TabPFN

Outline

- Background
- TabPFN
- Results

Outline

- **Background**
- TabPFN
- Results

Prediction

Given observed data $\mathcal{D} = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^n$ and new covariates $\boldsymbol{x}_{n+1}$,

predict the response y_{n+1}.

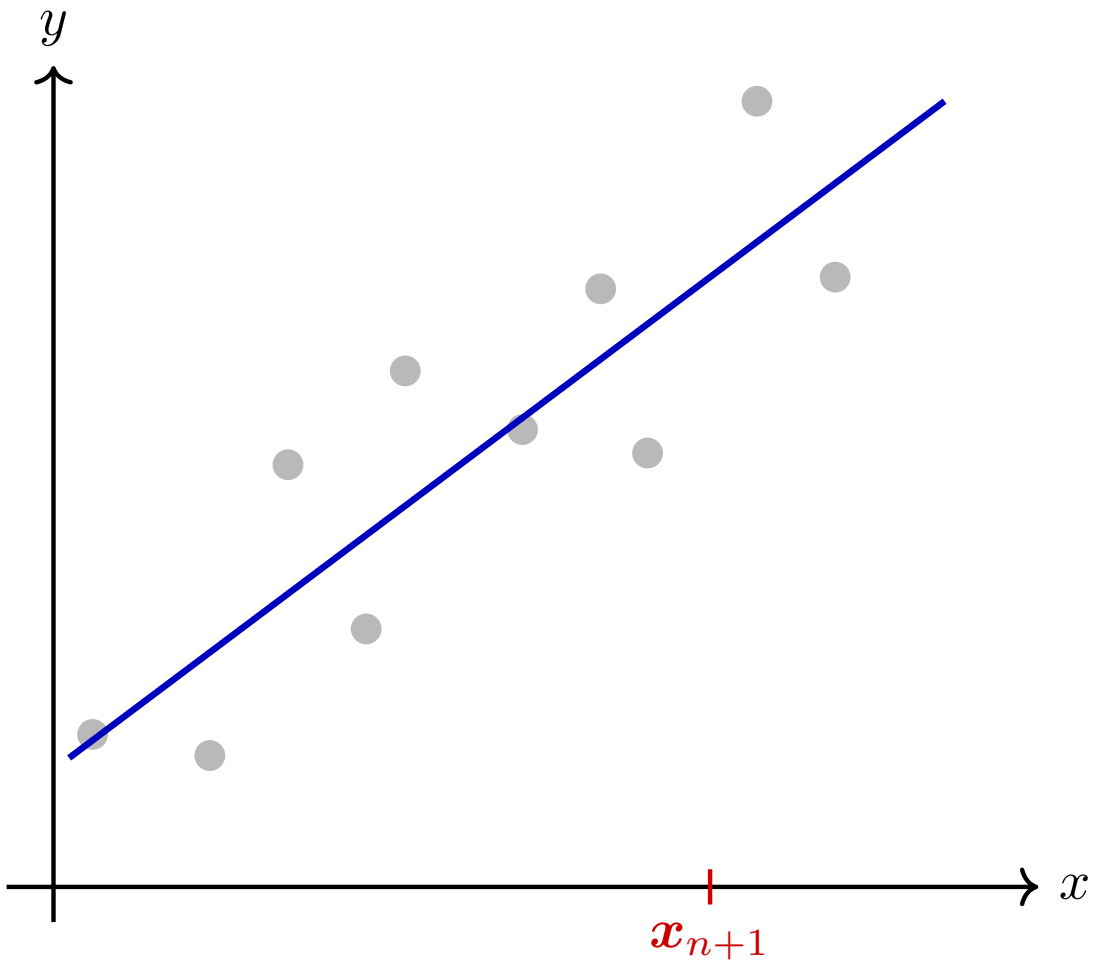
Tabular data

	x_1	x_2	$\cdots$	x_d	y
1	2.1	0	$\cdots$	1.7	4.5
2	0.8	1	$\cdots$	0.4	2.1
$\vdots$	$\vdots$	$\vdots$		$\vdots$	$\vdots$
n	1.4	1	$\cdots$	2.3	3.8

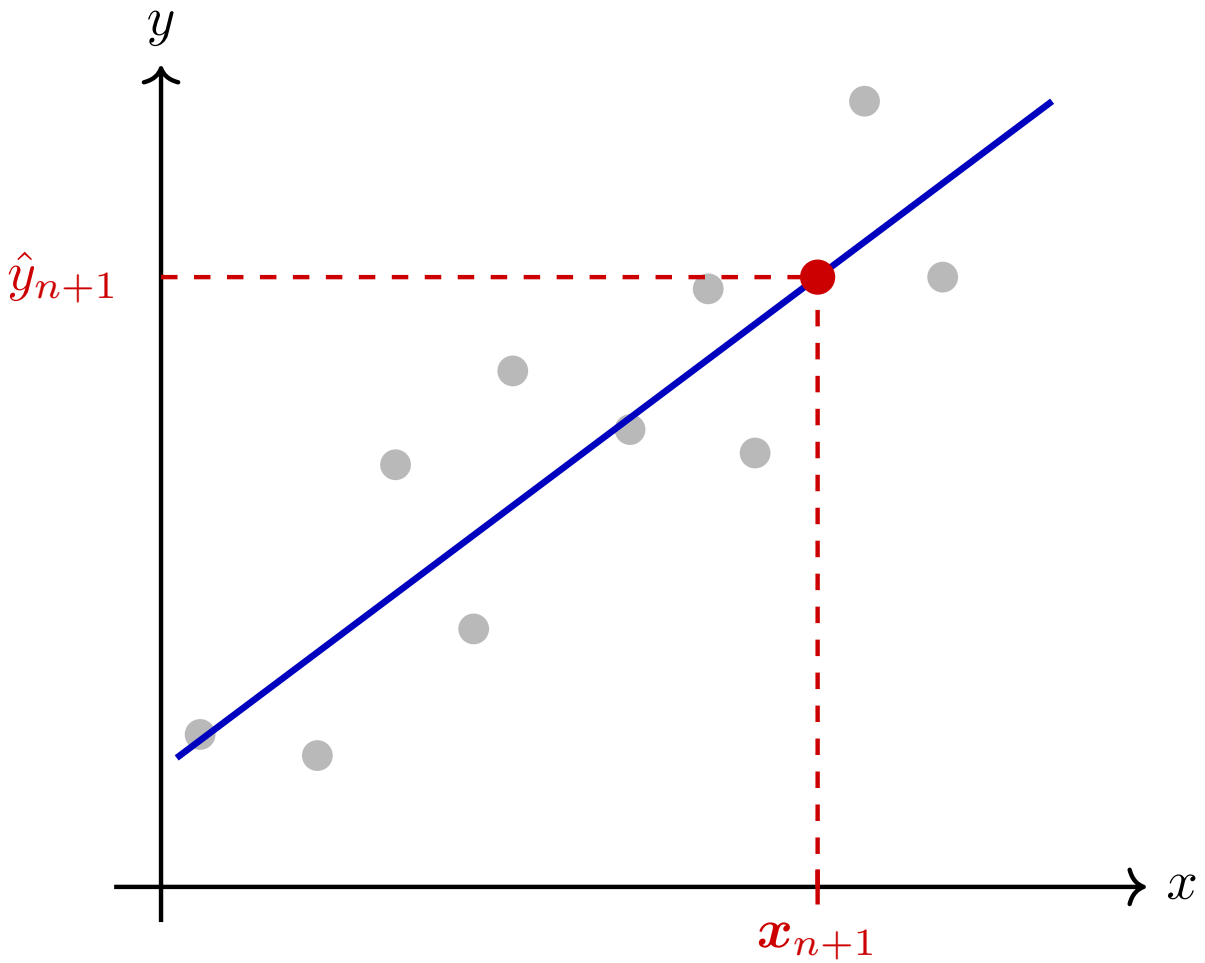
Tabular data

	x_1	x_2	$\cdots$	x_d	y
1	2.1	0	$\cdots$	1.7	4.5
2	0.8	1	$\cdots$	0.4	2.1
$\vdots$	$\vdots$	$\vdots$		$\vdots$	$\vdots$
n	1.4	1	$\cdots$	2.3	3.8
$n+1$	3.2	0	$\cdots$	1.1	?

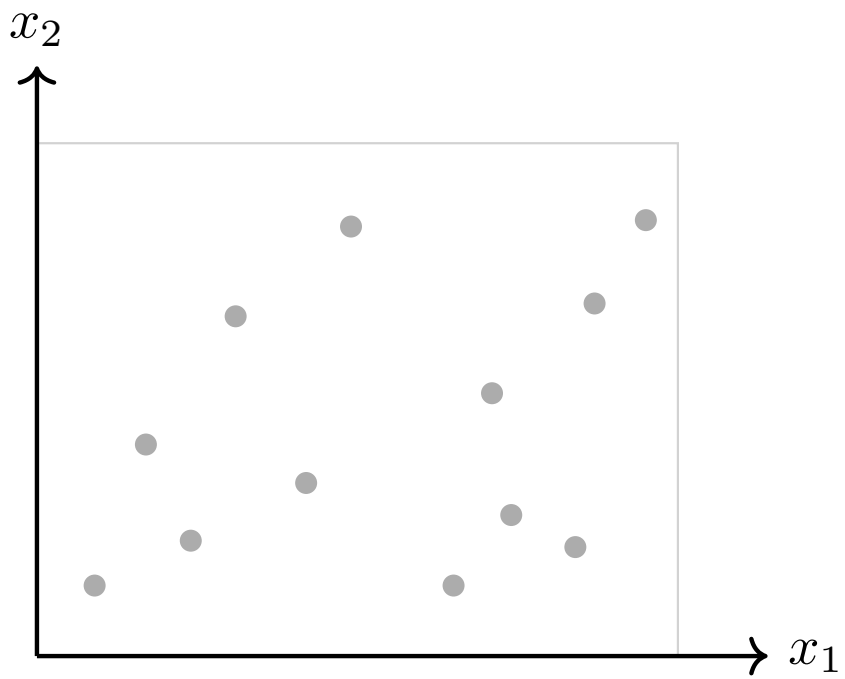
Linear regression



Linear regression

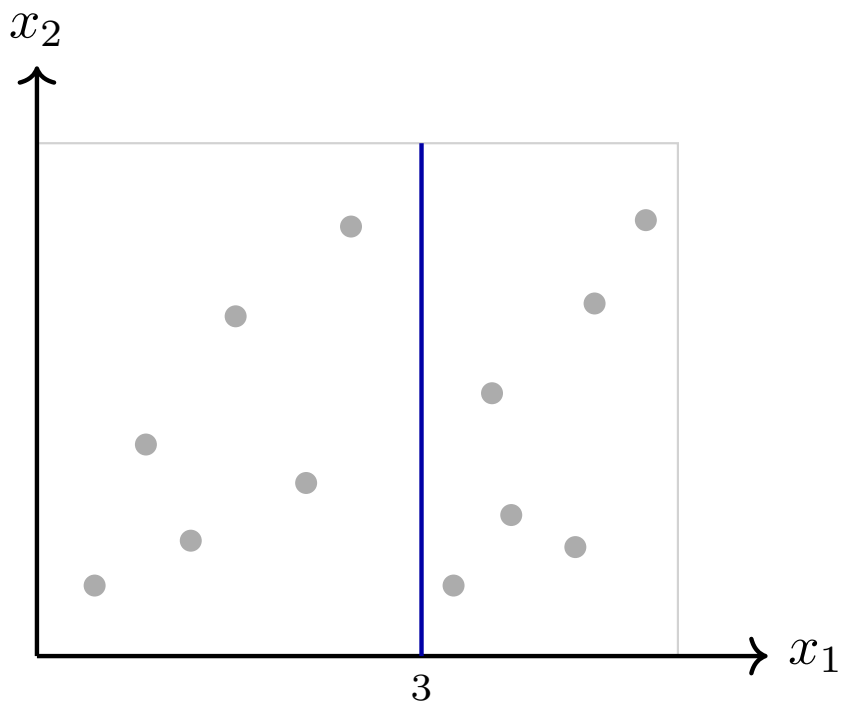


Tree-based methods

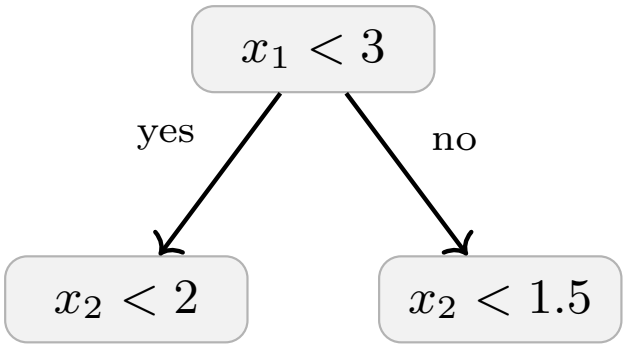


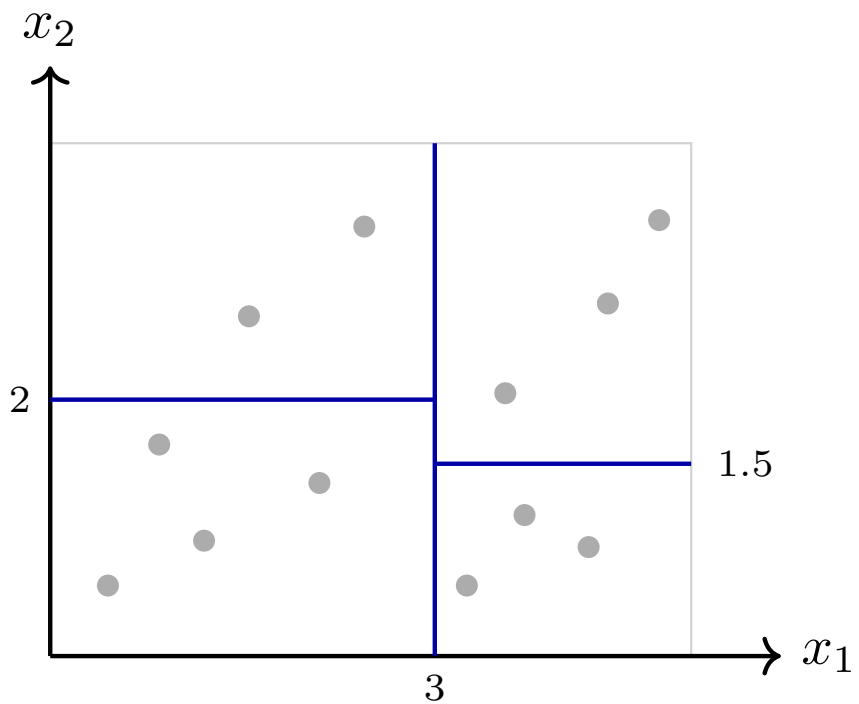
Tree-based methods

$x_1 < 3$

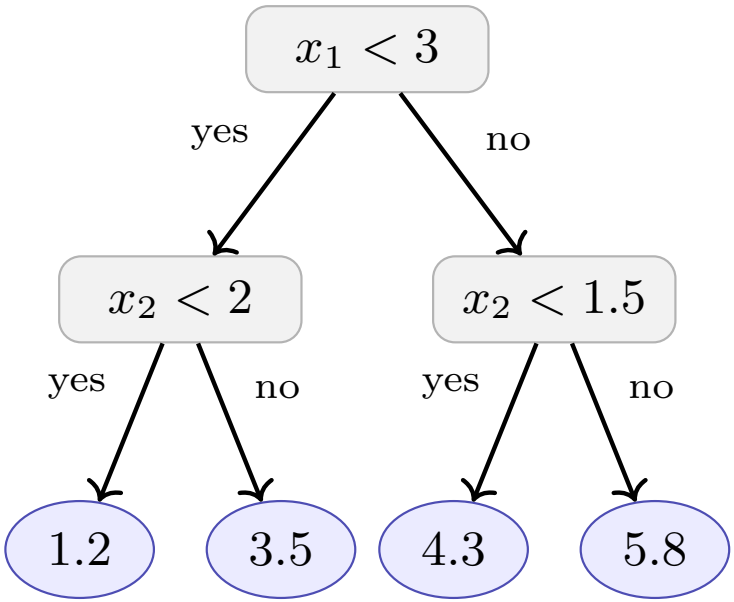


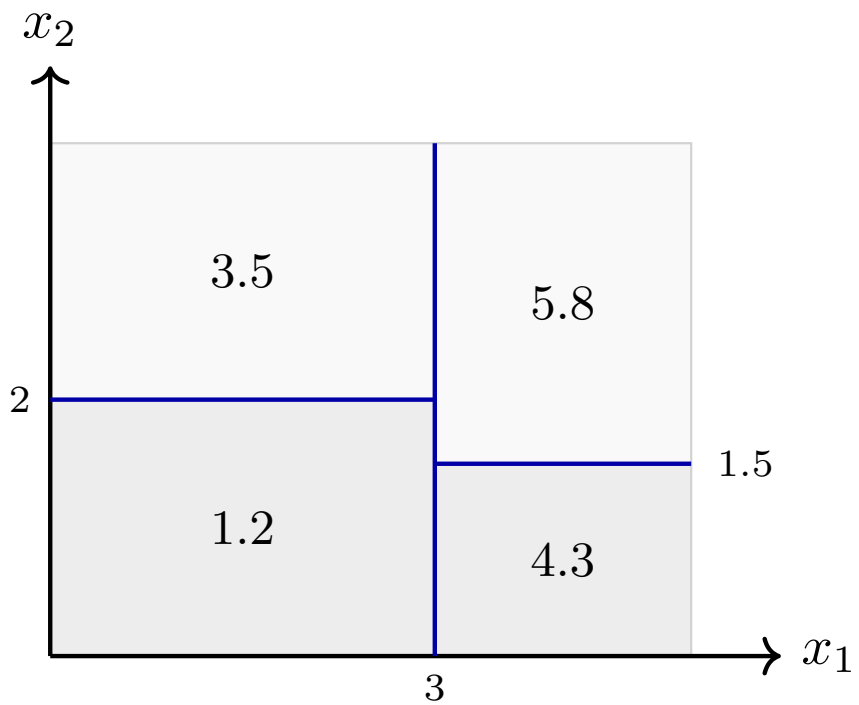
Tree-based methods



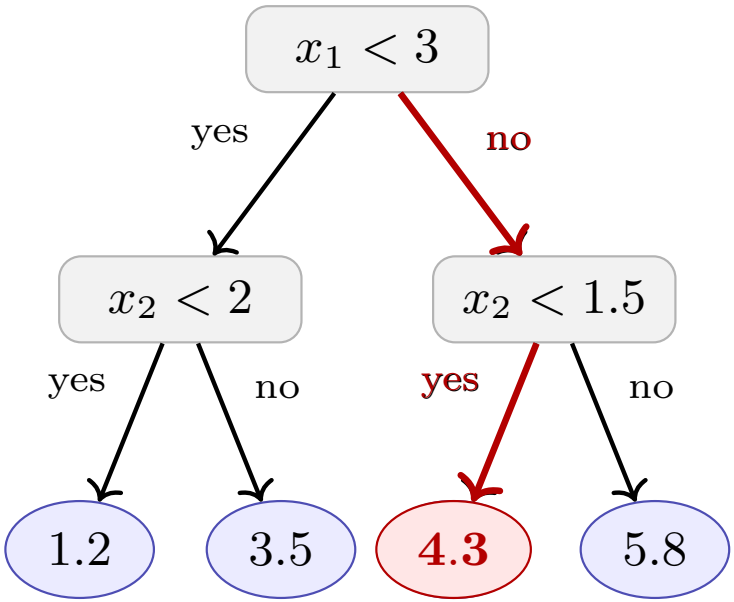


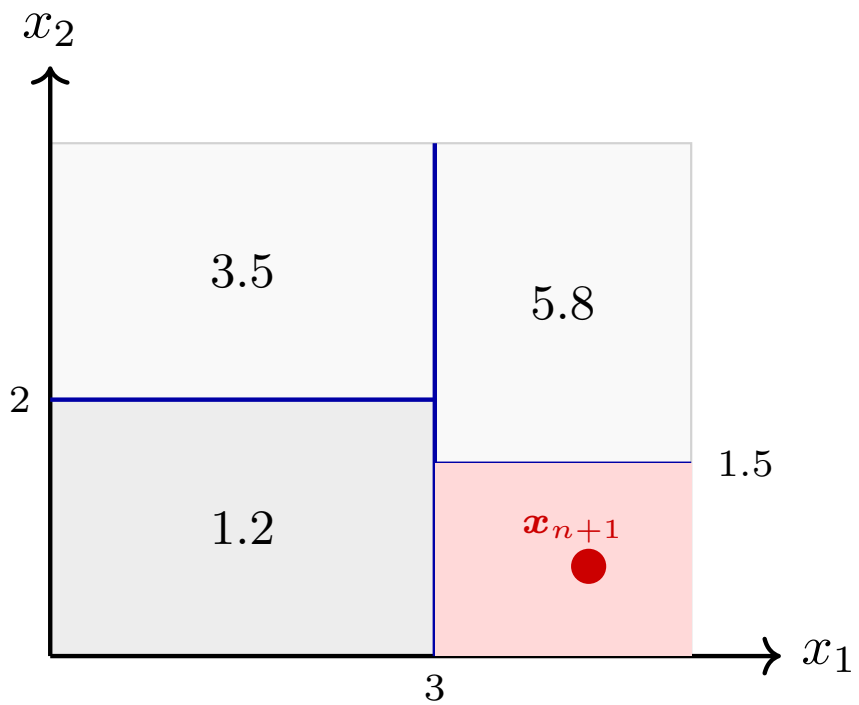
Tree-based methods



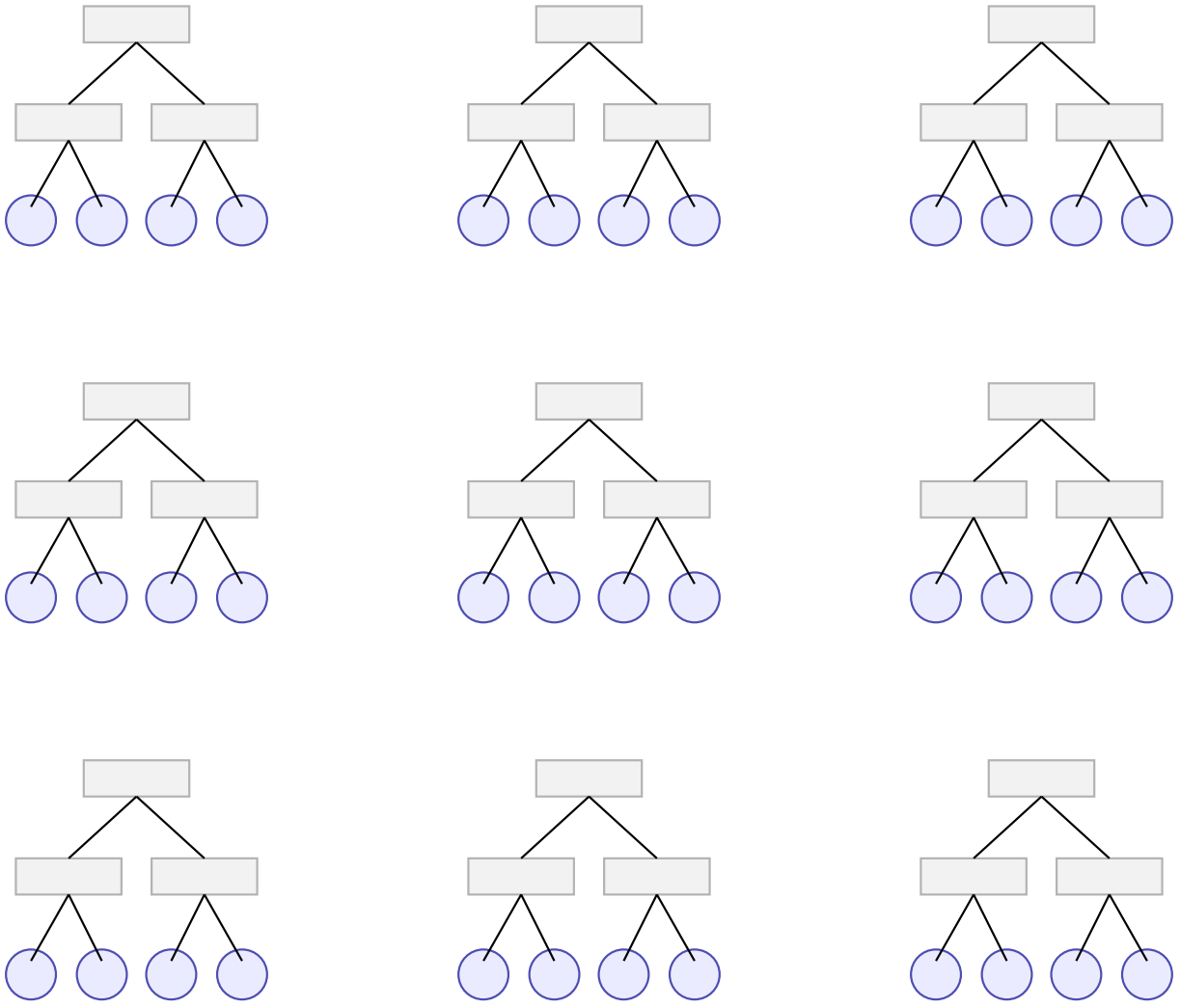


Tree-based methods





Tree-based methods

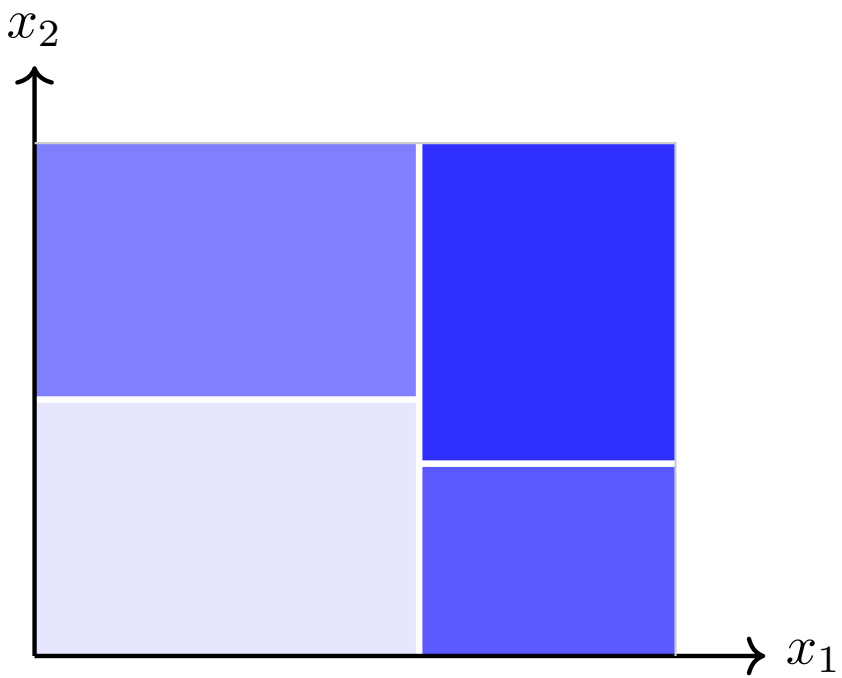


Tree-based methods

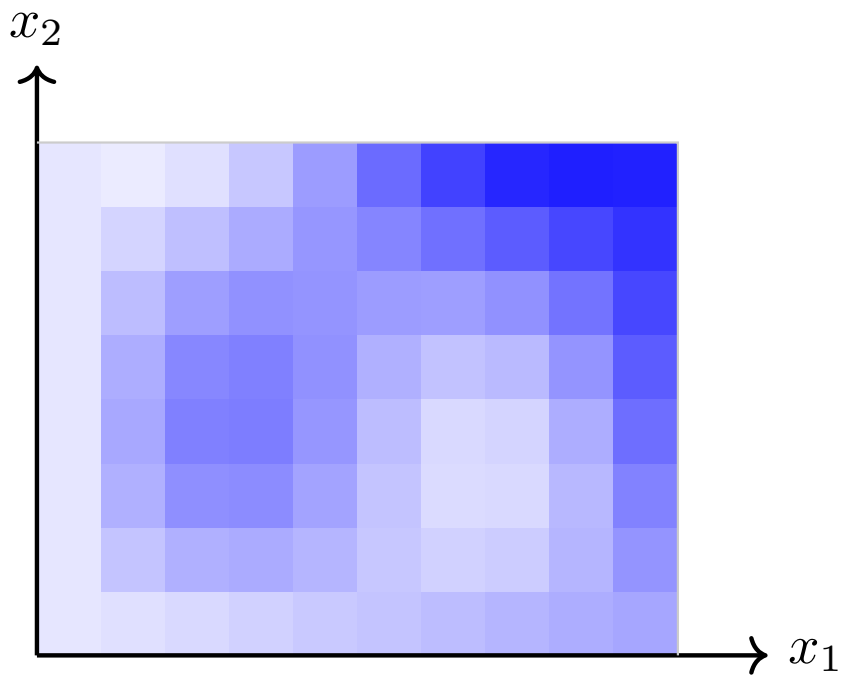
$$\hat{y}_{n+1} = \frac{1}{|\mathcal{T}|} \sum_{T \in \mathcal{T}} \hat{y}_T(\boldsymbol{x}_{n+1})$$

Tree-based methods

Single tree

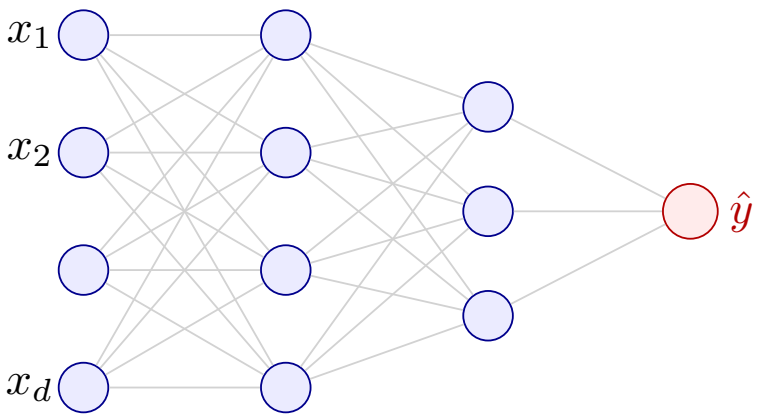


Tree ensemble



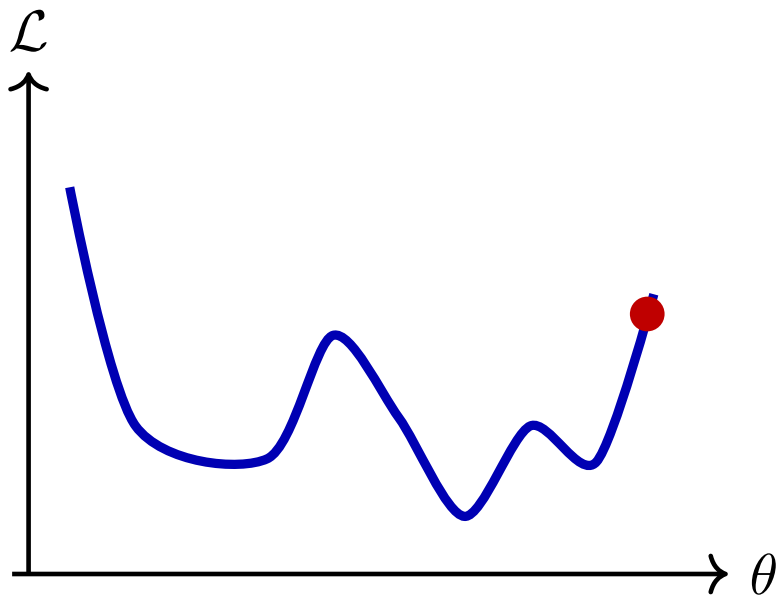
Deep learning

Neural network



$f_\theta(\boldsymbol{x})$

Train the weights

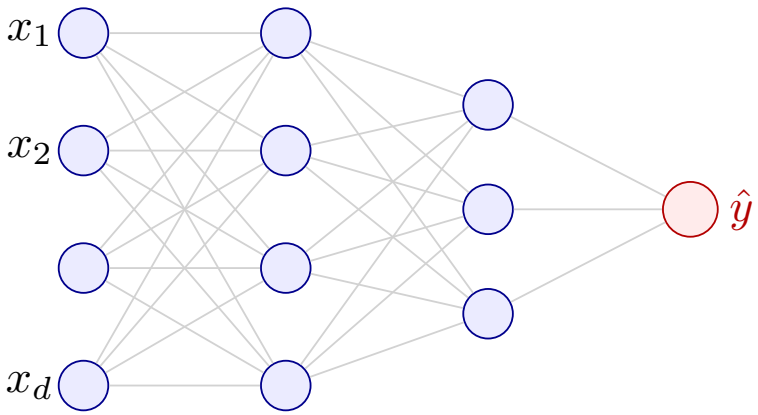


$$\min_\theta \frac{1}{n} \sum_{i=1}^{n} \ell(f_\theta(\boldsymbol{x}_i), y_i)$$

$$\theta \leftarrow \theta - \eta \nabla_\theta \mathcal{L}(\theta)$$

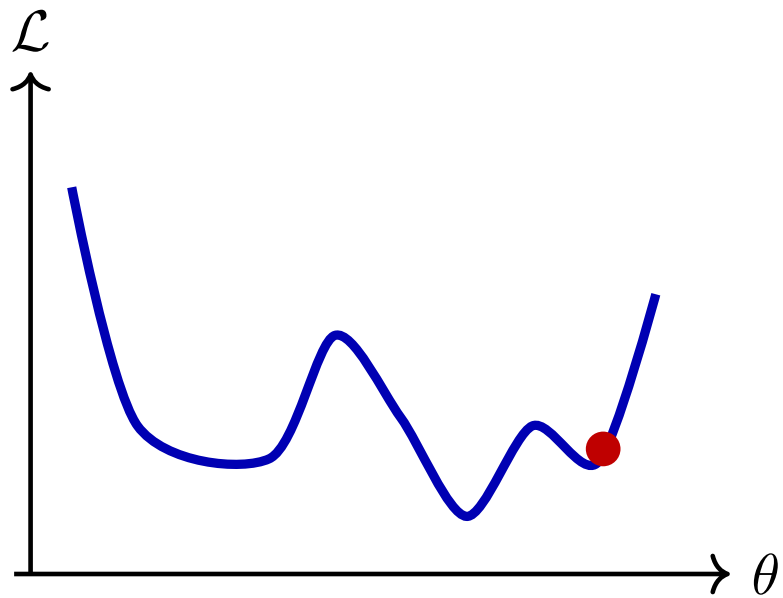
Deep learning

Neural network



$f_\theta(\boldsymbol{x})$

Train the weights

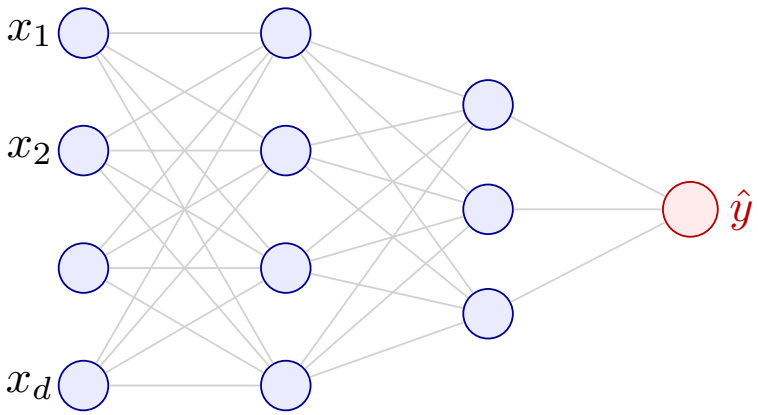


$$\min_\theta \frac{1}{n} \sum_{i=1}^{n} \ell(f_\theta(\boldsymbol{x}_i), y_i)$$

$$\theta \leftarrow \theta - \eta \nabla_\theta \mathcal{L}(\theta)$$

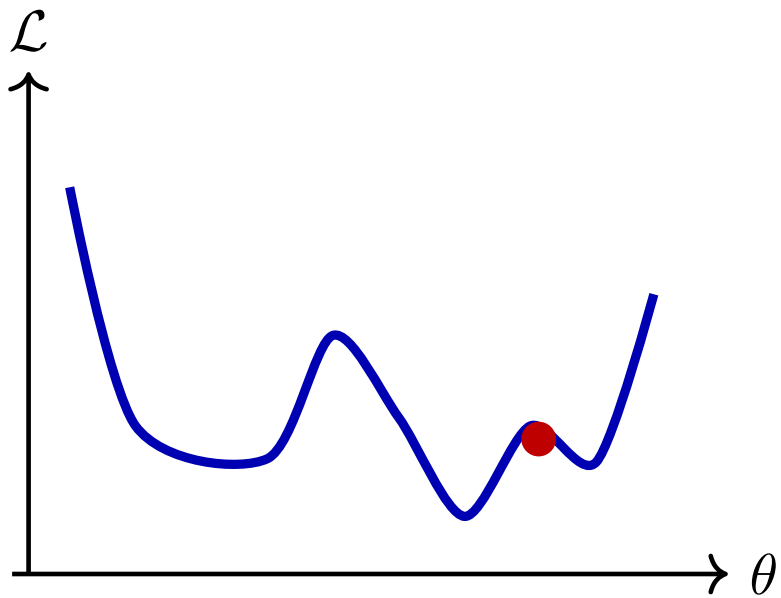
Deep learning

Neural network



$f_\theta(\boldsymbol{x})$

Train the weights

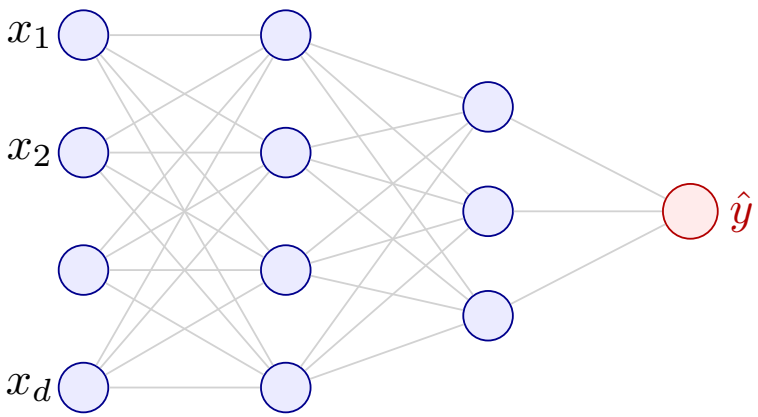


$$\min_\theta \frac{1}{n} \sum_{i=1}^{n} \ell(f_\theta(\boldsymbol{x}_i), y_i)$$

$$\theta \leftarrow \theta - \eta \nabla_\theta \mathcal{L}(\theta)$$

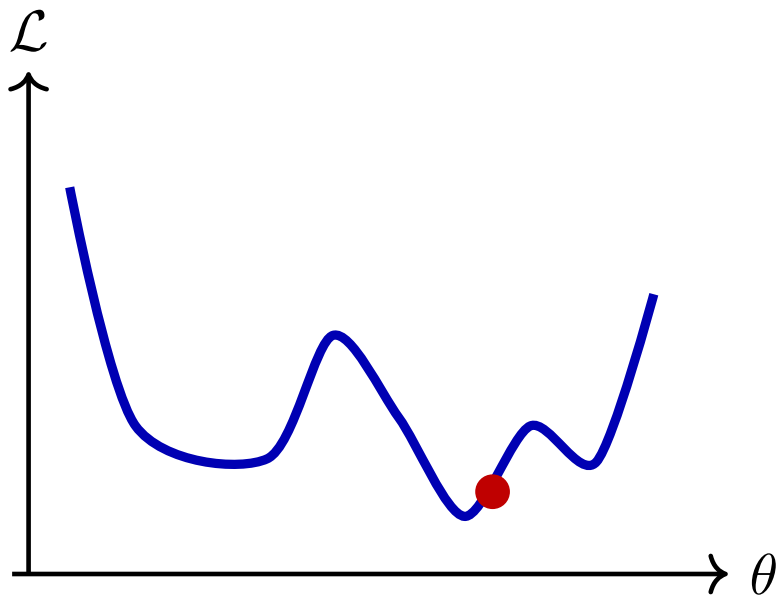
Deep learning

Neural network



$f_\theta(\boldsymbol{x})$

Train the weights

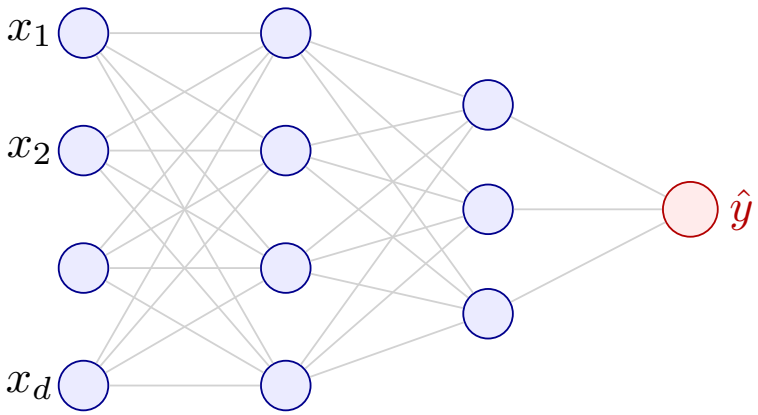


$$\min_\theta \frac{1}{n} \sum_{i=1}^{n} \ell(f_\theta(\boldsymbol{x}_i), y_i)$$

$$\theta \leftarrow \theta - \eta \nabla_\theta \mathcal{L}(\theta)$$

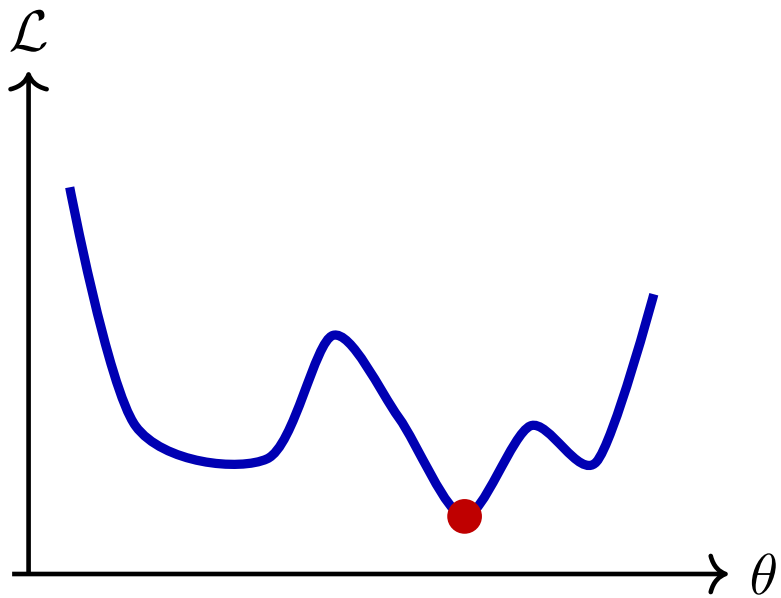
Deep learning

Neural network



$f_\theta(\boldsymbol{x})$

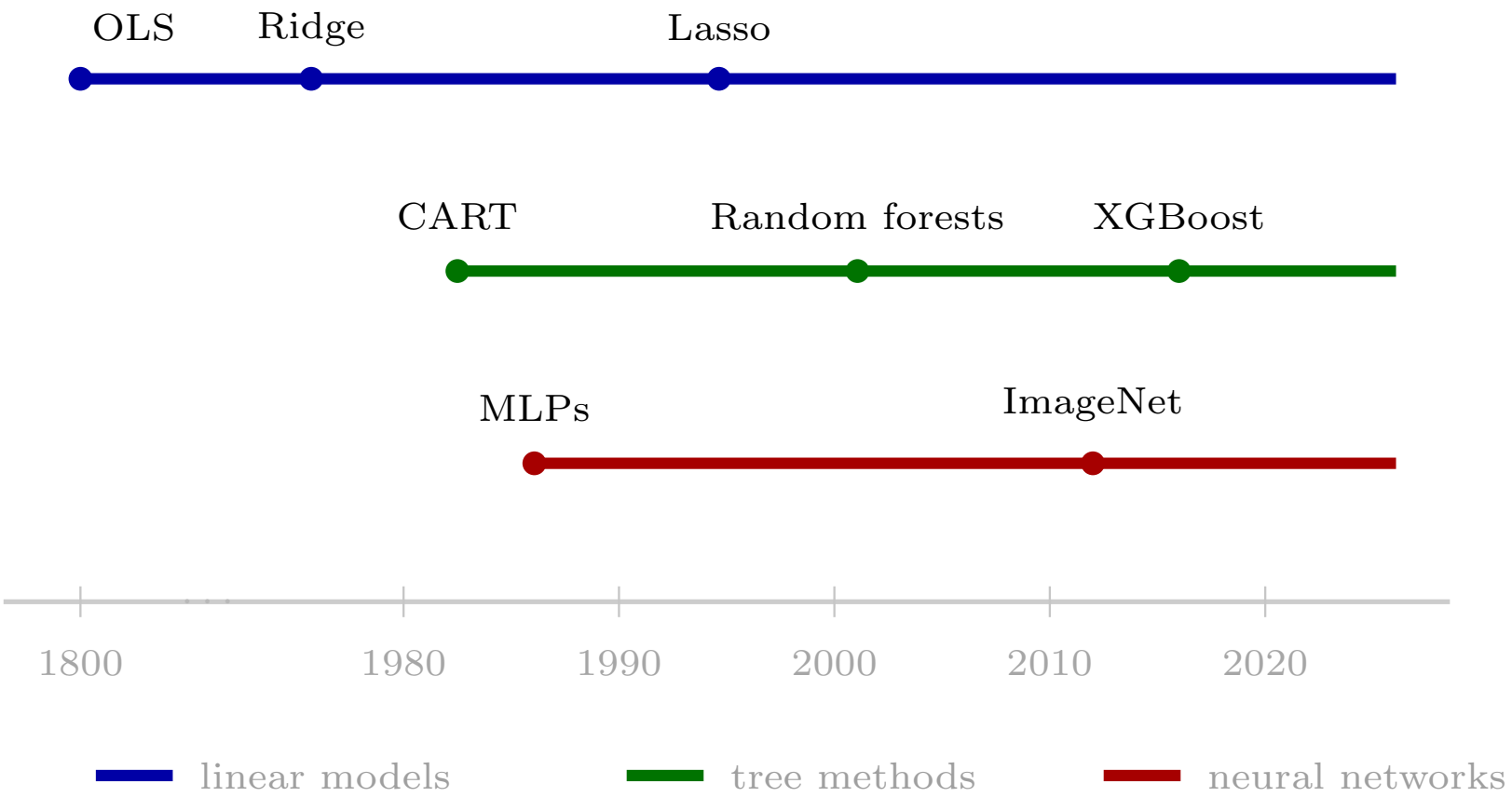
Train the weights



$$\min_\theta \frac{1}{n} \sum_{i=1}^{n} \ell(f_\theta(\boldsymbol{x}_i), y_i)$$

$$\theta \leftarrow \theta - \eta \nabla_\theta \mathcal{L}(\theta)$$

A brief history



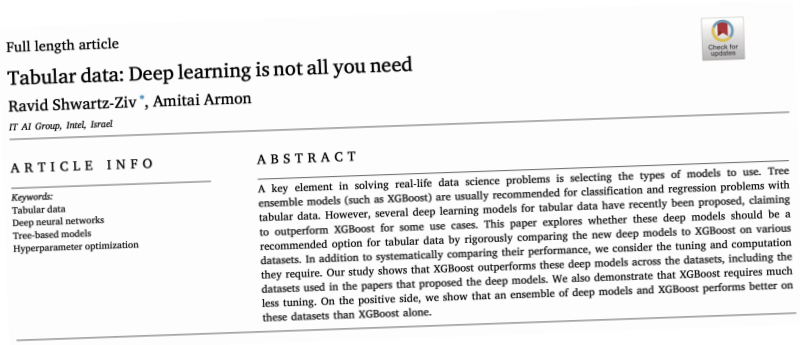
Bayesian analogues

Model class	**Bayesian analogue**	**Parameters**
Linear regression	Bayesian linear regression	$\boldsymbol{\beta}$, σ^2
Lasso	Horseshoe, Spike & slab	$\boldsymbol{\beta}$, σ^2, shrinkage
Tree ensembles	BART	trees, leaf values
Neural nets	Bayesian neural nets	network weights

State of the art

Despite success of neural nets in other areas (image, text),
tree-based methods remain state of the art for tabular data

State of the art



Full length article

Tabular data: Deep learning is not all you need

Ravid Shwartz-Ziv *, Amitai Armon

IT AI Group, Intel, Israel

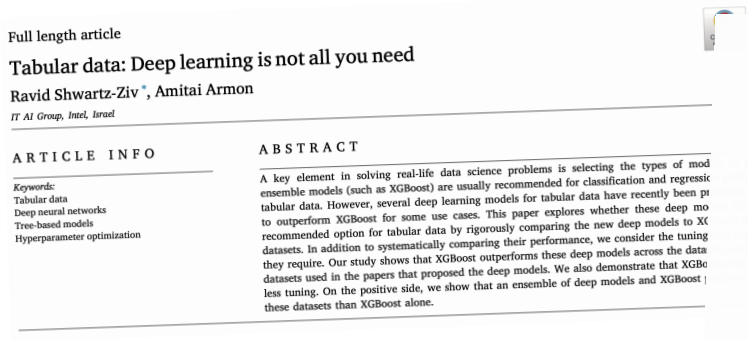
ARTICLE INFO

Keywords:
Tabular data
Deep neural networks
Tree-based models
Hyperparameter optimization

ABSTRACT

A key element in solving real-life data science problems is selecting the types of models to use. Tree ensemble models (such as XGBoost) are usually recommended for classification and regression problems with tabular data. However, several deep learning models for tabular data have recently been proposed, claiming to outperform XGBoost for some use cases. This paper explores whether these deep models should be a recommended option for tabular data by rigorously comparing the new deep models to XGBoost on various datasets. In addition to systematically comparing their performance, we consider the tuning and computation they require. Our study shows that XGBoost outperforms these deep models across the datasets, including the datasets used in the papers that proposed the deep models. We also demonstrate that XGBoost requires much less tuning. On the positive side, we show that an ensemble of deep models and XGBoost performs better on these datasets than XGBoost alone.

State of the art



Full length article

Tabular data: Deep learning is not all you need

Ravid Shwartz-Ziv *, Amitai Armon

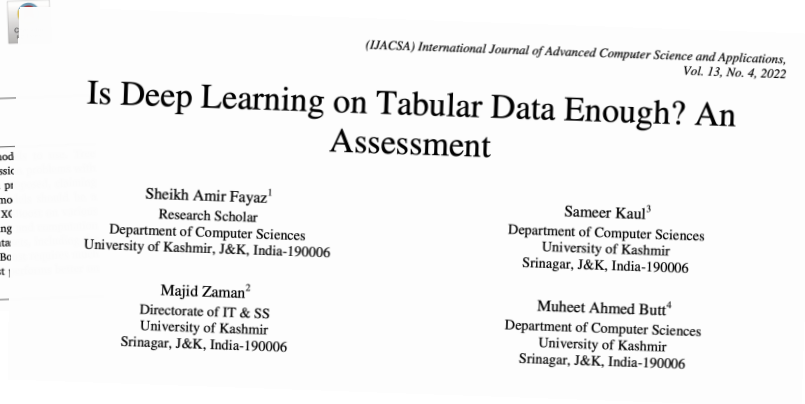
IT AI Group, Intel, Israel

ARTICLE INFO

Keywords:
Tabular data
Deep neural networks
Tree-based models
Hyperparameter optimization

ABSTRACT

A key element in solving real-life data science problems is selecting the types of mod
ensemble models (such as XGBoost) are usually recommended for classification and regressio
tabular data. However, several deep learning models for tabular data have recently been pr
to outperform XGBoost for some use cases. This paper explores whether these deep mo
recommended option for tabular data by rigorously comparing the new deep models to XG
datasets. In addition to systematically comparing their performance, we consider the tuning
they require. Our study shows that XGBoost outperforms these deep models across the data
datasets used in the papers that proposed the deep models. We also demonstrate that XGBo
less tuning. On the positive side, we show that an ensemble of deep models and XGBoost
these datasets than XGBoost alone.



(IJACSA) International Journal of Advanced Computer Science and Applications,
Vol. 13, No. 4, 2022

Is Deep Learning on Tabular Data Enough? An Assessment

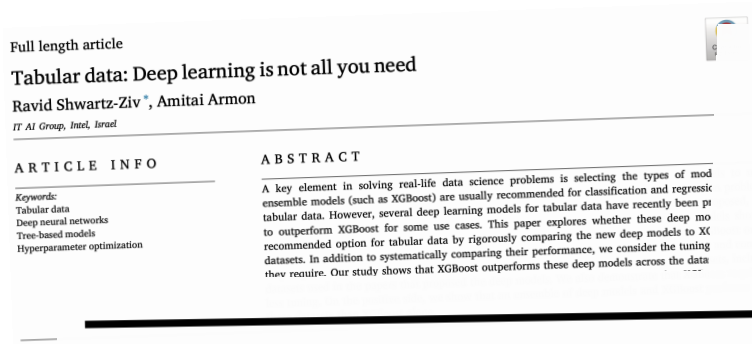
Sheikh Amir Fayaz[1]
Research Scholar
Department of Computer Sciences
University of Kashmir, J&K, India-190006

Sameer Kaul[3]
Department of Computer Sciences
University of Kashmir
Srinagar, J&K, India-190006

Majid Zaman[2]
Directorate of IT & SS
University of Kashmir
Srinagar, J&K, India-190006

Muheet Ahmed Butt[4]
Department of Computer Sciences
University of Kashmir
Srinagar, J&K, India-190006

State of the art



Full length article

Tabular data: Deep learning is not all you need

Ravid Shwartz-Ziv *, Amitai Armon

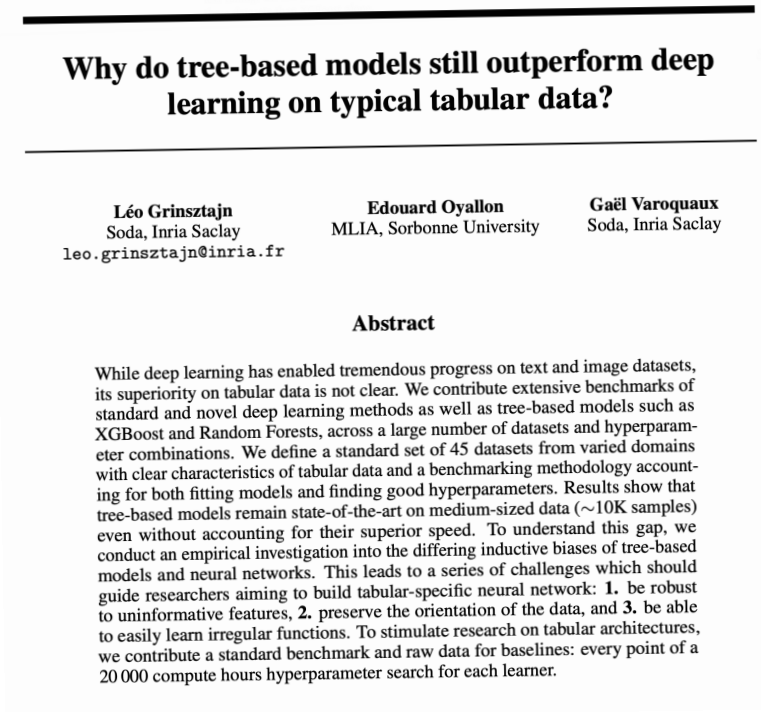
IT AI Group, Intel, Israel

ARTICLE INFO

Keywords:
Tabular data
Deep neural networks
Tree-based models
Hyperparameter optimization

ABSTRACT

A key element in solving real-life data science problems is selecting the types of mod
ensemble models (such as XGBoost) are usually recommended for classification and regressic
tabular data. However, several deep learning models for tabular data have recently been pr
to outperform XGBoost for some use cases. This paper explores whether these deep mo
recommended option for tabular data by rigorously comparing the new deep models to XG
datasets. In addition to systematically comparing their performance, we consider the tuning
they require. Our study shows that XGBoost outperforms these deep models across the data



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

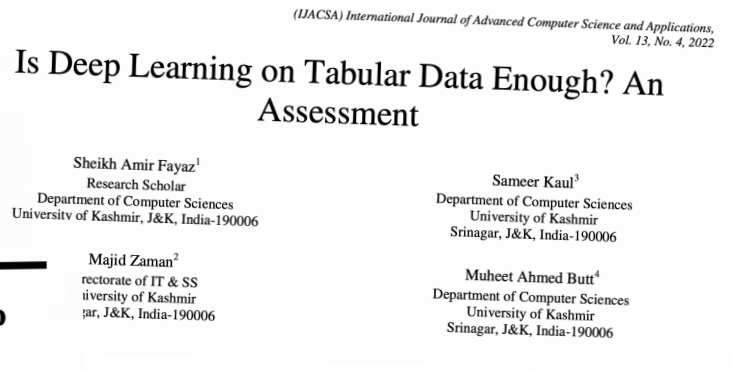
Léo Grinsztajn
Soda, Inria Saclay
leo.grinsztajn@inria.fr

Edouard Oyallon
MLIA, Sorbonne University

Gaël Varoquaux
Soda, Inria Saclay

Abstract

While deep learning has enabled tremendous progress on text and image datasets, its superiority on tabular data is not clear. We contribute extensive benchmarks of standard and novel deep learning methods as well as tree-based models such as XGBoost and Random Forests, across a large number of datasets and hyperparameter combinations. We define a standard set of 45 datasets from varied domains with clear characteristics of tabular data and a benchmarking methodology accounting for both fitting models and finding good hyperparameters. Results show that tree-based models remain state-of-the-art on medium-sized data (~10K samples) even without accounting for their superior speed. To understand this gap, we conduct an empirical investigation into the differing inductive biases of tree-based models and neural networks. This leads to a series of challenges which should guide researchers aiming to build tabular-specific neural network: **1.** be robust to uninformative features, **2.** preserve the orientation of the data, and **3.** be able to easily learn irregular functions. To stimulate research on tabular architectures, we contribute a standard benchmark and raw data for baselines: every point of a 20 000 compute hours hyperparameter search for each learner.



(IJACSA) International Journal of Advanced Computer Science and Applications,
Vol. 13, No. 4, 2022

Is Deep Learning on Tabular Data Enough? An Assessment

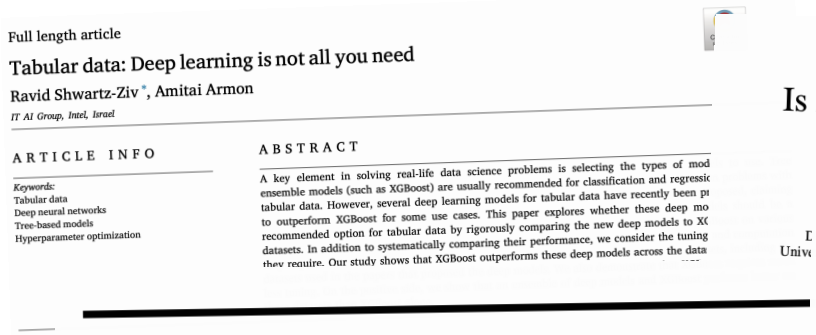
Sheikh Amir Fayaz[1]
Research Scholar
Department of Computer Sciences
University of Kashmir, J&K, India-190006

Majid Zaman[2]
rectorate of IT & SS
iversity of Kashmir
ar, J&K, India-190006

Sameer Kaul[3]
Department of Computer Sciences
University of Kashmir
Srinagar, J&K, India-190006

Muheet Ahmed Butt[4]
Department of Computer Sciences
University of Kashmir
Srinagar, J&K, India-190006

State of the art



Full length article

Tabular data: Deep learning is not all you need

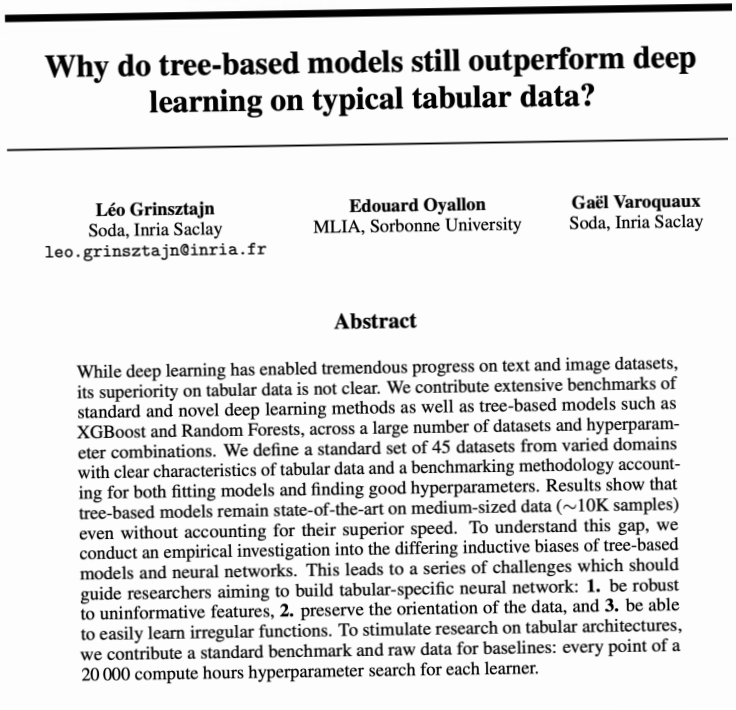
Ravid Shwartz-Ziv *, Amitai Armon
IT AI Group, Intel, Israel

ARTICLE INFO

Keywords:
Tabular data
Deep neural networks
Tree-based models
Hyperparameter optimization

ABSTRACT

A key element in solving real-life data science problems is selecting the types of mod
ensemble models (such as XGBoost) are usually recommended for classification and regressic
tabular data. However, several deep learning models for tabular data have recently been pr
to outperform XGBoost for some use cases. This paper explores whether these deep mo
recommended option for tabular data by rigorously comparing the new deep models to XG
datasets. In addition to systematically comparing their performance, we consider the tuning
they require. Our study shows that XGBoost outperforms these deep models across the data



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

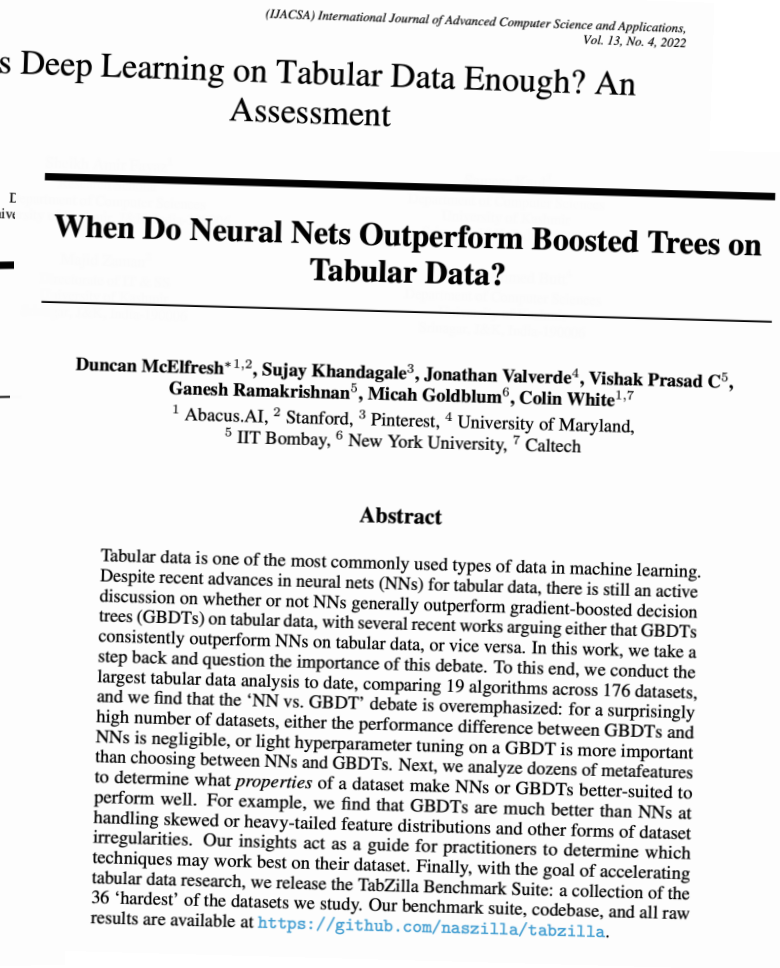
Léo Grinsztajn
Soda, Inria Saclay
leo.grinsztajn@inria.fr

Edouard Oyallon
MLIA, Sorbonne University

Gaël Varoquaux
Soda, Inria Saclay

Abstract

While deep learning has enabled tremendous progress on text and image datasets, its superiority on tabular data is not clear. We contribute extensive benchmarks of standard and novel deep learning methods as well as tree-based models such as XGBoost and Random Forests, across a large number of datasets and hyperparameter combinations. We define a standard set of 45 datasets from varied domains with clear characteristics of tabular data and a benchmarking methodology accounting for both fitting models and finding good hyperparameters. Results show that tree-based models remain state-of-the-art on medium-sized data (∼10K samples) even without accounting for their superior speed. To understand this gap, we conduct an empirical investigation into the differing inductive biases of tree-based models and neural networks. This leads to a series of challenges which should guide researchers aiming to build tabular-specific neural network: **1.** be robust to uninformative features, **2.** preserve the orientation of the data, and **3.** be able to easily learn irregular functions. To stimulate research on tabular architectures, we contribute a standard benchmark and raw data for baselines: every point of a 20 000 compute hours hyperparameter search for each learner.



(IJACSA) International Journal of Advanced Computer Science and Applications,
Vol. 13, No. 4, 2022

Is Deep Learning on Tabular Data Enough? An Assessment

When Do Neural Nets Outperform Boosted Trees on Tabular Data?

Duncan McElfresh[*1,2], **Sujay Khandagale**[3], **Jonathan Valverde**[4], **Vishak Prasad C**[5], **Ganesh Ramakrishnan**[5], **Micah Goldblum**[6], **Colin White**[1,7]
[1] Abacus.AI, [2] Stanford, [3] Pinterest, [4] University of Maryland,
[5] IIT Bombay, [6] New York University, [7] Caltech

Abstract

Tabular data is one of the most commonly used types of data in machine learning. Despite recent advances in neural nets (NNs) for tabular data, there is still an active discussion on whether or not NNs generally outperform gradient-boosted decision trees (GBDTs) on tabular data, with several recent works arguing either that GBDTs consistently outperform NNs on tabular data, or vice versa. In this work, we take a step back and question the importance of this debate. To this end, we conduct the largest tabular data analysis to date, comparing 19 algorithms across 176 datasets, and we find that the 'NN vs. GBDT' debate is overemphasized: for a surprisingly high number of datasets, either the performance difference between GBDTs and NNs is negligible, or light hyperparameter tuning on a GBDT is more important than choosing between NNs and GBDTs. Next, we analyze dozens of metafeatures to determine what *properties* of a dataset make NNs or GBDTs better-suited to perform well. For example, we find that GBDTs are much better than NNs at handling skewed or heavy-tailed feature distributions and other forms of dataset irregularities. Our insights act as a guide for practitioners to determine which techniques may work best on their dataset. Finally, with the goal of accelerating tabular data research, we release the TabZilla Benchmark Suite: a collection of the 36 'hardest' of the datasets we study. Our benchmark suite, codebase, and all raw results are available at https://github.com/naszilla/tabzilla.

State of the art

Full length article

Tabular data: Deep learning is not all you need

Ravid Shwartz-Ziv *, Amitai Armon

IT AI Group, Intel, Israel

ARTICLE INFO

Keywords:
Tabular data
Deep neural networks
Tree-based models
Hyperparameter optimization

ABSTRACT

A key element in solving real-life data science problems is selecting the types of mod
ensemble models (such as XGBoost) are usually recommended for classification and regressic
tabular data. However, several deep learning models for tabular data have recently been pr
to outperform XGBoost for some use cases. This paper explores whether these deep mo
recommended option for tabular data by rigorously comparing the new deep models to XG
datasets. In addition to systematically comparing their performance, we consider the tuning
they require. Our study shows that XGBoost outperforms these deep models across the data

(IJACSA) International Journal of Advanced Computer Science and Applications,
Vol. 13, No. 4, 2022

Is Deep Learning on Tabular Data Enough? An Assessment

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

When Do Neural Nets Outperform Boosted Trees on Tabular Data?

Duncan McElfresh*[1,2], Sujay Khandagale[3], Jonathan Valverde[4], Vishak Prasad C[5], Ganesh Ramakrishnan[5], Micah Goldblum[6], Colin White[1,7]

[1] Abacus.AI, [2] Stanford, [3] Pinterest, [4] University of Maryland,
[5] IIT Bombay, [6] New York University, [7] Caltech

RESEARCH ARTICLE

Confirming the statistically significant superiority of tree-based machine learning algorithms over their counterparts for tabular data

Shahadat Uddin *, Haohui Lu

School of Project Management, Faculty of Engineering, The University of Sydney, Forest Lodge, NSW, Australia

Abstract

While deep learning has enabled tremen
its superiority on tabular data is not clea
standard and novel deep learning meth
XGBoost and Random Forests, across a
eter combinations. We define a standar
with clear characteristics of tabular data
ing for both fitting models and finding
tree-based models remain state-of-the-a
even without accounting for their sup
conduct an empirical investigation into
models and neural networks. This lea
guide researchers aiming to build tabu
to uninformative features, **2.** preserve the orientation of the data, and **3.** be able
to easily learn irregular functions. To stimulate research on tabular architectures,
we contribute a standard benchmark and raw data for baselines: every point of a
20 000 compute hours hyperparameter search for each learner.

types of data in machine learning.
tabular data, there is still an active
erform gradient-boosted decision
works arguing either that GBDTs
vice versa. In this work, we take a
bate. To this end, we conduct the
19 algorithms across 176 datasets,
veremphasized: for a surprisingly
difference between GBDTs and
ng on a GBDT is more important
ve analyze dozens of metafeatures
NNs or GBDTs better-suited to
Ts are much better than NNs at
utions and other forms of dataset
practitioners to determine which
ally, with the goal of accelerating
tabular data research, we release the TabZilla Benchmark Suite: a collection of the
36 'hardest' of the datasets we study. Our benchmark suite, codebase, and all raw
results are available at https://github.com/naszilla/tabzilla.

State of the art

Why do trees still win on tabular data?

- Neural nets require large sample sizes (often $> 10{,}000$)
- Heterogeneity between tabular datasets
- Heterogeneity within tabular datasets (categorical, numeric, etc.)
- Trees more robust to, e.g., uniformative features, missing data
- Gap persists after extensive hyperparameter search

Outline

- Background
- **TabPFN**
- Results

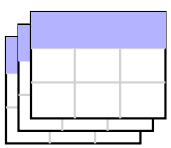
TabPFN: A tabular foundation model

Classical modeling: Fit new model for each new dataset

Foundation model: Pre-train on a ton of different data, then deploy

TabPFN: A tabular foundation model

text → **LLM** → prompt → **response**



tabular data → **TabPFN** → D, $\boldsymbol{x}_{n+1}$ → $\hat{y}_{n+1}$

$D = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^{n}$

Posterior predictive distribution

TabPFN targets the **posterior predictive distribution**

$$p(y \mid \boldsymbol{x}, D)$$

where $D = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^{n}$ and $(\boldsymbol{x}, y)$ a new point from same population

Posterior predictive distribution

TabPFN targets the **posterior predictive distribution**

$$p(y \mid \boldsymbol{x}, D)$$

where $D = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^{n}$ and $(\boldsymbol{x}, y)$ a new point from same population

If we know $p(\cdot \mid \boldsymbol{x}, D)$, then prediction done via, e.g.,

$$\hat{y} = \int y p(y \mid \boldsymbol{x}, D)\, dy.$$

Posterior predictive distribution

TabPFN targets the **posterior predictive distribution**

$$p(y \mid \boldsymbol{x}, D)$$

where $D = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^{n}$ and $(\boldsymbol{x}, y)$ a new point from same population

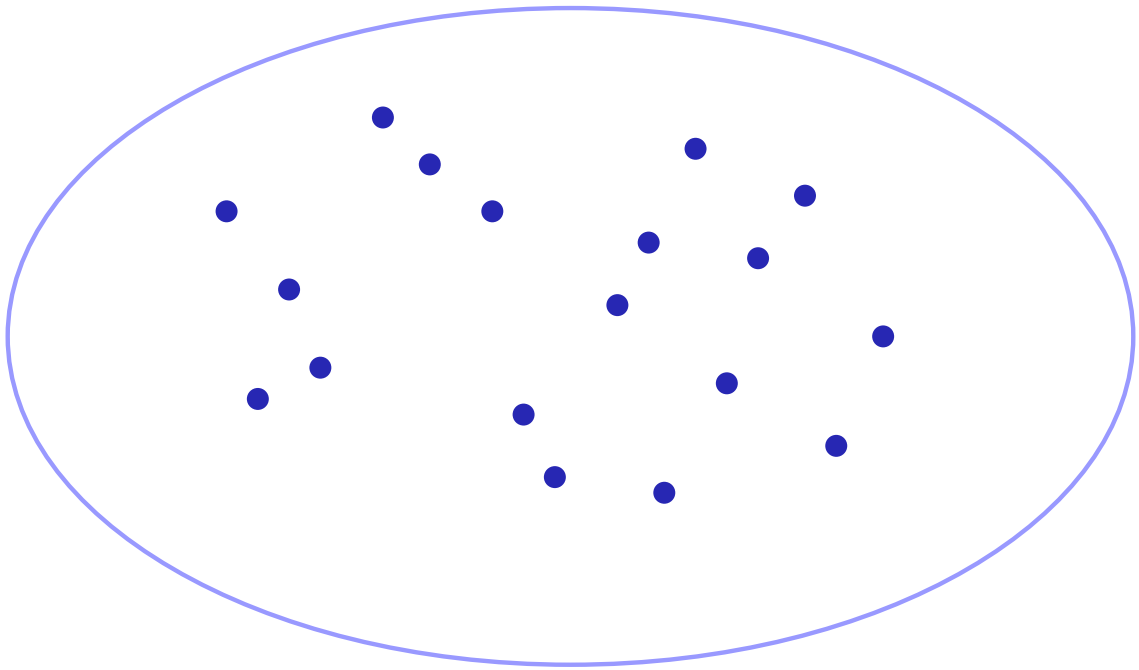
Most methods treat D as fixed and learn the map

$$\boldsymbol{x} \;\mapsto\; p(y \mid \boldsymbol{x}, D)$$

TabPFN learns the map

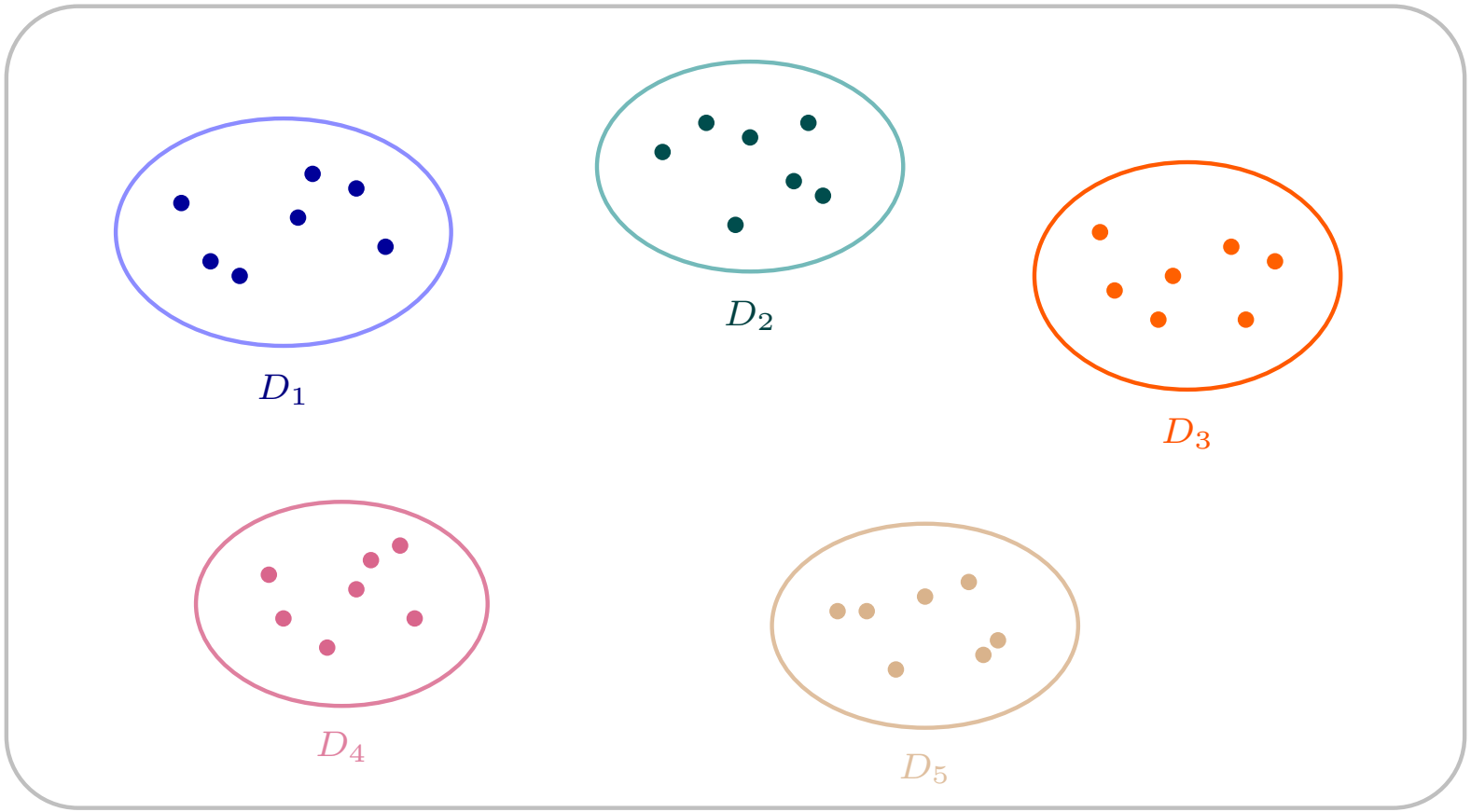
$$(\boldsymbol{x},\, D) \;\mapsto\; p(y \mid \boldsymbol{x}, D)$$

Single population, $(\boldsymbol{x}_i, y_i) \sim p(\boldsymbol{x}, y)$



$$D = \{(\boldsymbol{x}_i,\, y_i)\}_{i=1}^{n}$$

Space of all datasets, $D \sim \Pi(D)$



Traditional Bayesian approach

$$p(y \mid \boldsymbol{x}, D) = \int p(y \mid \boldsymbol{x}, \theta) p(\theta \mid D)\, d\theta$$

where

- $\theta \in \mathbb{R}^d$, e.g., $\theta = (\beta, \sigma^2)$
- $p(y \mid \boldsymbol{x}, \theta)$ the *conditional likelihood*, e.g., $\mathcal{N}\left(\beta^\top \boldsymbol{x}, \sigma^2\right)$
- $p(\theta \mid D)$ the *posterior*, e.g., $p(\beta, \sigma^2 \mid D)$

Traditional Bayesian approach

$$p(y \mid \boldsymbol{x}, D) = \int p(y \mid \boldsymbol{x}, \theta) p(\theta \mid D)\, d\theta$$

where

- $\theta \in \mathbb{R}^d$, e.g., $\theta = (\beta, \sigma^2)$
- $p(y \mid \boldsymbol{x}, \theta)$ the *conditional likelihood*, e.g., $\mathcal{N}\left(\beta^\top \boldsymbol{x}, \sigma^2\right)$
- $p(\theta \mid D)$ the *posterior*, e.g., $p(\beta, \sigma^2 \mid D)$

To predict using PPD,

- Draw $\theta^{(i)} \sim p(\theta \mid D)$
- Draw $y^{(i)} \sim p(y \mid \boldsymbol{x}, \theta^{(i)})$
- Monte Carlo estimate: $\hat{y} \approx \frac{1}{S} \sum_{i=1}^{S} y^{(i)}$

Traditional Bayesian approach

$$p(y \mid \boldsymbol{x}, D) = \int p(y \mid \boldsymbol{x}, \theta) p(\theta \mid D)\, d\theta$$

- D is fixed throughout
- Sampling from $p(\theta \mid D)$ can be very challenging/expensive/slow

TabPFN approach

$$p(y \mid \boldsymbol{x}, D) = \int p(y \mid \boldsymbol{x}, \theta)\, p(\theta \mid D)\, d\theta = \int \theta(y \mid \boldsymbol{x})\, p(\theta \mid D)\, d\theta$$

where

- $\theta \colon \mathcal{X} \to \mathcal{P}(\mathcal{Y})$ a *conditional density*, i.e., $p(y \mid \boldsymbol{x}, \theta) = \theta(y \mid \boldsymbol{x})$
- $p(\theta \mid D)$ the posterior over an *infinite-dimensional* function space
- The integral is intractable

TabPFN approach

$$p(y \mid \boldsymbol{x}, D) = \int p(y \mid \boldsymbol{x}, \theta)\, p(\theta \mid D)\, d\theta = \int \theta(y \mid \boldsymbol{x})\, p(\theta \mid D)\, d\theta$$

Rather than approximating this integral, train q_λ such that

$$q_\lambda(y \mid \boldsymbol{x},\, D) \;\approx\; p(y \mid \boldsymbol{x},\, D)$$

TabPFN approach

$$p(y \mid \boldsymbol{x}, D) = \int p(y \mid \boldsymbol{x}, \theta)\, p(\theta \mid D)\, d\theta = \int \theta(y \mid \boldsymbol{x})\, p(\theta \mid D)\, d\theta$$

Rather than approximating this integral, train q_λ such that

$$q_\lambda(y \mid \boldsymbol{x},\, D) \;\approx\; p(y \mid \boldsymbol{x},\, D)$$

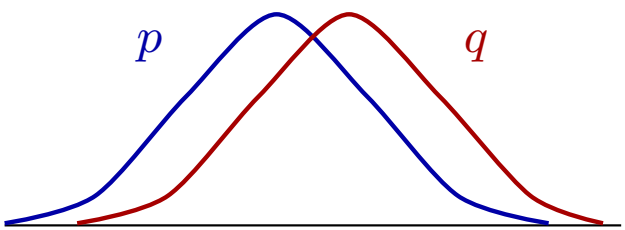
How?

KL-divergence

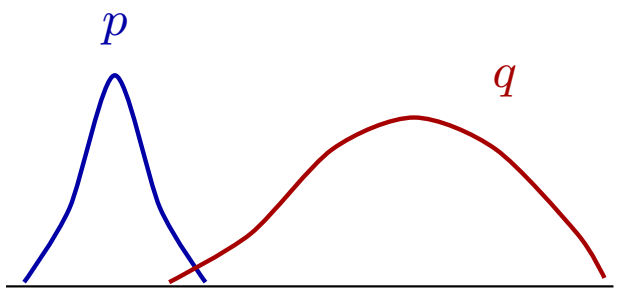
The *KL-divergence* between probability densities p and q is

$$\mathrm{KL}(p \,\|\, q) \;=\; \int p(y) \,\log \frac{p(y)}{q(y)}\, dy$$

KL-divergence is a pseudo-distance: non-negative and zero iff $p = q$.



small KL



large KL

Key insight

Prior Π on joint distributions $\pi(x, y)$ induces distribution on datasets:

- Draw $\pi \sim \Pi$
- Set $D = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^n$ where $(\boldsymbol{x}_i, y_i) \sim \pi$

$\implies$ Think of Π as "prior on space of datasets"

Key insight

Prior Π on joint distributions $\pi(x, y)$ induces distribution on datasets:

- Draw $\pi \sim \Pi$
- Set $D = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^n$ where $(\boldsymbol{x}_i, y_i) \sim \pi$

$$\implies \text{Think of } \Pi \text{ as "prior on space of datasets"}$$

The minimizer of the loss

$$\ell_\lambda = \mathbb{E}_{(\boldsymbol{x}, D) \sim \Pi}[-\log q_\lambda(y \mid \boldsymbol{x}, D)]$$

is the closest point to PPD in $\text{supp}(\Pi)$ wrt KL-divergence

Key insight

Prior Π on joint distributions $\pi(x, y)$ induces distribution on datasets:

- Draw $\pi \sim \Pi$
- Set $D = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^n$ where $(\boldsymbol{x}_i, y_i) \sim \pi$

$$\implies \text{Think of } \Pi \text{ as "prior on space of datasets"}$$

The minimizer of the loss

$$\ell_\lambda = \mathbb{E}_{(\boldsymbol{x}, D) \sim \Pi}[-\log q_\lambda(y \mid \boldsymbol{x}, D)]$$

is the closest point to PPD in $\text{supp}(\Pi)$ wrt KL-divergence

$$\implies \text{Train model (transformer) to find } \lambda_* = \arg\min_\lambda \ell_\lambda$$

Training

Algorithm 1: Training a PFN model by Fitting Prior-Data

Input : A prior distribution over datasets $p(\mathcal{D})$, from which samples can be drawn and the number of samples K to draw

Output : A model q_θ that will approximate the PPD

Initialize the neural network q_θ;

for $j \leftarrow 1$ **to** K **do**

  Sample $D \cup \{(x_i, y_i)\}_{i=1}^{m} \sim p(\mathcal{D})$;

  Compute stochastic loss approximation $\bar{\ell}_\theta = \sum_{i=1}^{m}(-\log q_\theta(y_i|x_i, D))$;

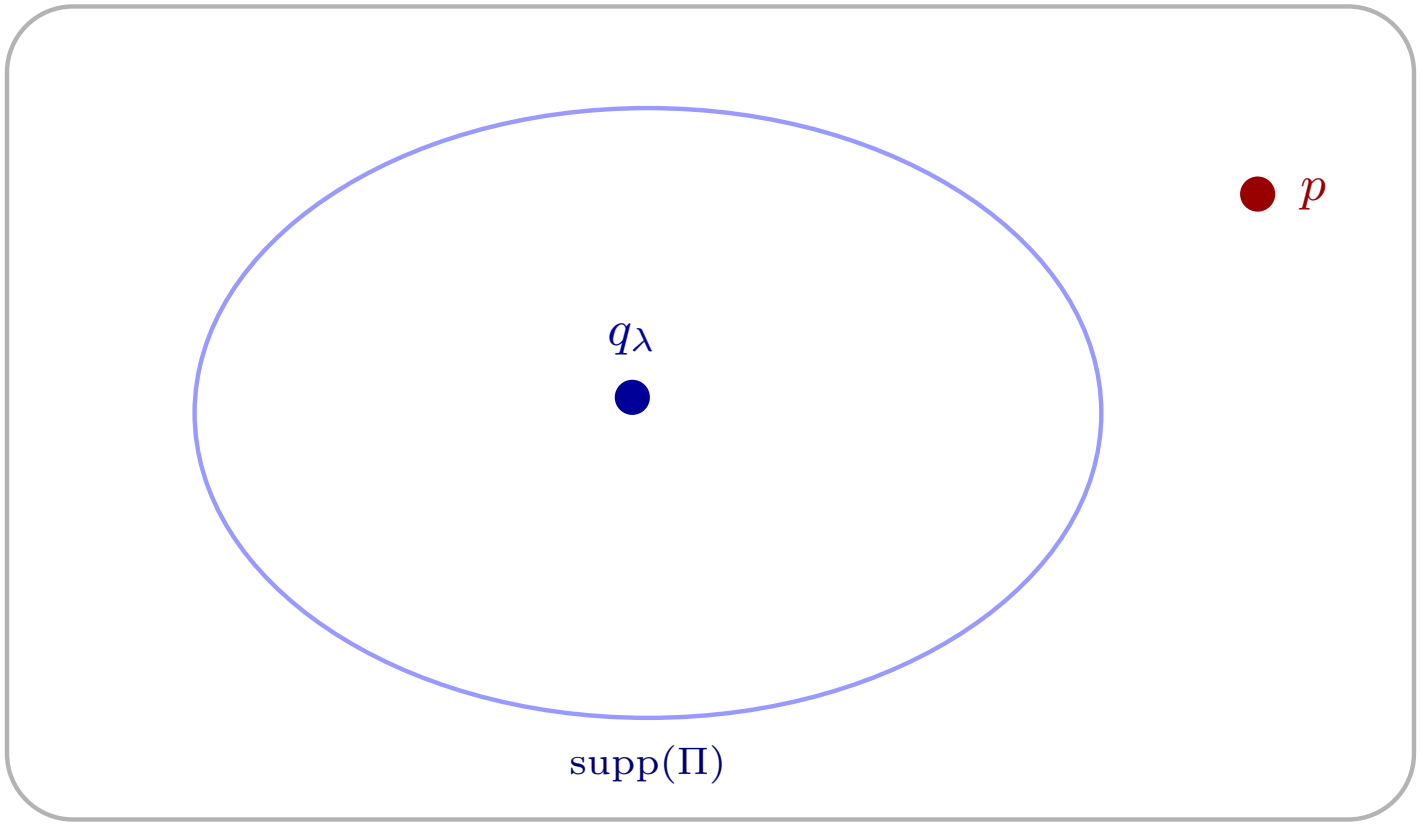
  Update parameters θ with stochastic gradient descent on $\nabla_\theta \bar{\ell}_\theta$;

end

Müller, Hollmann, Arango, Grabocka, Hutter. "*Transformers can do Bayesian inference*" (ICLR, 2022)

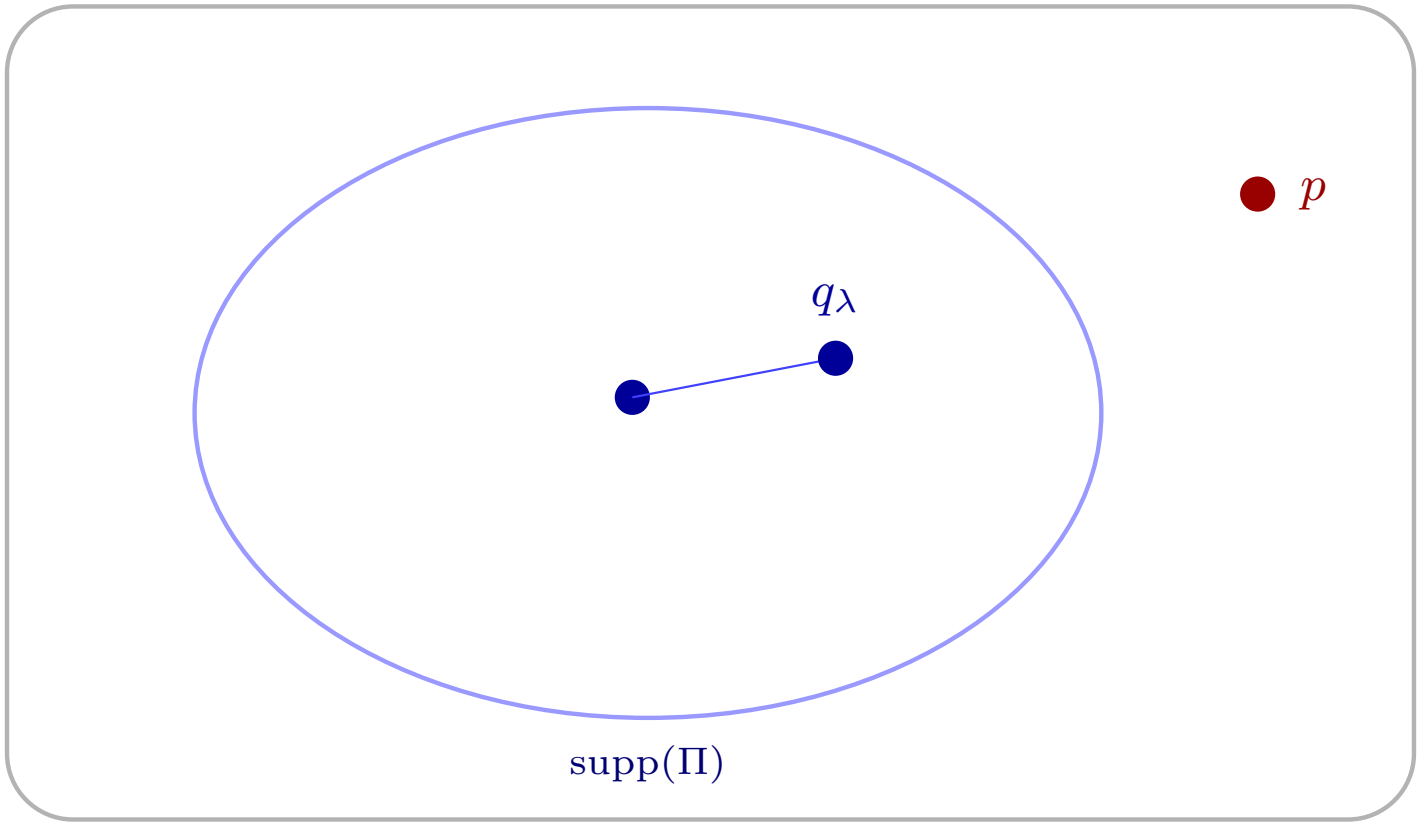
Training

Minimize "distance" between q_λ and p with respect to prior Π



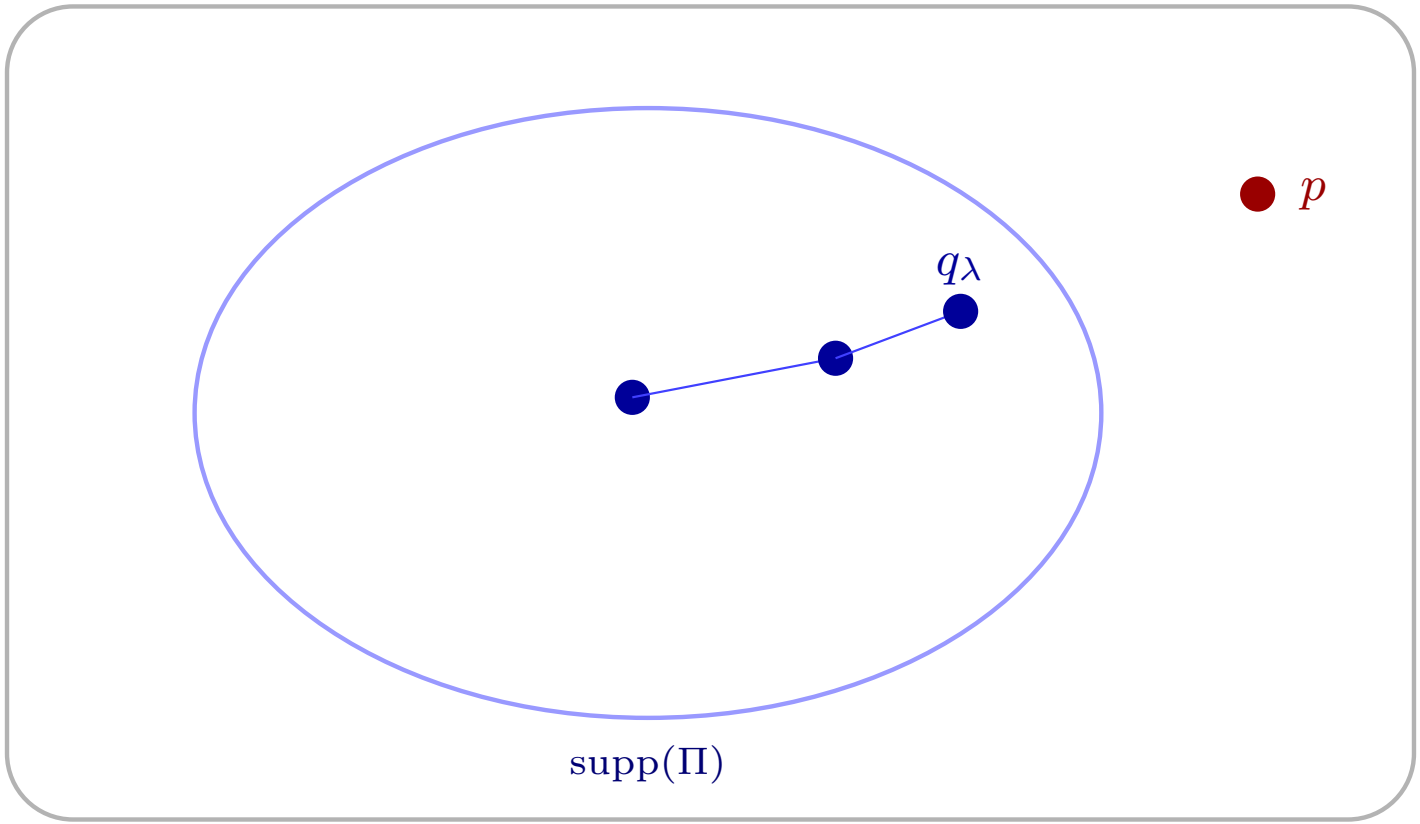
Training

Minimize "distance" between q_λ and p with respect to prior Π



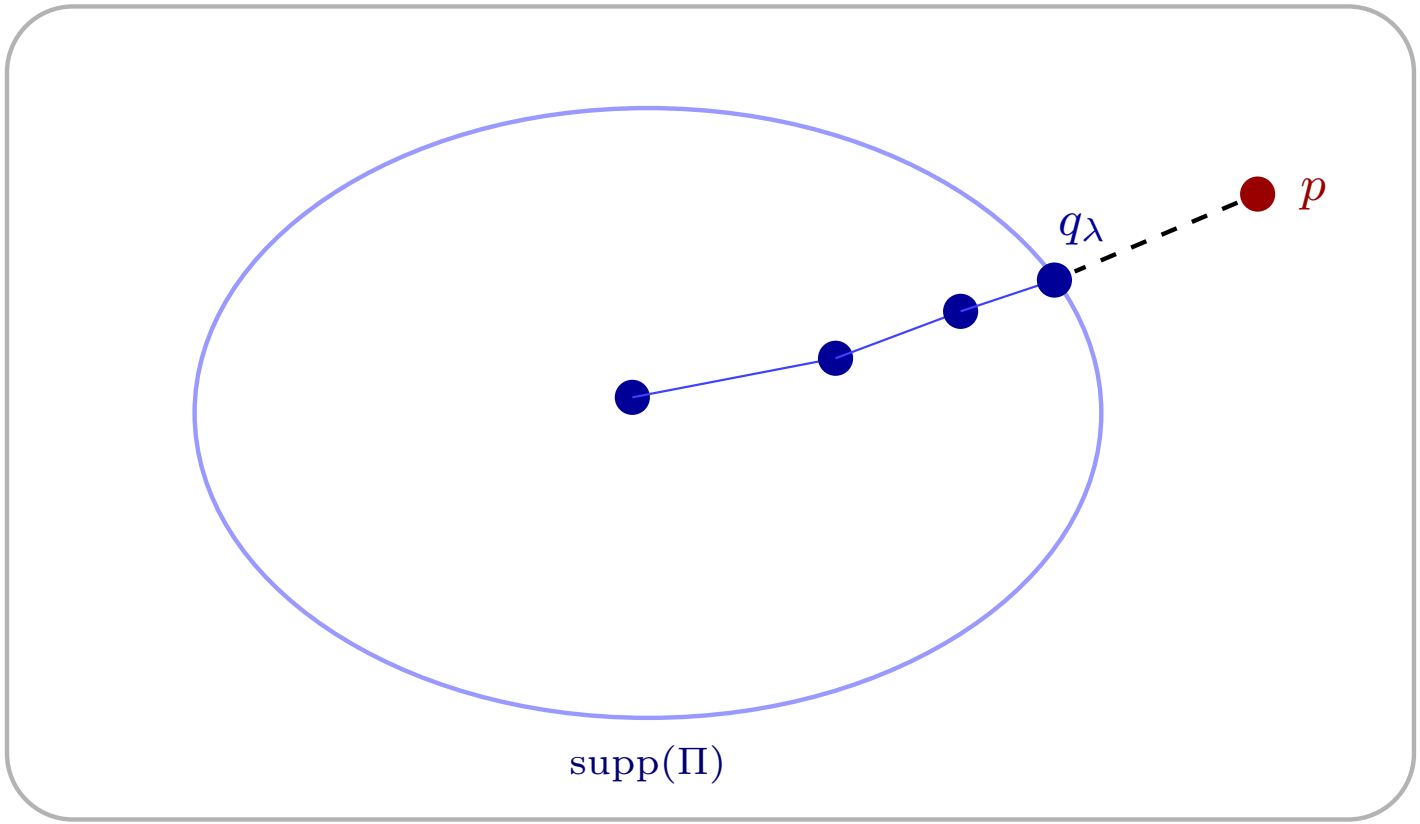
Training

Minimize "distance" between q_λ and p with respect to prior Π



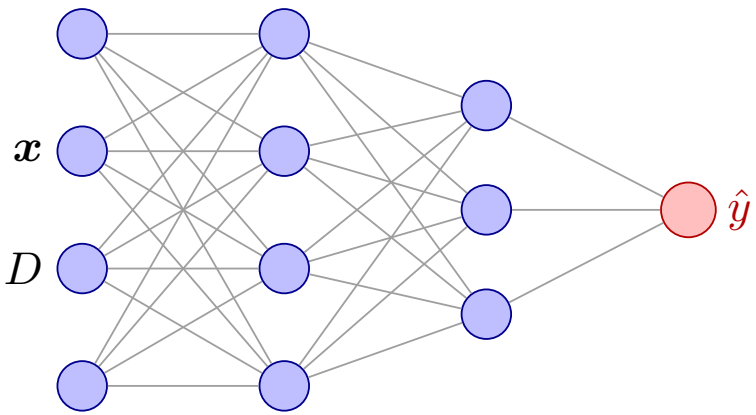
Training

Minimize "distance" between q_λ and p with respect to prior Π



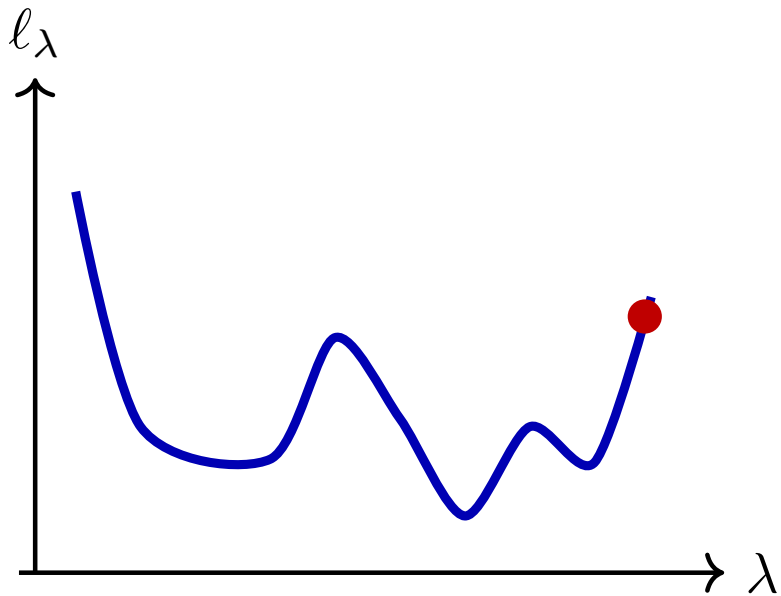
Training

Transformer



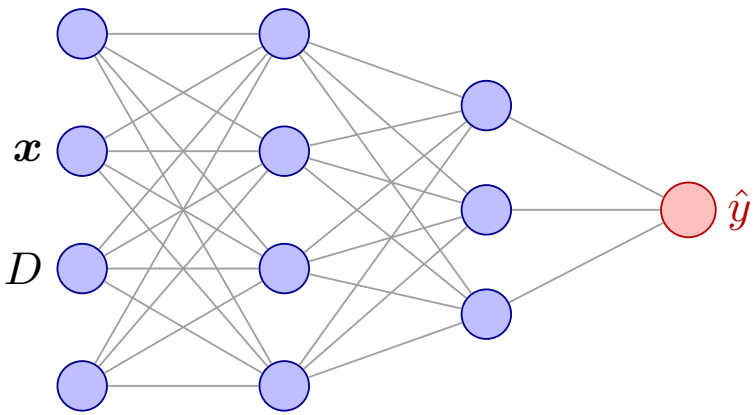
$q_\lambda(\boldsymbol{x}, D)$

Train the weights



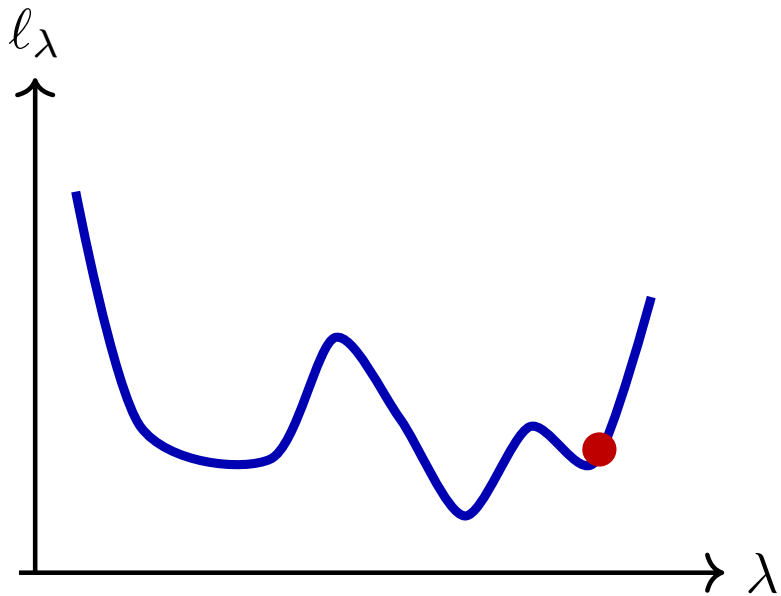
Training

Transformer



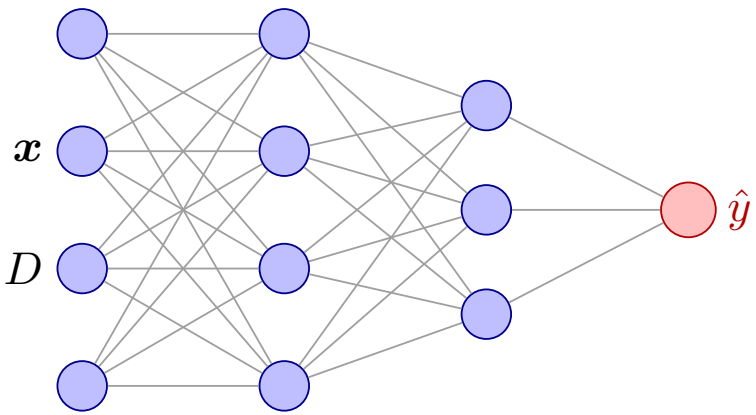
$q_\lambda(\boldsymbol{x}, D)$

Train the weights



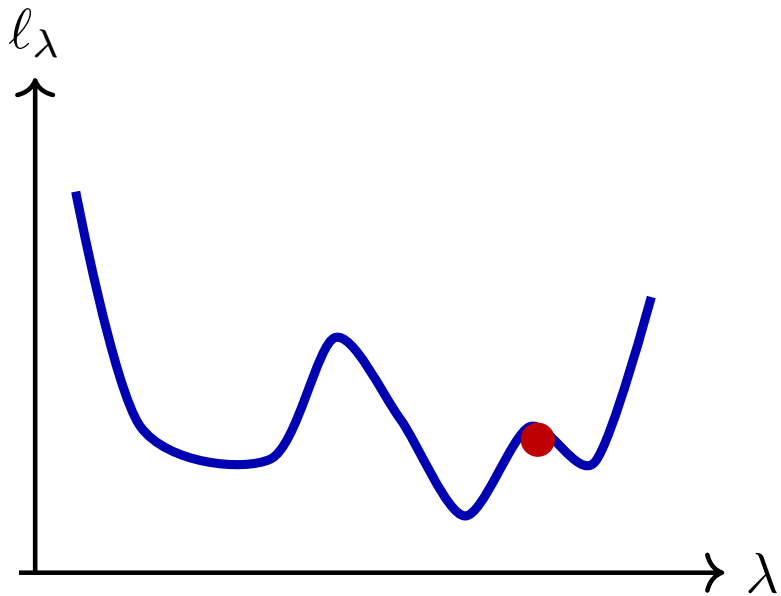
Training

Transformer



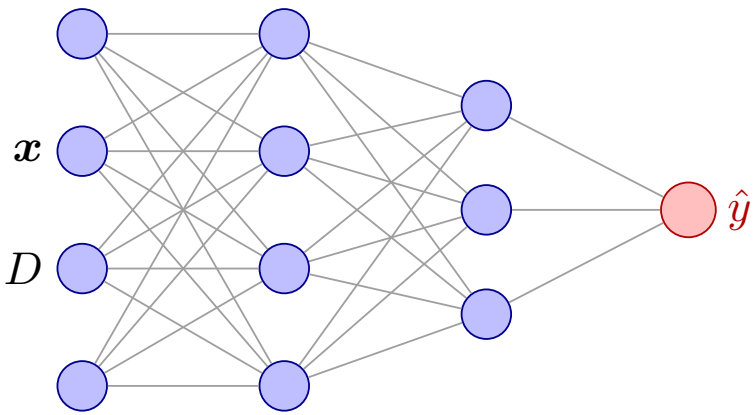
$q_\lambda(\boldsymbol{x}, D)$

Train the weights



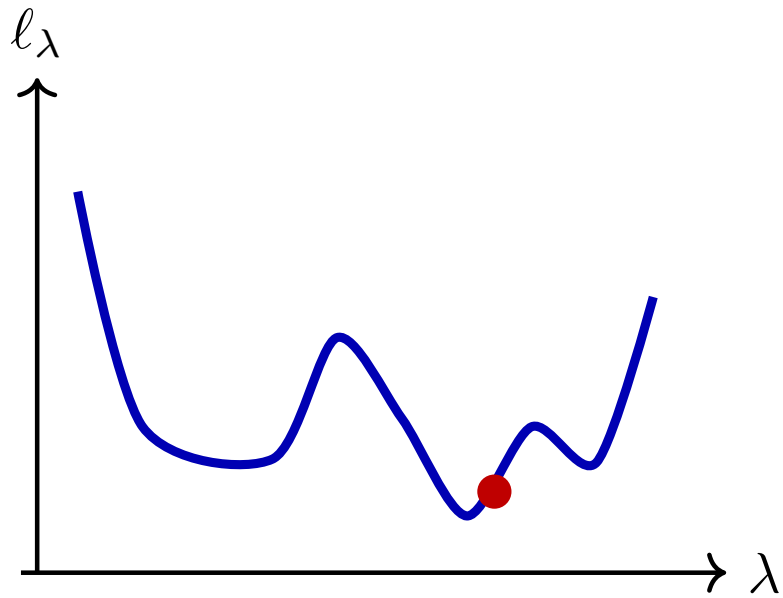
Training

Transformer



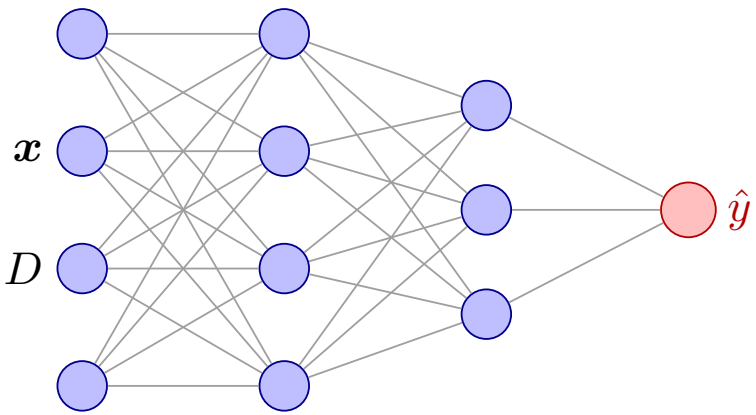
$q_\lambda(\boldsymbol{x}, D)$

Train the weights



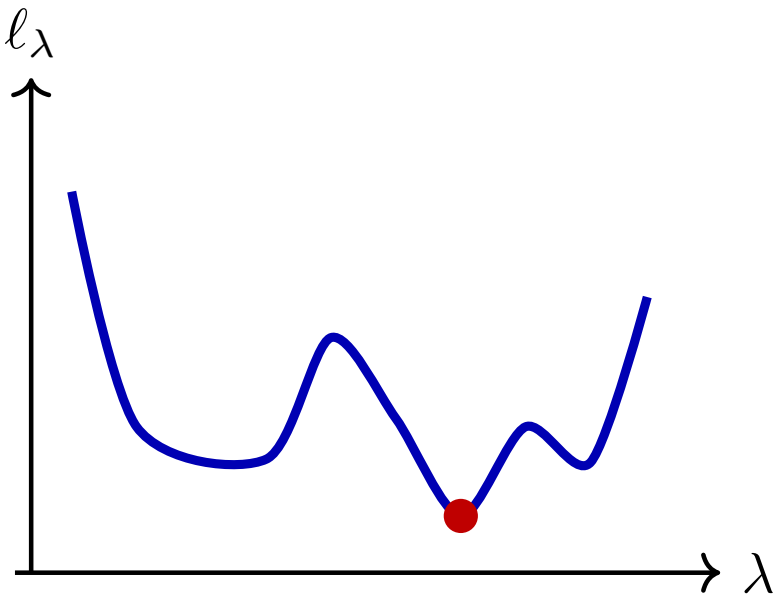
Training

Transformer



$q_\lambda(\boldsymbol{x}, D)$

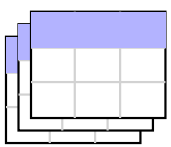
Train the weights



TabPFN: A summary

Specify prior Π over datasets and train a transformer by minimizing

$$\ell_\lambda = \mathbb{E}_{(\boldsymbol{x},D)\sim\Pi}[-\log q_\lambda(y \mid \boldsymbol{x}, D)]$$



tabular data

TabPFN

D, $\boldsymbol{x}_{n+1}$

$D = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^n$

$\hat{y}_{n+1}$

Outline

- Background
- TabPFN
- **Results**

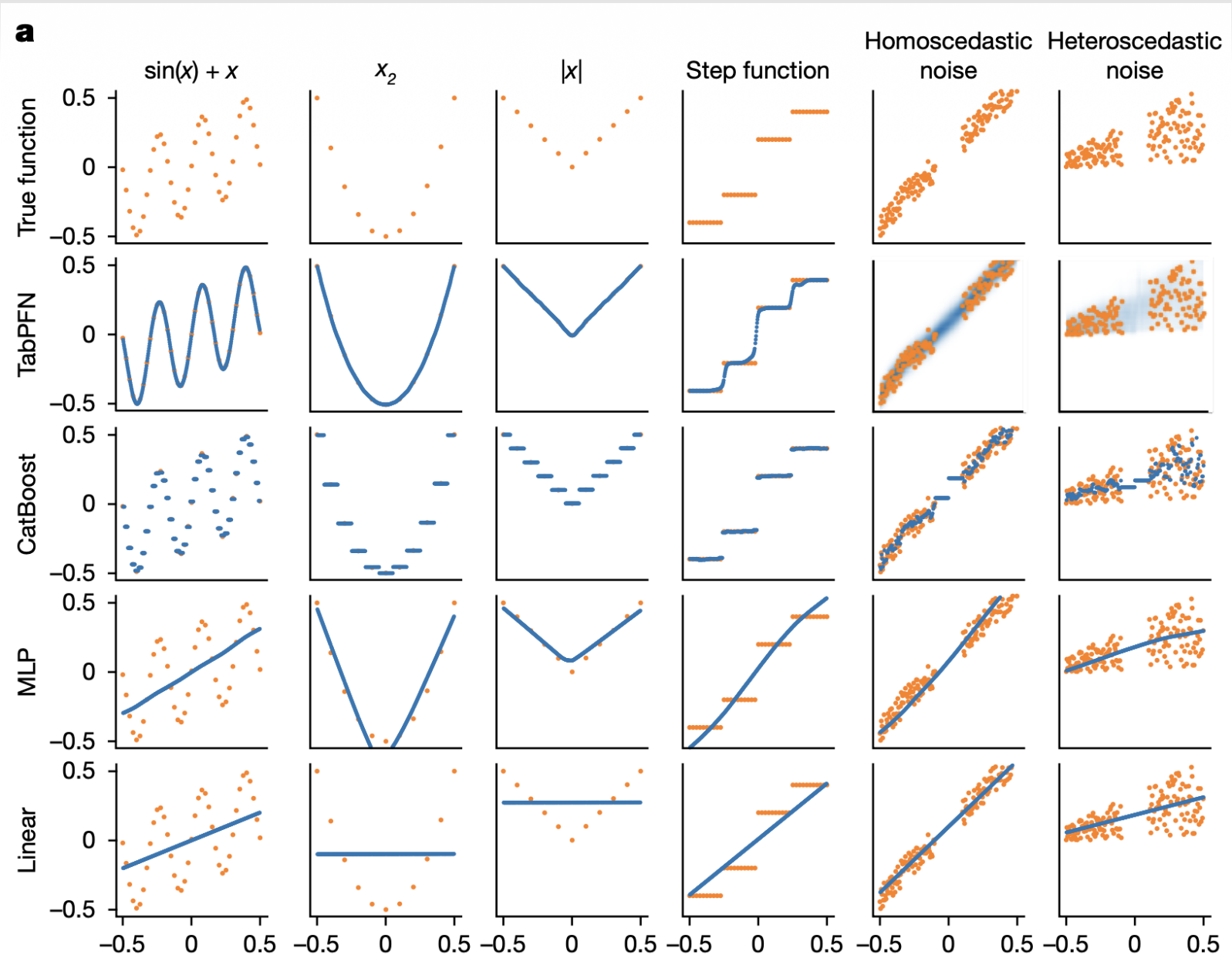
Results

Accurate predictions on small data with a tabular foundation model

Hollmann, Müller, Purucker, Krishnakumar, Körfer, Hoo, Schirrmeister, Hutter

Nature, 2025

The main advantage of TabPFN over all baselines is its inherent ability to model uncertainty at no extra cost. Whereas classical regression methods output a single real-valued prediction, TabPFN returns a target distribution, capturing the uncertainty of predictions. These uncertainty modelling abilities of TabPFN extend beyond simple distributions and can handle complex, multi-modal distributions. Figure 3b


a
sin(x) + x
x2
|x|
Step function
Homoscedastic noise
Heteroscedastic noise
True function
TabPFN
CatBoost
MLP
Linear
0.5
0
−0.5
−0.5
0
0.5

Quantitative analysis

We quantitatively evaluate TabPFN on two dataset collections: the AutoML Benchmark[36] and OpenML-CTR23[37]. These benchmarks comprise diverse real-world tabular datasets, curated for complexity, relevance and domain diversity. From these benchmarks, we use the 29 classification datasets and 28 regression datasets that have up to 10,000 samples, 500 features and 10 classes. We further evaluated additional benchmark suites from refs. 14,15, as well as five Kaggle competitions from the Tabular Playground Series.

We compared TabPFN against state-of-the-art baselines, including tree-based methods (random forest[38], XGBoost (XGB)[7], CatBoost[9], LightGBM[8]), linear models, support vector machines (SVMs)[39] and MLPs[34].

Evaluation metrics include ROC AUC (area under the receiver operating characteristic curve; One-vs-Rest) and accuracy for classification, and R^2 (coefficient of determination) and negative RMSE (root mean squared error) for regression. Scores were normalized per dataset, with 1.0 representing the best and 0.0 the worst performance with respect to all baselines.

For each dataset and method, we ran 10 repetitions with different random seeds and train–test splits (90% train, 10% test). We tuned hyperparameters using random search with five-fold cross-validation, with time budgets ranging from 30 s to 4 h. All methods were evaluated using eight CPU cores, with TabPFN additionally using a consumer-grade GPU (RTX 2080 Ti; other methods did not benefit from this, see Extended Data Fig. 2d). TabPFN was pre-trained once using eight NVIDIA RTX 2080 GPUs over 2 weeks, allowing for ICL on all new datasets in a single forward pass. These modest computational requirements make similar research accessible to academic labs. For details, refer to the section ‘Detailed evaluation protocol’.

Extended Data Table 3 | List of test datasets used for primary evaluation of classification tasks

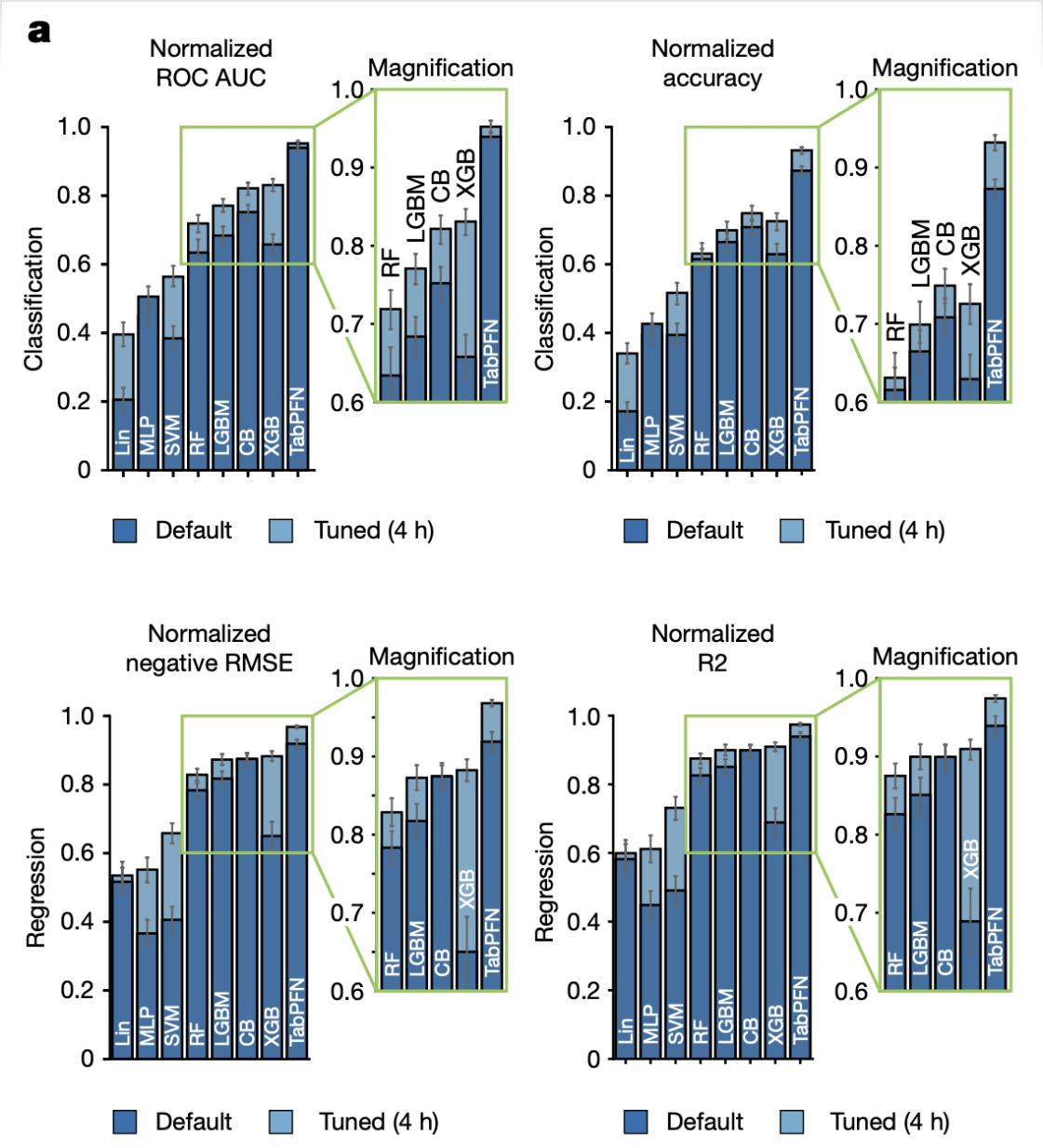
Name	OpenML ID	Domain	Features	Samples	Targets	Categorical Feats.
ada	41156	Census	48	4147	2	0
Australian	40981	Finance	14	690	2	8
blood-transfusion-service-center	1464	Healthcare	4	748	2	0
car	40975	Automotive	6	1728	4	6
churn	40701	Telecommunication	20	5000	2	4
cmc	23	Public Health	9	1473	3	7
credit-g	31	Finance	20	1000	2	13
dna	40670	Biology	180	3186	3	180
eucalyptus	188	Agriculture	19	736	5	5
first-order-theorem-proving	1475	Computational Logic	51	6118	6	0
GesturePhase Segmentation Processed	4538	Human-Computer Interaction	32	9873	5	0
jasmine	41143	Natural Language Processing	144	2984	2	136
kc1	1067	Software Engineering	21	2109	2	0
kr-vs-kp	3	Game Strategy	36	3196	2	36
madeline	41144	Artificial	259	3140	2	0
mfeat-factors	12	Handwriting Recognition	216	2000	10	0
ozone-level-8hr	1487	Environmental	72	2534	2	0
pc4	1049	Software Engineering	37	1458	2	0
philippine	41145	Bioinformatics	308	5832	2	0
phoneme	1489	Audio	5	5404	2	0
qsar-biodeg	1494	Environmental	41	1055	2	0
Satellite	40900	Environmental Science	36	5100	2	0
segment	40984	Computer Vision	16	2310	7	0
steel-plates-fault	40982	Industrial	27	1941	7	0
sylvine	41146	Environmental Science	20	5124	2	0
vehicle	54	Image Classification	18	846	4	0
wilt	40983	Environmental	5	4839	2	0
wine-quality-white	40498	Food and Beverage	11	4898	7	0
yeast	181	Biology	8	1484	10	0

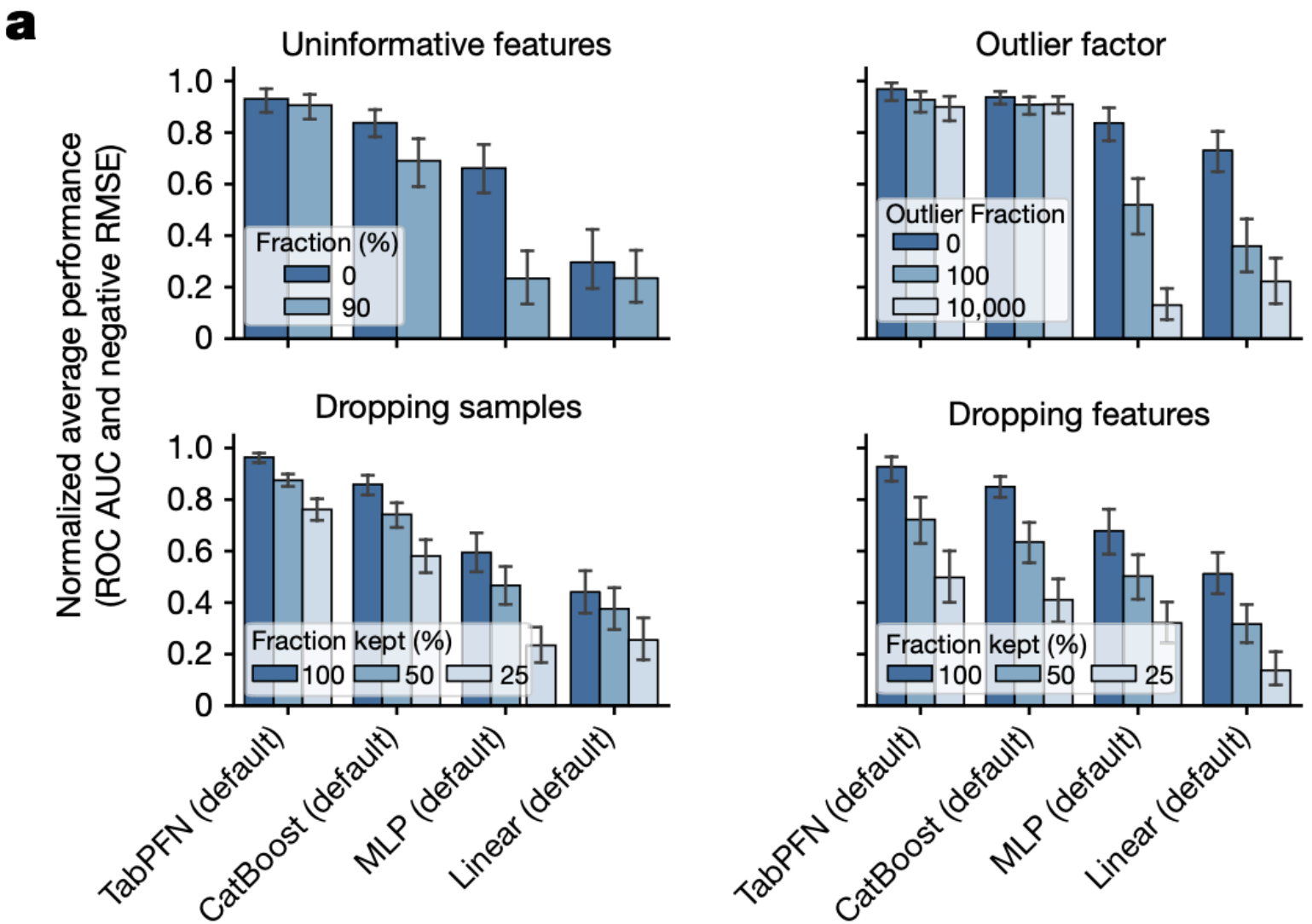
All classification tasks from the AutoML Benchmark[36] with fewer 10,000 samples and 500 features. The benchmark comprises diverse real-world tabular datasets, curated for complexity, relevance, and domain diversity.

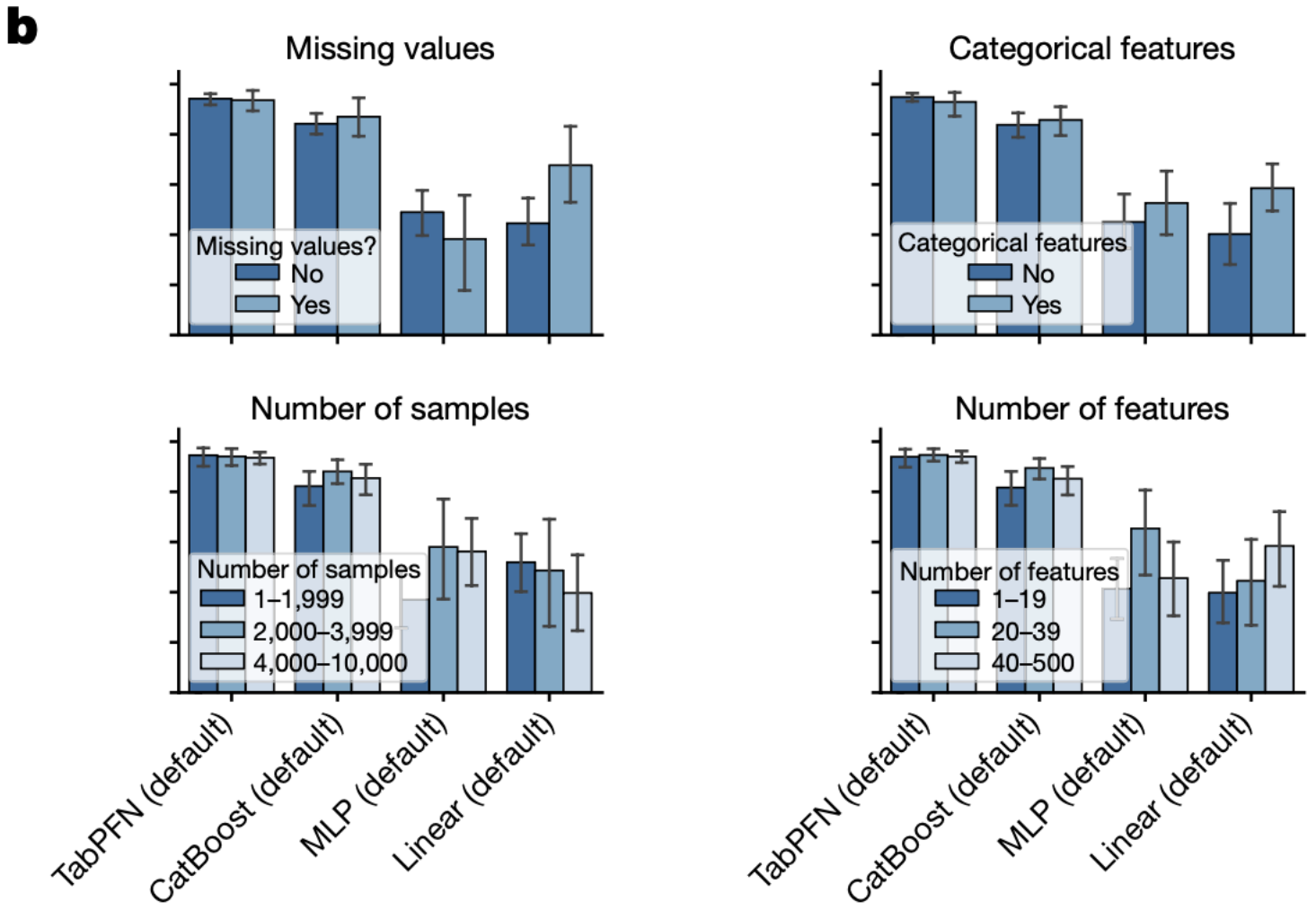
Extended Data Table 4 | List of test datasets used for primary evaluation of regression tasks

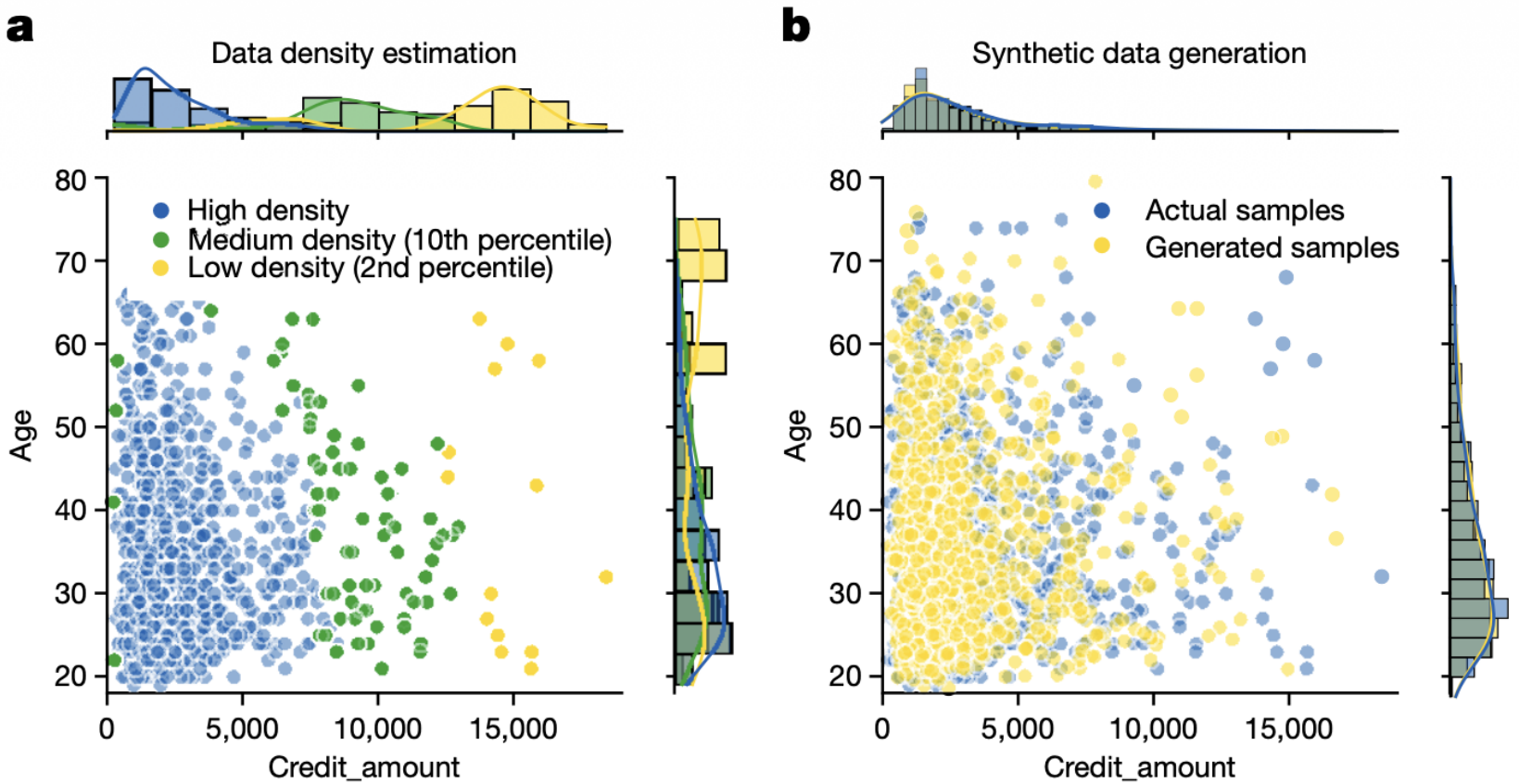
Name	OpenML ID	Domain	Features	Samples	Categorical Features
abalone	42726	Marine Biology	8	4177	1
airfoil_self_noise	44957	Aerospace Engineering	5	1503	0
auction_verification	44958	Economics	7	2043	2
boston	531	Real Estate	13	506	2
cars	44994	Automotive Engineering	17	804	0
colleges	42727	Education	44	7063	12
concrete_compressive_ strength	44959	Materials Science	8	1030	0
cpu_activity	44978	Computer Engineering	21	8192	0
energy_efficiency	44960	Architectural Engineering	8	768	0
geographical_origin _of_music	44965	Music Information Retrieval	116	1059	0
grid_stability	44973	Power Systems Engineering	12	10000	0
house_prices_nominal	42563	Real Estate	79	1460	43
kin8nm	44980	Robotics	8	8192	0
Mercedes_Benz_ Greener_ Manufacturing	42570	Manufacturing	376	4209	8
MIP-2016-regression	43071	Operations Research	144	1090	1
Moneyball	41021	Sports Analytics	14	1232	6
pumadyn32nh	44981	Robotics	32	8192	0
QSAR_fish_toxicity	44970	Toxicology	6	908	0
quake	550	Geophysics	3	2178	0
SAT11-HAND-runtime-regression	41980	Computational Logic	116	4440	1
sensory	546	Food Science	11	576	11
socmob	541	Sociology	5	1156	4
space_ga	507	Political Science	6	3107	0
student_performance	44967	Education	30	649	17
tecator	505	Food Science	124	240	0
topo_2_1	422	Cheminformatics	266	8885	0
us_crime	42730	Criminology	126	1994	0
yprop_4_1	416	Cheminformatics	251	8885	0

All regression tasks from the AutoML[36] and OpenML-CTR23[37] Benchmarks with fewer 10,000 samples and 500 features. The benchmark comprises diverse real-world tabular datasets, curated for complexity, relevance, and domain diversity.


a
Normalized ROC AUC
Magnification
Classification
Lin
MLP
SVM
RF
LGBM
CB
XGB
TabPFN
Default
Tuned (4 h)
Normalized accuracy
Normalized negative RMSE
Regression
Normalized R2


a
Uninformative features
Outlier factor
Dropping samples
Dropping features
Normalized average performance
(ROC AUC and negative RMSE)
1.0
0.8
0.6
0.4
0.2
0
Fraction (%)
0
90
Outlier Fraction
0
100
10,000
Fraction kept (%)
100
50
25
TabPFN (default)
CatBoost (default)
MLP (default)
Linear (default)


b
Missing values
Missing values?
No
Yes
Categorical features
Categorical features
No
Yes
Number of samples
Number of samples
1–1,999
2,000–3,999
4,000–10,000
Number of features
Number of features
1–19
20–39
40–500
TabPFN (default)
CatBoost (default)
MLP (default)
Linear (default)
TabPFN (default)
CatBoost (default)
MLP (default)
Linear (default)


a
Data density estimation
High density
Medium density (10th percentile)
Low density (2nd percentile)
Age
Credit_amount
b
Synthetic data generation
Actual samples
Generated samples
Age
Credit_amount

References

- Hollmann, Müller, Purucker, Krishnakumar, Körfer, Ho, Schirmeister, Hutter. *Accurate predictions on small data with a tabular foundation model.* Nature, 2025.
- Müller, Hollmann, Pineda, Grabocka, Hutter. *Transformers Can Do Bayesian Inference.* ICLR, 2022.
- Nagler. *Statistical Foundations of Prior-Data Fitted Networks.* ICML, 2023.
- Zhang, Tan, Tian, Li. *TabPFN: One Model to Rule Them All?* arXiv, 2025.