Hochschule Konstanz
Technik, Wirtschaft und Gestaltung

Seminar – SS25

Advanced Topics in Data Analysis and Deep Learning – Jan Schlegel

[DLT4] TabPFN

*https://github.com/**jsjs19xx/DLT4.git***

Agenda

1. Einleitung & Motivation
2. PFNs
3. Architektur von TabPFN
4. TabPFN Prior
5. TabPFN Training (Prior-Fitting)
6. Inferenz & Code-Live-Demo
7. Ergebnisse & Analyse
8. Limitationen & Ausblick
9. Q & A

Einleitung & Motivation

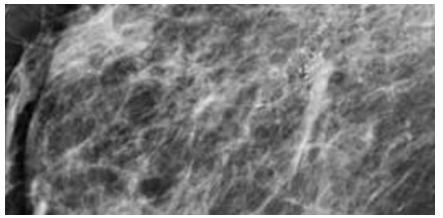
Tabellarische Daten dominieren

Einleitung & Motivation

TabPFN: A Transformer That Solves Small Tabular Classification Problems in a Second (2023)

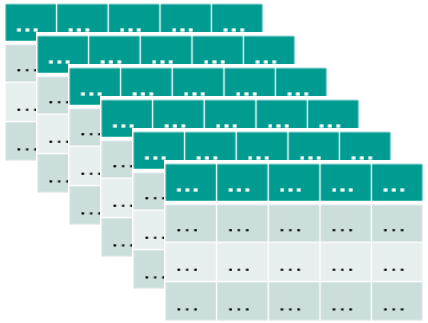
(Noah Hollmann, Samuel Müller, Katharina Eggensperger, Frank Hutter)

mean radius	mean texture	mean perimeter	…	worst symmetry	worst fractal dimension	Diagnosis
17.99	10.38	122.8	…	0.4601	0.1189	0
20.57	17.77	132.9	…	0.275	0.08902	0
19.69	21.25	130	…	0.3613	0.08758	0
⋮	⋮	⋮	⋮	⋮	⋮	⋮
19.51	**16.03**	**119**	**…**	**0.213**	**0.1152**	**?**

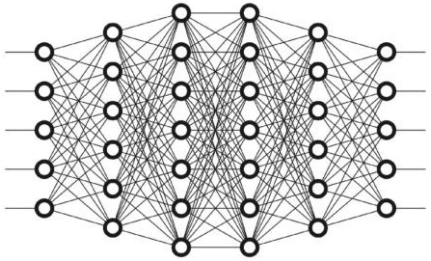


`sklearn.datasets: breast_cancer`

Einleitung & Motivation
Herkömmliche Ansätze

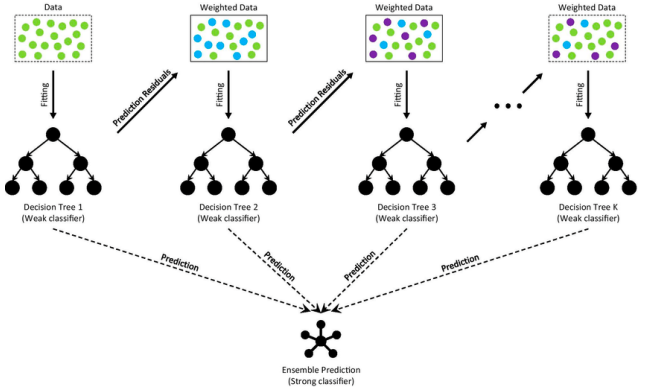


oft kleine Klassifizierungsaufgaben mit
$\leq$ **1000 Trainingsdaten**

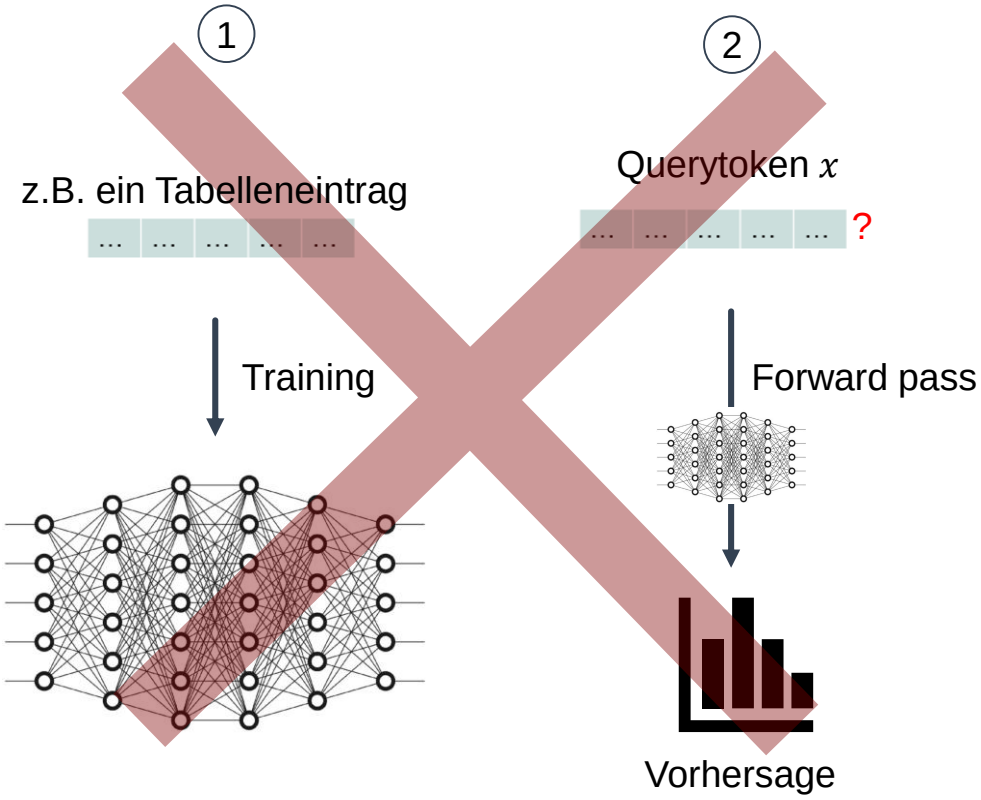


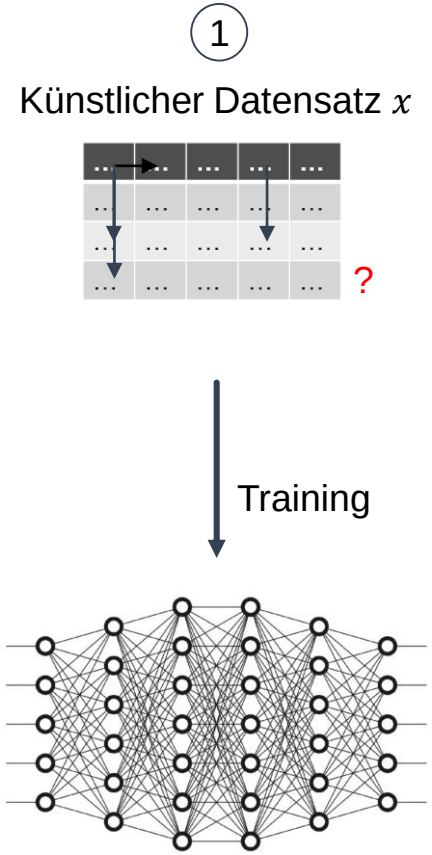
DL-Ansätze bisher nicht geeignet,
aufgrund geringer Trainingsdaten
Overfitting, nicht robust

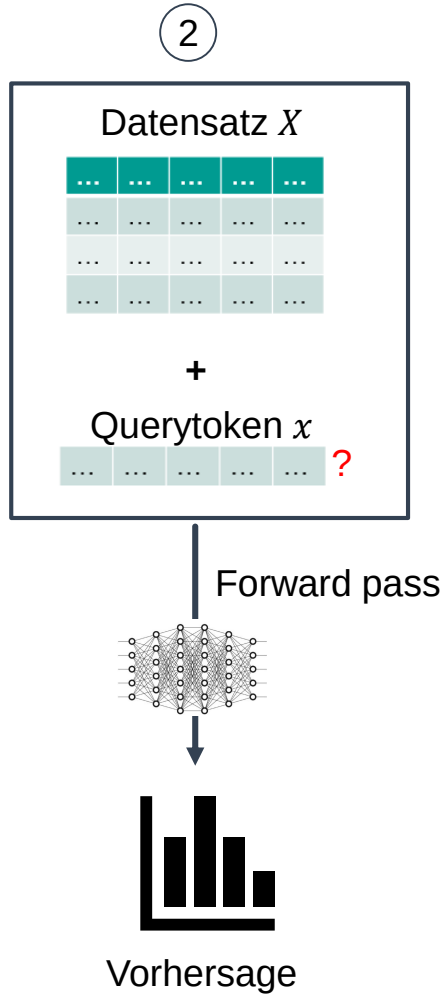
Bei tabellarischer Klassifikation
dominieren **Gradient-Boosted Decision Trees**



Einleitung & Motivation
Grundidee





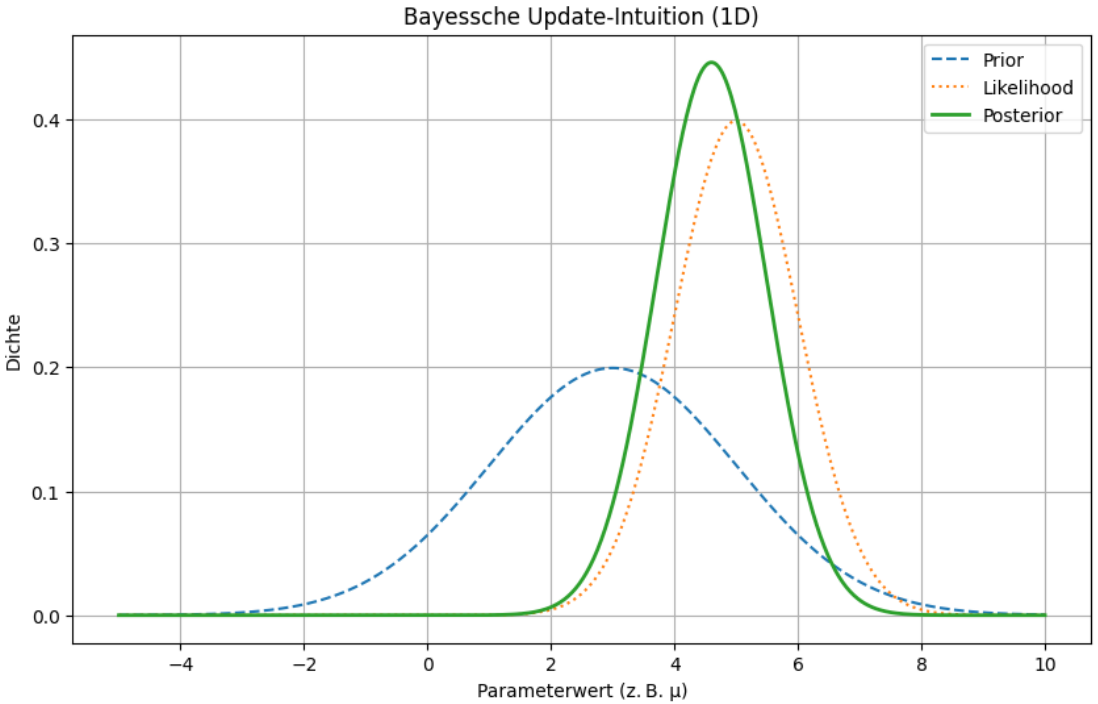


PFNs (Prior-Fitted-Networks)

Bayesianische Grundidee

$$\underbrace{p(\theta \mid D)}_{\text{Posterior}} = \frac{\overbrace{p(D \mid \theta)}^{\text{Likelihood}}\ \overbrace{p(\theta)}^{\text{Prior}}}{\underbrace{p(D)}_{\text{Evidence}}}$$

Bayes Formel



Posterior-Predictive-Distribution (PPD)

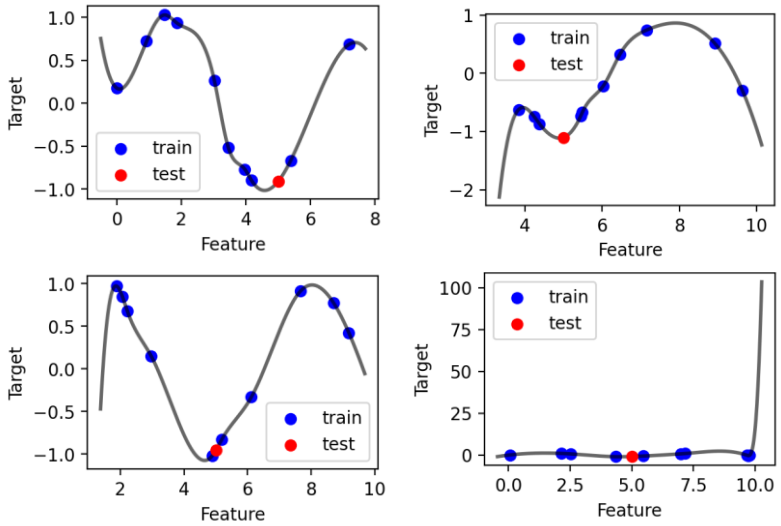
$$\overbrace{p(y \mid x, D)}^{\text{Vorhersage eines konkreten Modells } \varphi} \propto \int_{\Phi} p(y \mid x, \varphi)\ \underbrace{p(D \mid \varphi)}_{\text{Likelihood von } \varphi \text{ für die Trainingsdaten}}\ \overbrace{p(\varphi)}^{\text{Prior}}\ d\varphi$$

nicht exakt berechenbar bei großem Hypothesenraum Φ, wird z.B. mit *MCMC* oder *VI* approximiert

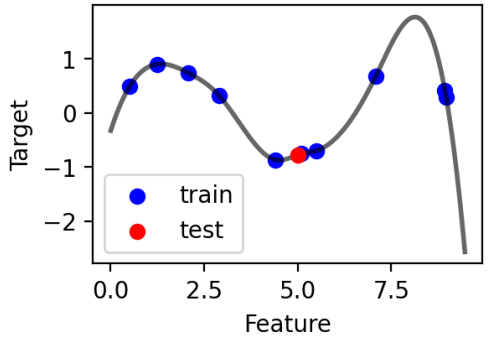
→ In der Bayesschen Sichtweise berücksichtigt man alle möglichen Modelle φ und gewichtet sie nach deren Anpassung an die Trainingsdaten (Posterior).

PFNs

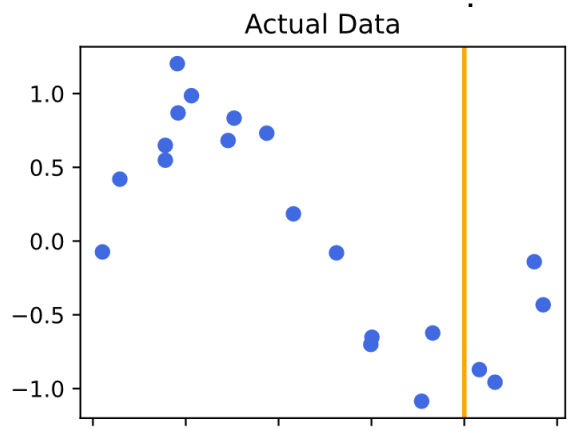
PFN-Idee



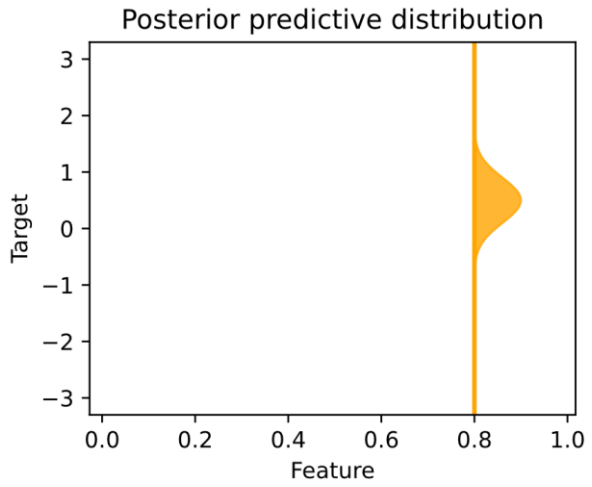
vom Prior samplen:
Datengenerierung (x10mio)



Anlernen des PFN_θ um Test aus Trainingsdaten vorherzusagen



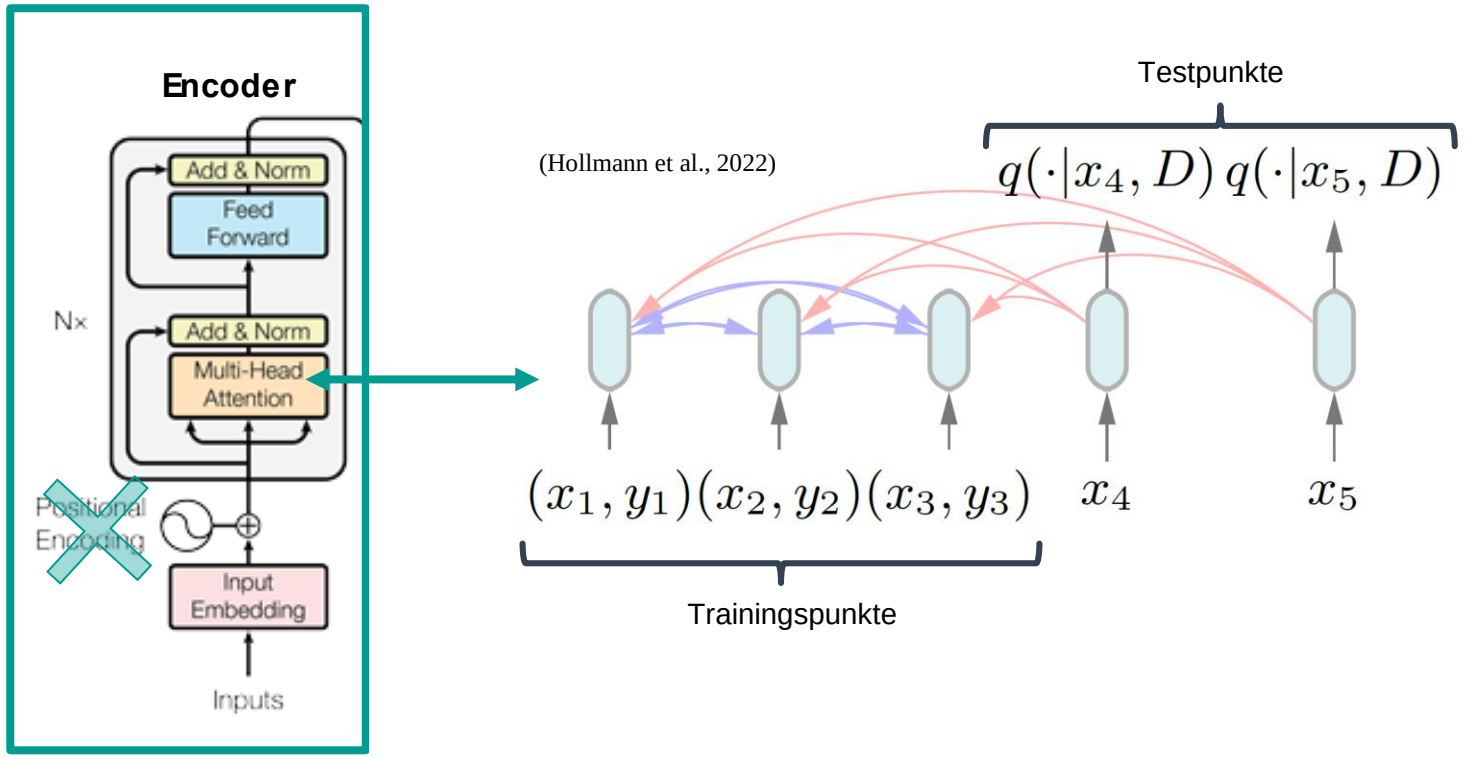
PFN_θ ein forward pass



angelehnt an: https://youtu.be/qFnYgM2Yvfs?t=846

Architektur von TabPFN

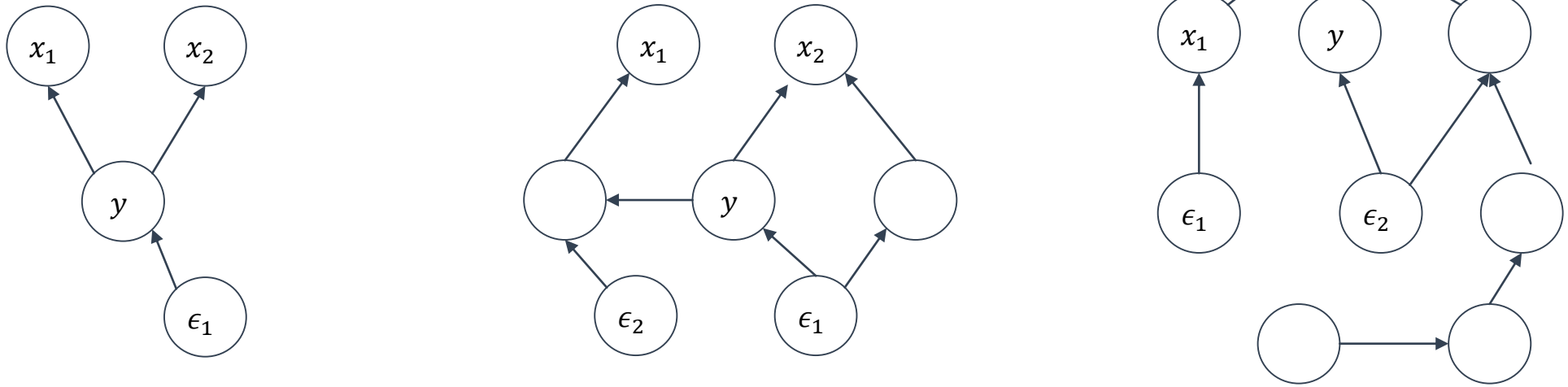
Attention Mechanism

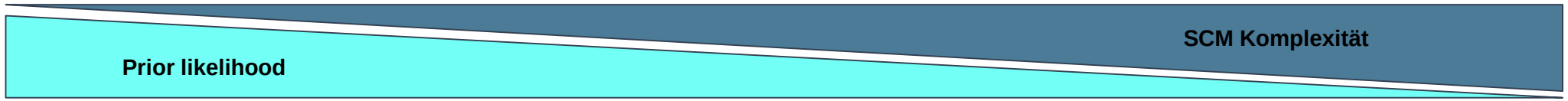


- Transformer ähnlich zu Vaswani et al., 2017, jedoch nur der Encoder, da TabPFN keine Sequenz generiert sondern klassifiziert.
- Kein Positionsencoding, da Trainings- und Testpunkte als Menge und nicht als Sequenz betrachtet werden („Permutation Invariance“).
- Kein Data Leakage: Trainingsdaten dürfen keine Informationen über Testdaten enthalten.
- Unabhängige Testdaten: Testdaten dürfen keine gegenseitigen Abhängigkeiten aufweisen.

TabPFN Prior

SCMs (Structural Causal Models)



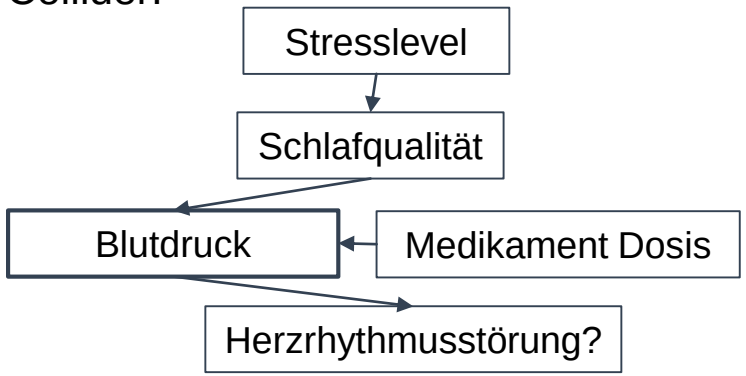


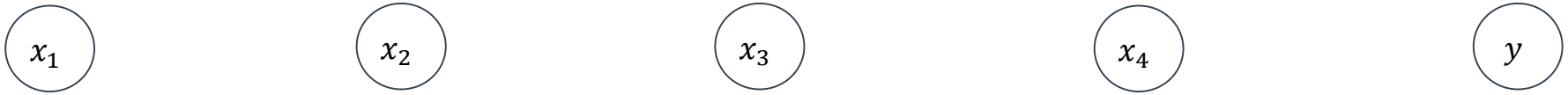
Angelehnt an https://youtu.be/9cE8lqQiLyM?t=1596

TabPFN Prior

SCMs – Warum ein SCM Prior?

Tabellendaten spiegeln oft Kausalstrukturen wider. Solche Zusammenhänge beschreibt man mit Structural Causal Models (SCMs). Beispiel Gesundheitsnetz mit Collider:



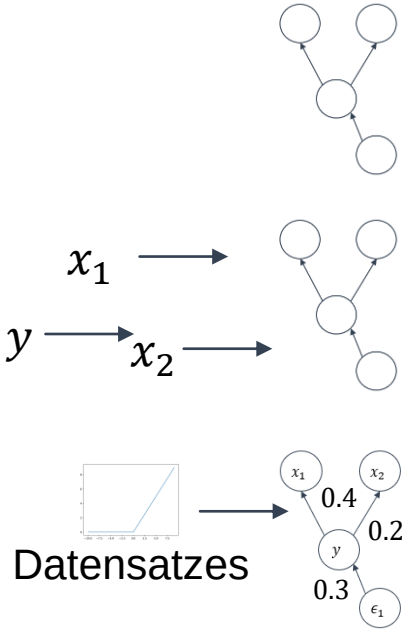


Stresslevel	Schlafqualität	Blutdruck	Medikament Dosis	HR-Störung?
…	…	…	…	…
…	…	…	…	…

TabPFN Prior
SCMs – Anwendung

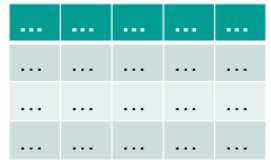
Abfolge:

1. SCM-Erzeugen
   - Ziehe zufällig einen DAG G (k Knoten = potentielle Spalten)
2. Features & Target wählen
   - Wähle zufällig k Knoten als Feature-Spalten
   - Wähle einen Knoten y als Target-Spalte
3. Datensatz simulieren
   - Ziehe n-mal alle Rauschvariablen
   - Ziehe für jeden DAG i eine Funktion f_i (ReLU-Netz, Polynom, Tanh,…)
   - Propagiere Werte entlang G und lies schließlich z_x und z_y aus. Jede Ziehung erzeugt eine Zeile des Datensatzes



So entstehen Aufgaben, in denen:

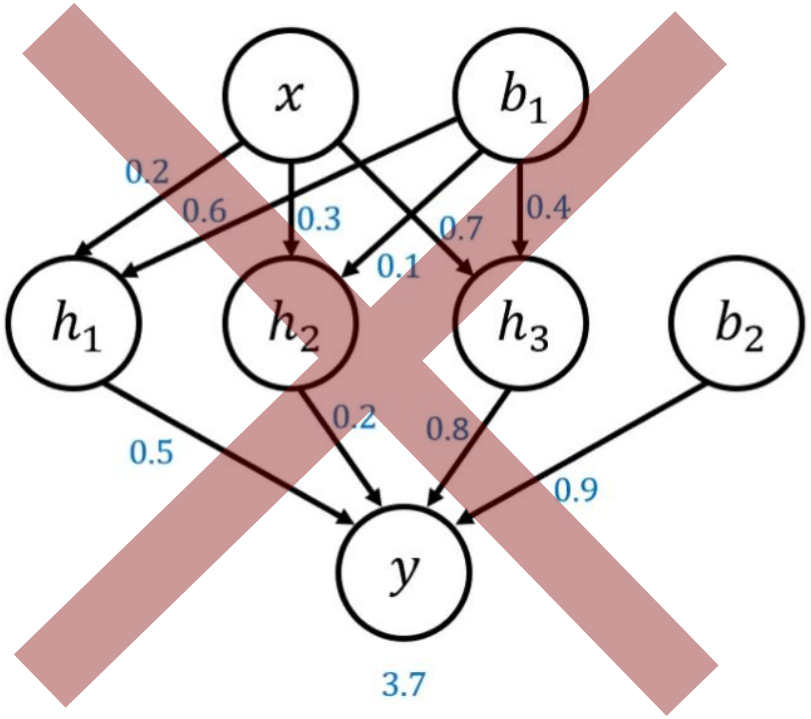
- Feature-Abhängigkeiten $p\left(z_{x_i}, z_{x_j}\right) \neq p(z_{x_i})p(z_{x_j})$ vorkommen,
- das Target Ursache oder Wirkung der Features sein kann,
- Nichtlinearität und Heteroskedastizität natürlich auftreten.

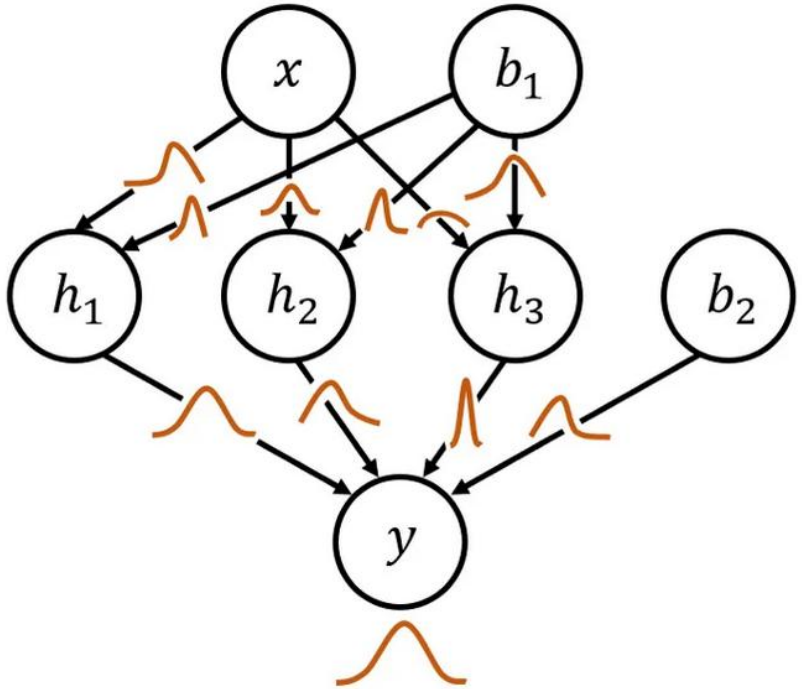


(Hollmann et al., 2022)

TabPFN Prior

BNNs (Bayesian Neuronal Networks)





https://medium.com/@costaleirbag/a-first-insight-into-bayesian-neural-networks-bnn-c767551e9526

TabPFN Prior

BNNs – Anwendung

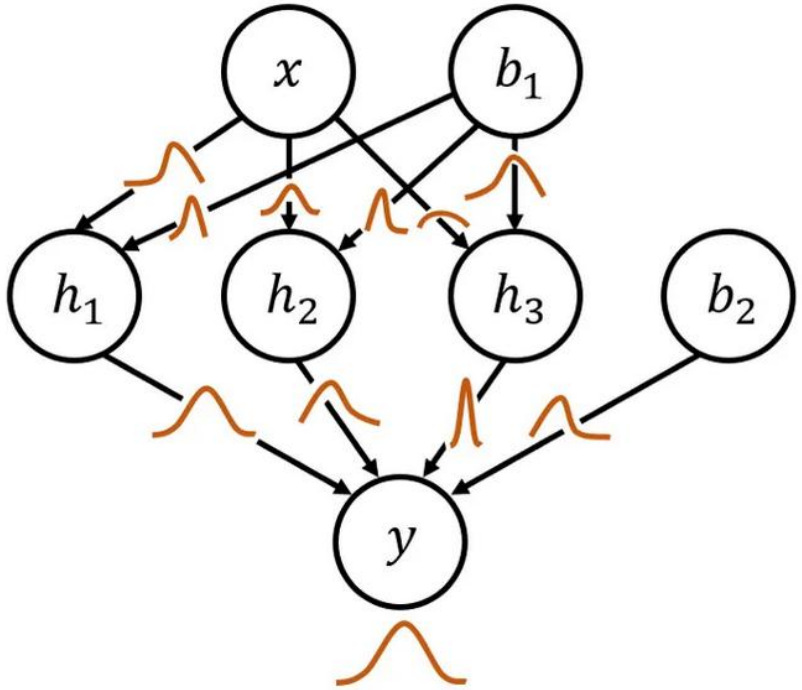
1. Architektur ziehen

$$A \sim \mathcal{P}(A)$$

2. Gewichte ziehen

$$\theta \mid A \sim \mathcal{P}(\theta \mid A)$$

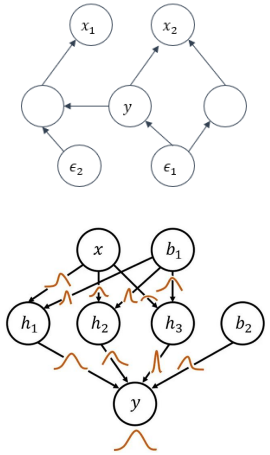
1. Für jede Zeile $i = 1, \ldots n$:
   - Eingabe $x_i \sim p(x)$
   - Rauschen $\epsilon_i \sim \mathcal{N}(0, \sigma^2)$
   - Ziel $y_i = \mathrm{f}_{\mathrm{A},\theta}(\mathrm{x_i}) + \epsilon_i$



https://medium.com/@costaleirbag/a-first-insight-into-bayesian-neural-networks-bnn-c767551e9526

TabPFN Training (Prior-Mix)

$$\text{Prior} = \begin{cases} \text{SCM mit Wahrscheinlichkeit } 0.5 \\ \text{BNN mit Wahrscheinlichkeit } 0.5 \end{cases}$$



Damit sieht das Modell gleichermaßen

- strukturelle, kausale Datensätze (SCM) – viele Pfeile, Collider, Forks …
- funktional-glatte Datensätze (BNN) – hoch-nichtlineare, aber “black-box”-Abbildungen.

Das vergrößert die Aufgabenvielfalt und macht den gelernten Prior robuster.

(Hollmann et al., 2022)

TabPFN Training
Loss (Kreuzentropie)

Die Erwartung wird über Datensätze genommen, die nach einer bestimmten Datenverteilung generiert werden.
Dabei wird ein Datensatz in ein Testset und ein Trainingsset unterteilt.

$$\underbrace{\mathcal{L}_{PFN}}_{\text{Verlustfunktion des PFNs}} = \underbrace{\mathop{\mathbb{E}}_{(\{(x_{test}, y_{test})\} \cup D_{train}) \sim p(D)}}_{} \underbrace{[-\log q_\theta(y_{test} \mid x_{test}, D_{train})]}_{\text{Für jedes Testbeispiel wird der negative Log-Likelihood der Vorhersage (das vom PFN gelernte Modell) berechnet, bedingt auf den Trainingsdaten und dem Testinput.}}$$

Minimieren dieser Verlustfunktion: lernen die wahre **Bayessche posterior-prädiktive Verteilung (PPD)** zu approximieren.

(Hollmann et al., 2022)

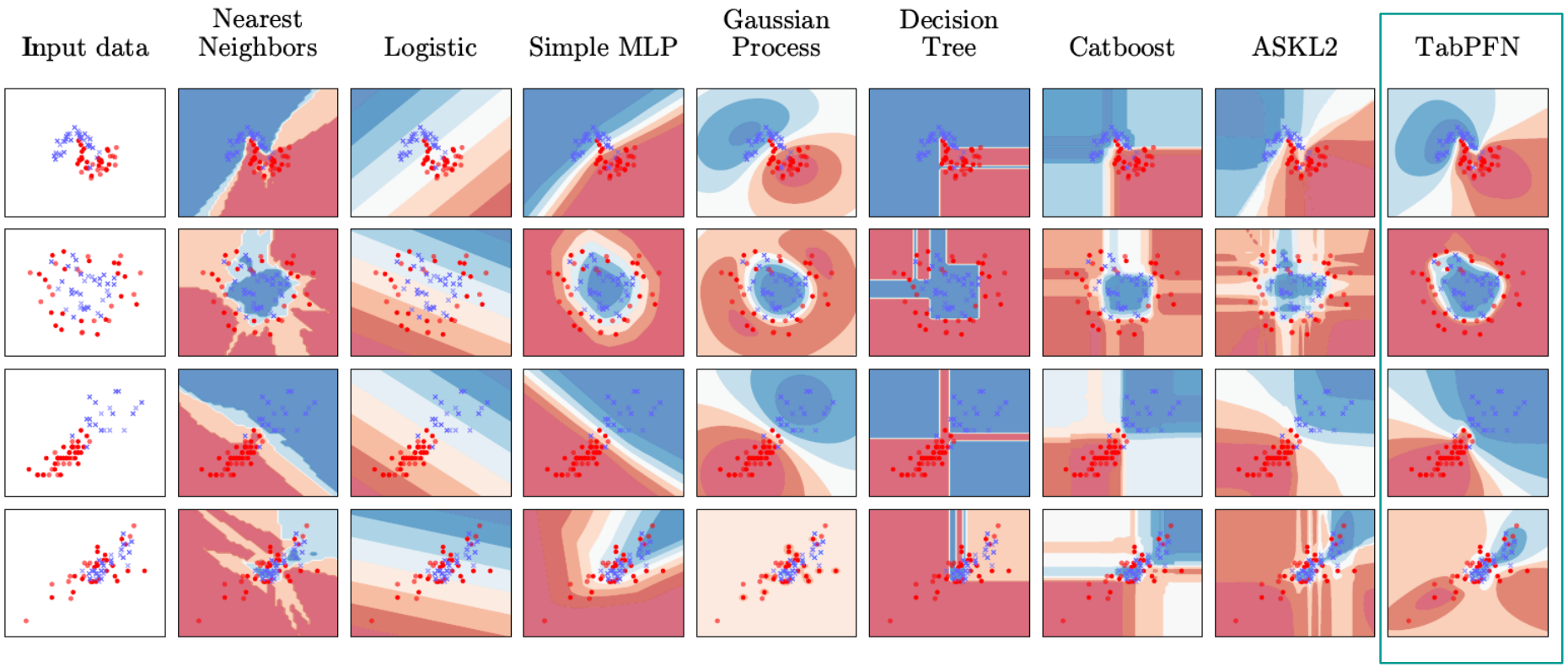
Inferenz & Code-Live-Demo

	precision	recall	f1-score	support
malignant	0.98	0.95	0.96	43
benign	0.97	0.99	0.98	71
accuracy			0.97	114
macro avg	0.97	0.97	0.97	114
weighted avg	0.97	0.97	0.97	114

https://github.com/jsjs19xx/DLT4/blob/main/src/tabpfn_breast_cancer_notebook.ipynb

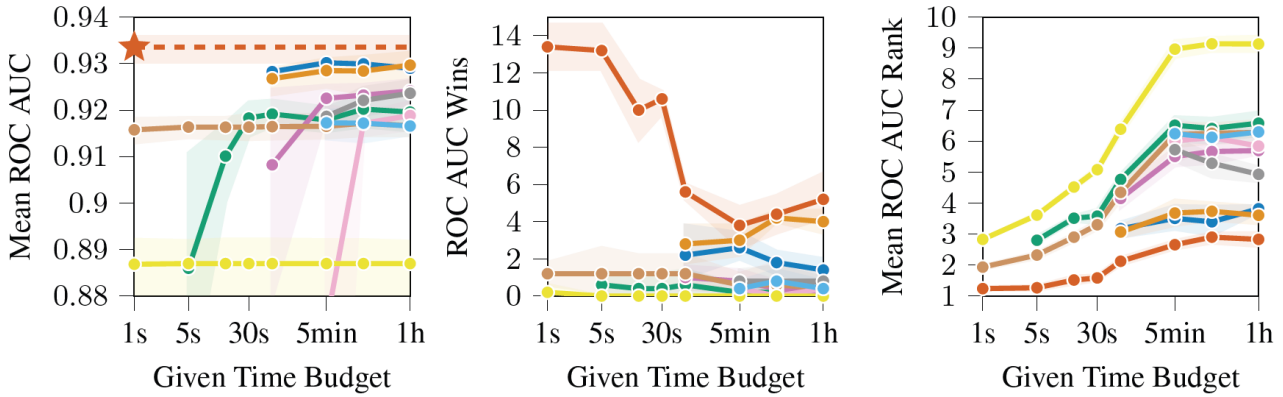
Ergebnisse & Analyse

Entscheidungsgrenze auf dem „toy-Datensatz“



(Hollmann et al., 2022, Pedregosa et al., 2011)

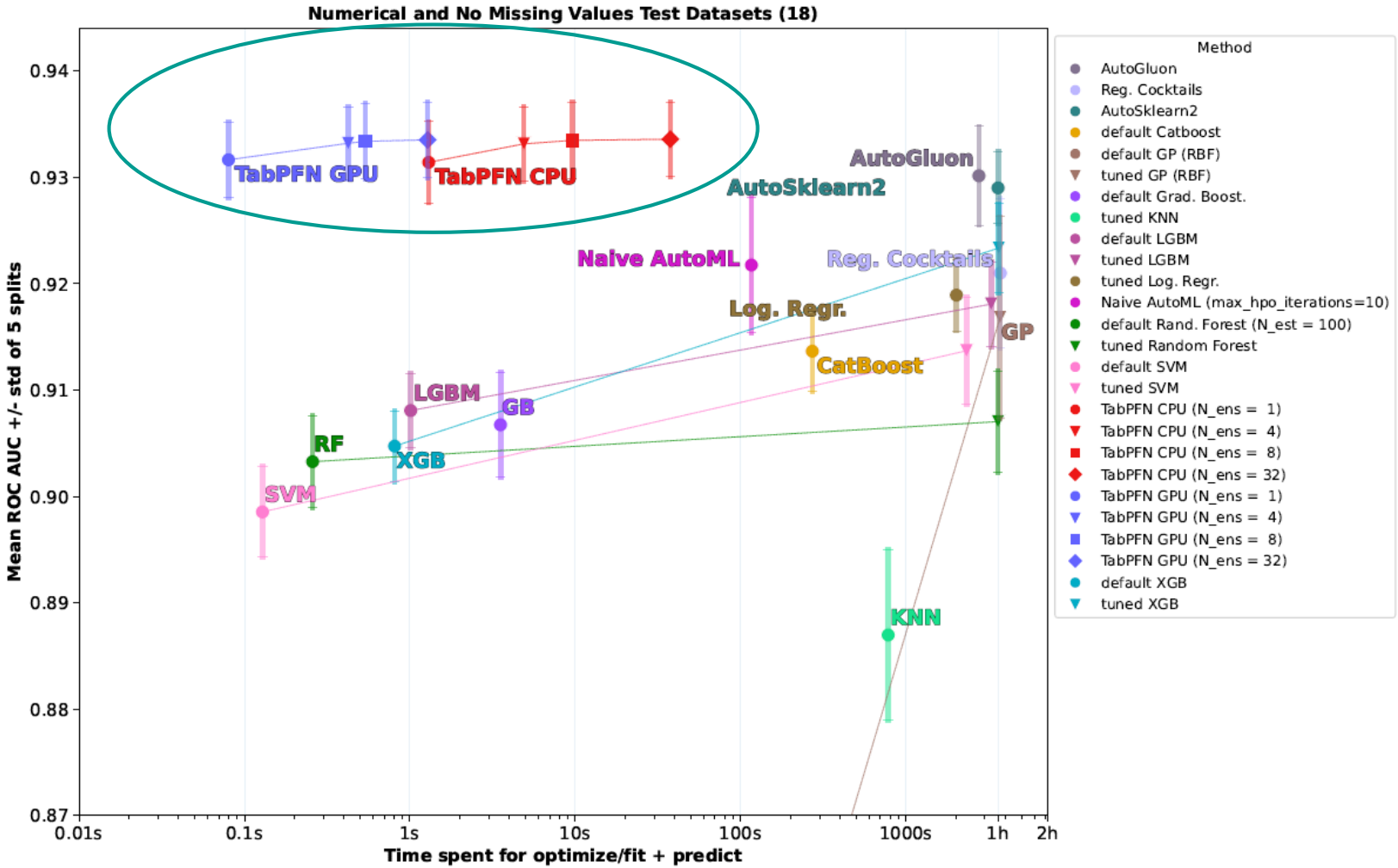
Ergebnisse & Analyse



	LightGBM	CatBoost	XGBoost	ASKL2.0	AutoGluon	TabPFN$_{n.e.}$	TabPFN	TabPFN + AutoGluon
M. rank AUC OVO	6.9722	4.9444	6.1944	4.4722	4	3.8056	2.9444	**2.6667**
Mean rank Acc.	6.8889	4.9722	6.0556	5.1667	3.8889	3.8889	2.8889	**2.25**
Mean rank CE	5.7778	5.4444	6	6.4167	3.1111	4.1389	3.0278	**2.0833**
Mean AUC OVO	0.92±.013	0.924±.011	0.924±.01	0.929±.0096	0.93±.0091	0.932±.0088	**0.934**±.0086	**0.934**±.0084
Mean Acc.	0.862±.012	0.864±.011	0.866±.011	0.87±.014	0.881±.01	0.873±.0095	0.879±.0089	**0.886**±.0094
Mean CE	0.75±.039	0.747±.029	0.759±.04	0.813±.073	0.714±.014	0.727±.021	0.716±.019	**0.711**±.014
Mean time (s) (Tune + Train + Predict)	3280	3746	3364	3601	3077	**1.301** (CPU) **0.0519** (GPU)	37.59 (CPU) 0.6172 (GPU)	3109 (CPU) 3077 (GPU)

(Hollmann et al., 2022)

Ergebnisse & Analyse



(Hollmann et al., 2022)

Limitationen & Ausblick

Skalierung

Bislang nur Verarbeitung von ca. 1000 Zeilen und 100 Features möglich

→ Größere Datensätze mittels Sparse-/Longformer-Attention integrieren

Datentyp

Bislang Prior & Experimente rein numerisch, Leistung sinkt bei kategorischen Spalten und NAs.

→ Kategoriale Features besser kodieren, Missing-Value Handling ins Modell aufnehmen

Nicht informative Features

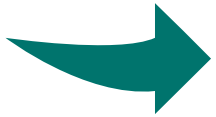
Viele Dummy-Spalten verschlechtern AUC

→ Prior um Rauschen ergänzen und mehr Robustheit gegenüber irrelevanten Features schaffen

Kausaler Scope

Bisher nur prädiktiv

→ Interventionen & Counterfactuals über SCM-Prior abschätzen



(Hollmann et al., 2022)

Q & A

Literaturverzeichnis

- *Hollmann, N., Müller, S., Eggensperger, K. & Hutter, F. (2022). TabPFN: A Transformer That Solves Small Tabular Classification Problems in a Second. arXiv (Cornell University). https://doi.org/10.48550/arxiv.2207.01848*

Hochschule Konstanz
Technik, Wirtschaft und Gestaltung

Vielen Dank für Ihre Aufmerksamkeit!