. weight** in context

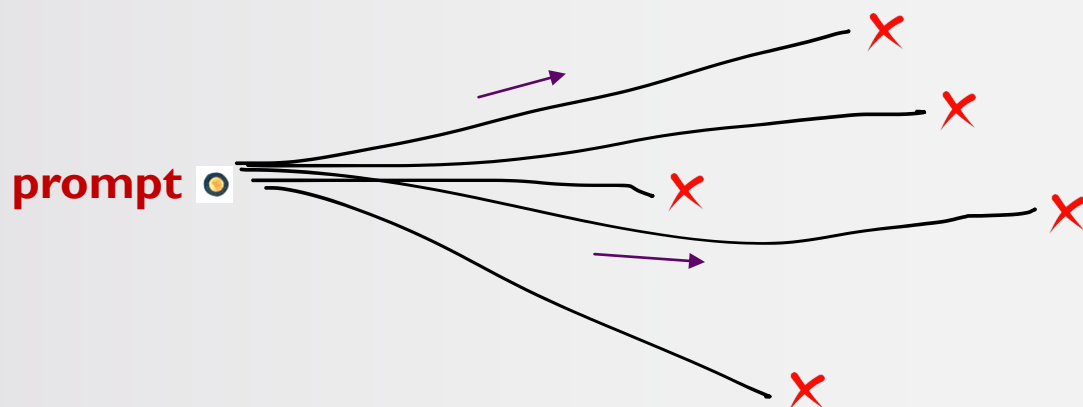


Test Results from avg. of 12 datasets, including addsub, aime 2024/2025, amc23, college math, gaokao 2023en, gaokao mathCloze, gpqa, math hard, math500, minerva math, olympiad.

Tabular Foundation Models for GRPO Budget Allocation

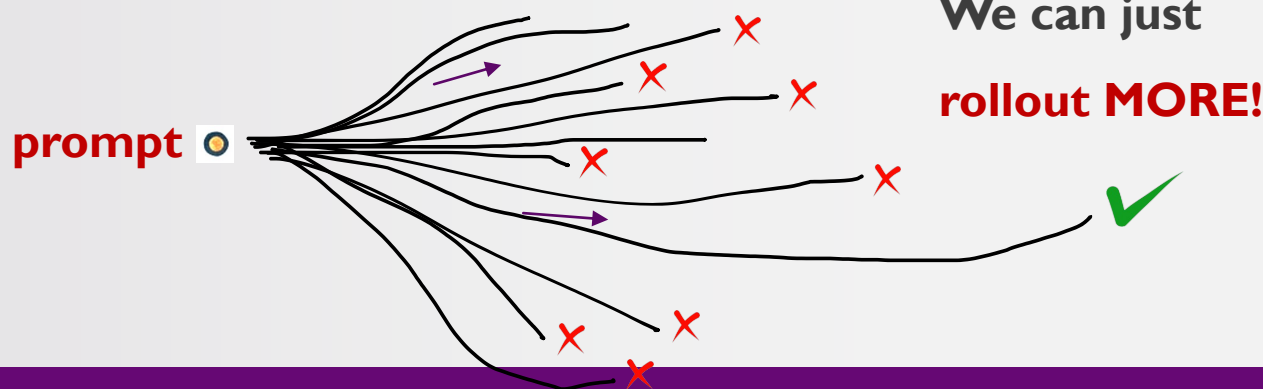
Details of the Budget Allocation

GRPO fails when all rollout results are consistent (entropy collapse).

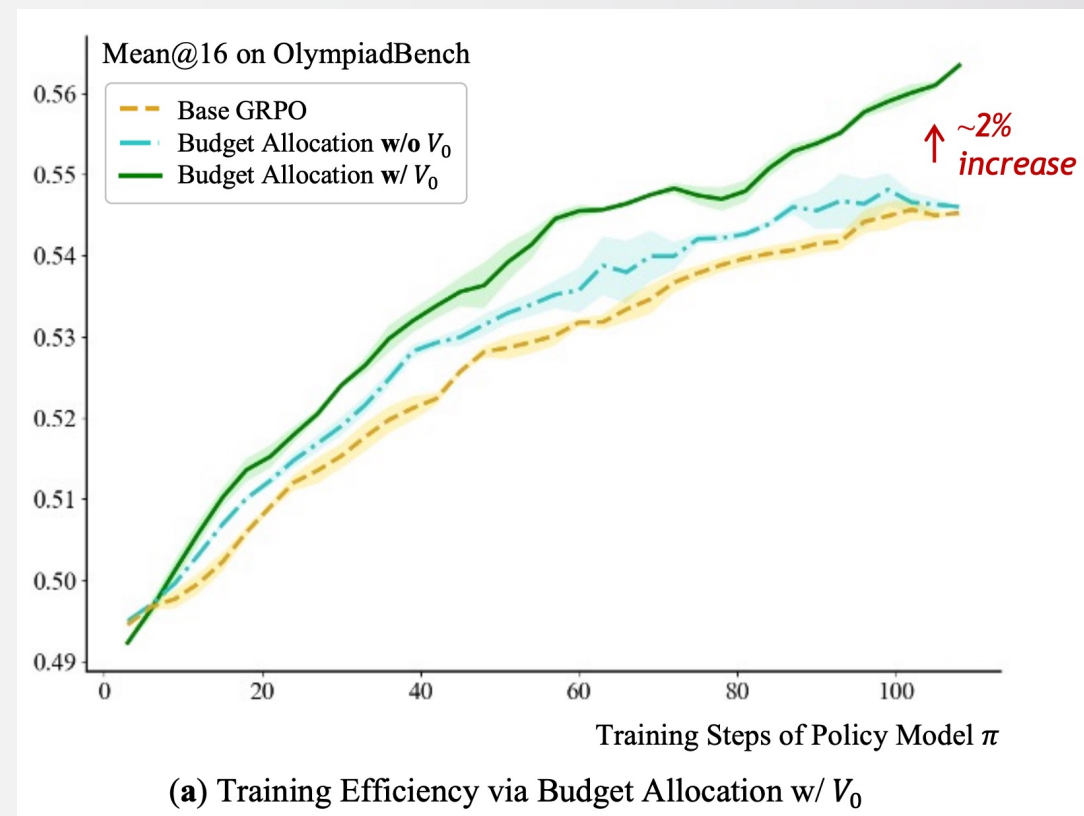


If the General Performance Predictor said:

It is a **difficult** prompt!



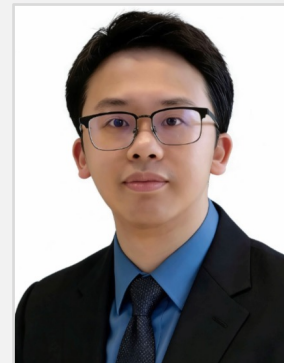
Training w/ Budget Allocation Results



Q&A

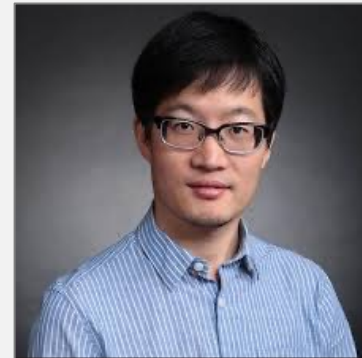


Presenter:
Prof. Han-Jia Ye
Associate Professor
School of Artificial Intelligence
Nanjing University
Email: yehj@nju.edu.cn



Presenter:
Prof. Jintai Chen
Assistant Professor
AI Thrust, Information Hub
The Hong Kong University of Science
and Technology (Guangzhou)
Email: jintaiCHEN@hkust-gz.edu.cn

Open Problems and Future Directions



Presenter:

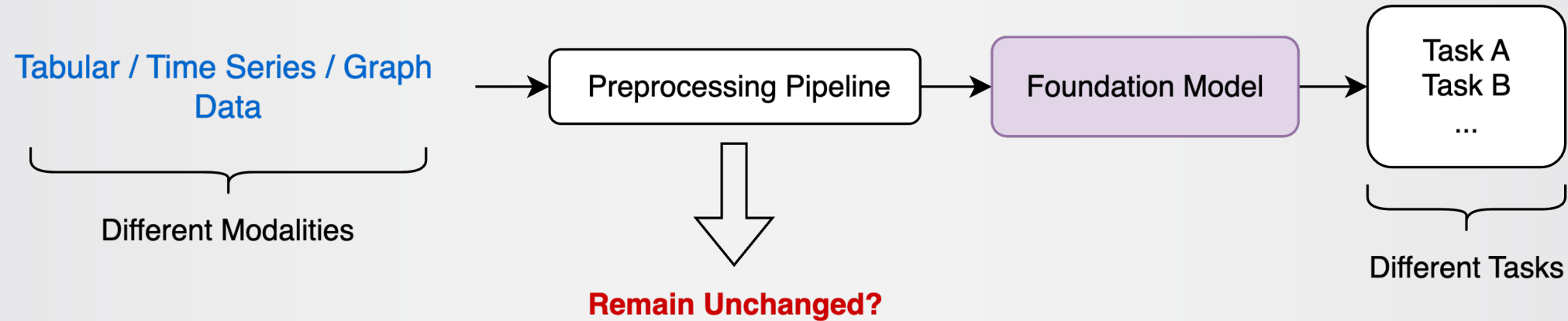
Prof. Peng Cui

Associate Professor, IEEE Fellow
Department of Computer Science
Tsinghua University
Email: cuip@tsinghua.edu.cn

For tabular foundation models, there are still many challenges that must be addressed:

- **Task Unification:** Effectively unifying different modeling tasks with a single model remains a challenging issue, as multi-task learning can be difficult to implement without compromising the performance of individual tasks;
- **Data Generation:** There is no widely accepted generation strategy, and evaluating the quality of generated data remains a challenge;
- **Scalability:** Current models are still relatively small, and the relationship between model size and performance remains largely unexplored, particularly for large-scale models;
- **Adaptation Across Modalities:** Up to date, multimodal task modeling relies on processing pipelines tailored to each specific modality.

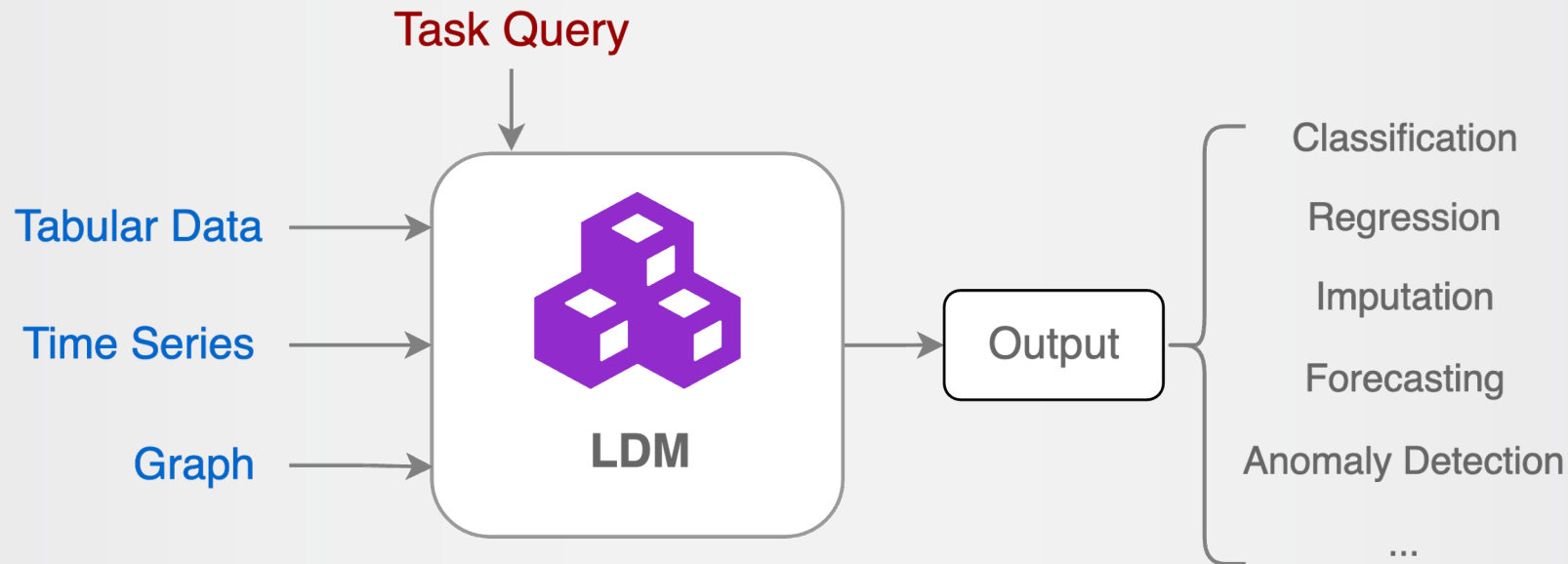
The Pipeline Paradox: A Blessing or A Nightmare?



As the number of modalities and tasks grows, the situation changes considerably :

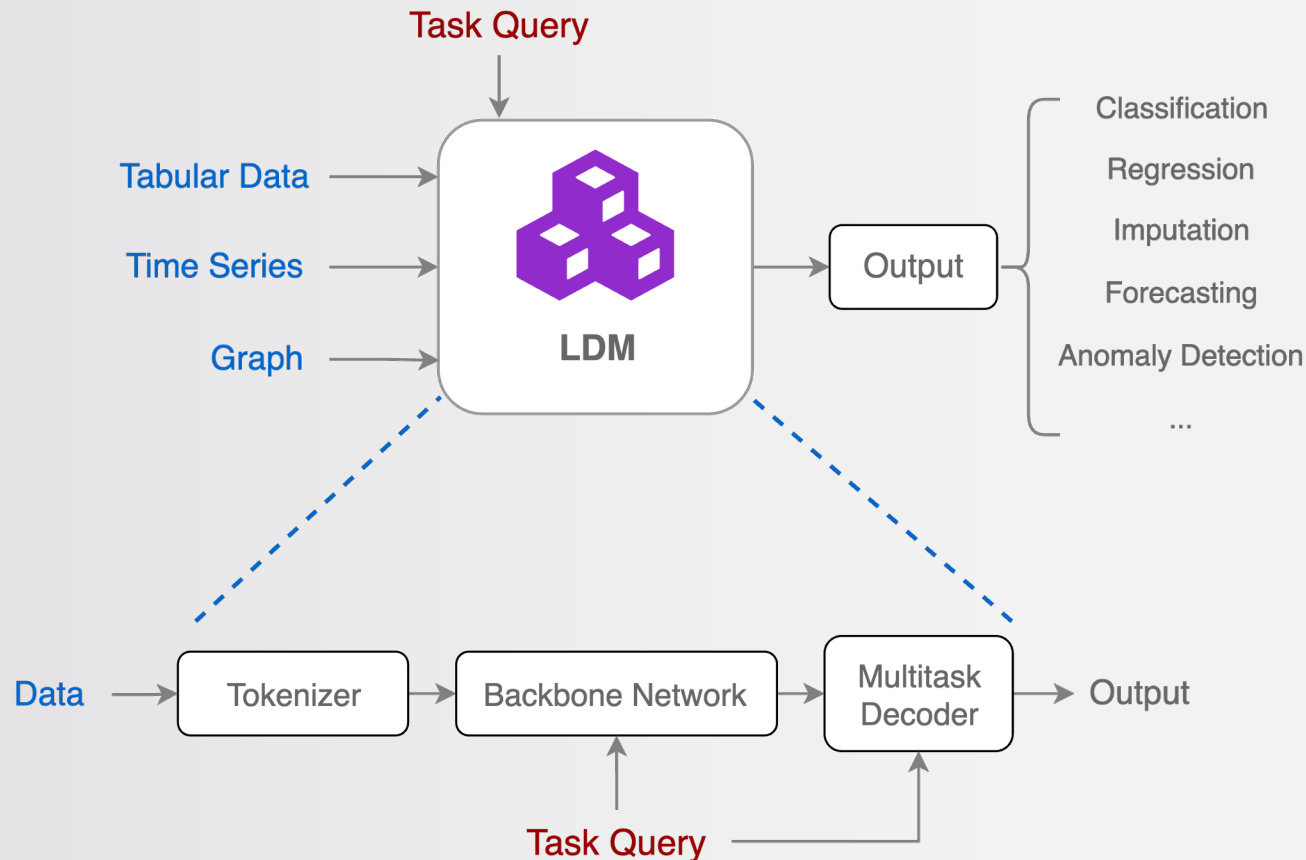
- Each modality may require a distinct preprocessing pipeline;
- Different tasks within the same modality may not be able to share a common pipeline;
- Integrating information from multiple modalities within the pipeline paradigm is both challenging and unnatural.

Large Structured-Data Models (LDMs)



This gap motivates the development of an LDM that treats structured data as a joint distribution over variables and modalities, allowing multiple tasks to be framed as queries to a single model.

Large Structured-Data Models (LDMs)



LDMs consist of three main components:

- **Tokenizer**: Converts data into a format suitable for model processing;
- **Backbone Network**: Processes and learns patterns from the data;
- **Multitask Decoder**: Allows the model to perform different tasks simultaneously (e.g., classification, regression, forecasting).

In terms of LDM model design, several key aspects need attention:

- **Tokenizer:** Given the distinct nature of tabular data, time series, and graphs, it is crucial to account for the specific dependencies inherent to each modality;
- **Backbone Network:** The backbone needs to create shared representations that capture and fuse the essential features of each data type;
- **Multitask Decoder:** The decoder should be capable of handling different tasks for each modality, accounting for the presence or absence of specific modalities;
- **Scalability:** When handling three modalities with different characteristics, it remains unclear whether a consistent scaling law can be observed.

Open Problems: Aligning Modalities

Aligning tabular, time-series, and graph data is complicated by their distinct structures:

- **Tabular data:** Features organized in rows and columns;
- **Time-series:** Sequences of values over time with temporal dependencies;
- **Graph:** Nodes connected by edges, each with unique relationships.

The challenge is to balance modality-specific features (e.g., temporal dependencies in time-series, node relationships in graphs) with a shared representation that is effective for multitask learning.

Open Problems: Preparing Training Data

Proper training data is crucial to ensure that the model can effectively learn patterns across modalities:

- **Data Generation:** For these modalities, publicly available high-quality data may be insufficient to train a foundation model, particularly when the model requires all modalities as input simultaneously. In such cases, having a robust data generation pipeline becomes crucial;
- **Data Augmentation:** For each modality, the augmentation methods can vary significantly, making it challenging to identify a comprehensive data augmentation strategy within a large search space;

Open Problems: Pretraining Strategy

The pretraining of LDMs is still largely unexplored:

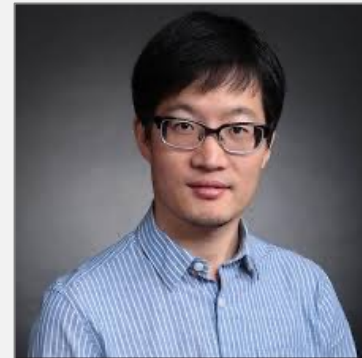
- **Multitask Pretraining:** Features organized in rows and columns;
- **Data Selection:** During training, determining the optimal ratio between real and synthetic data is crucial for improving model performance, with the ideal setting varying for each modality;
- **Training Strategy:** It remains unclear whether LDMs should undergo end-to-end training for all tasks and modalities, or if the training should be divided into multiple stages.

- The development of tabular foundation models is still in its early stages, with exciting potential to unlock more efficient data handling and better generalization across domains;
- LDMs have natural advantages in multimodal modeling, but key challenges such as scalability, cross-modal alignment, and multitask learning remain;
- Collaboration between academia and industry will be vital to solve these challenges and push the boundaries of what's possible in structured-data modeling.

Do not go where the path may lead, go instead where there is no path and leave a trail.

— Ralph Waldo Emerson

Q&A



Presenter:

Prof. Peng Cui

Associate Professor, IEEE Fellow
Department of Computer Science
Tsinghua University
Email: cuip@tsinghua.edu.cn



Thank You

