Tabular data

and Deep learning - state of affairs

Ivan Rubachev - 2022

Who am I

Outline

What we'll talk about

- what's the problem (tabular data)

- Survey:
  - History up until very recently (just good old ml)
  - Recent times (a bunch of tabular deep learning papers)
  - Better times (this bunch of tabular deep learning doesn't really *work*)
  - Our new stuff (but is it?)

- Closing remarks

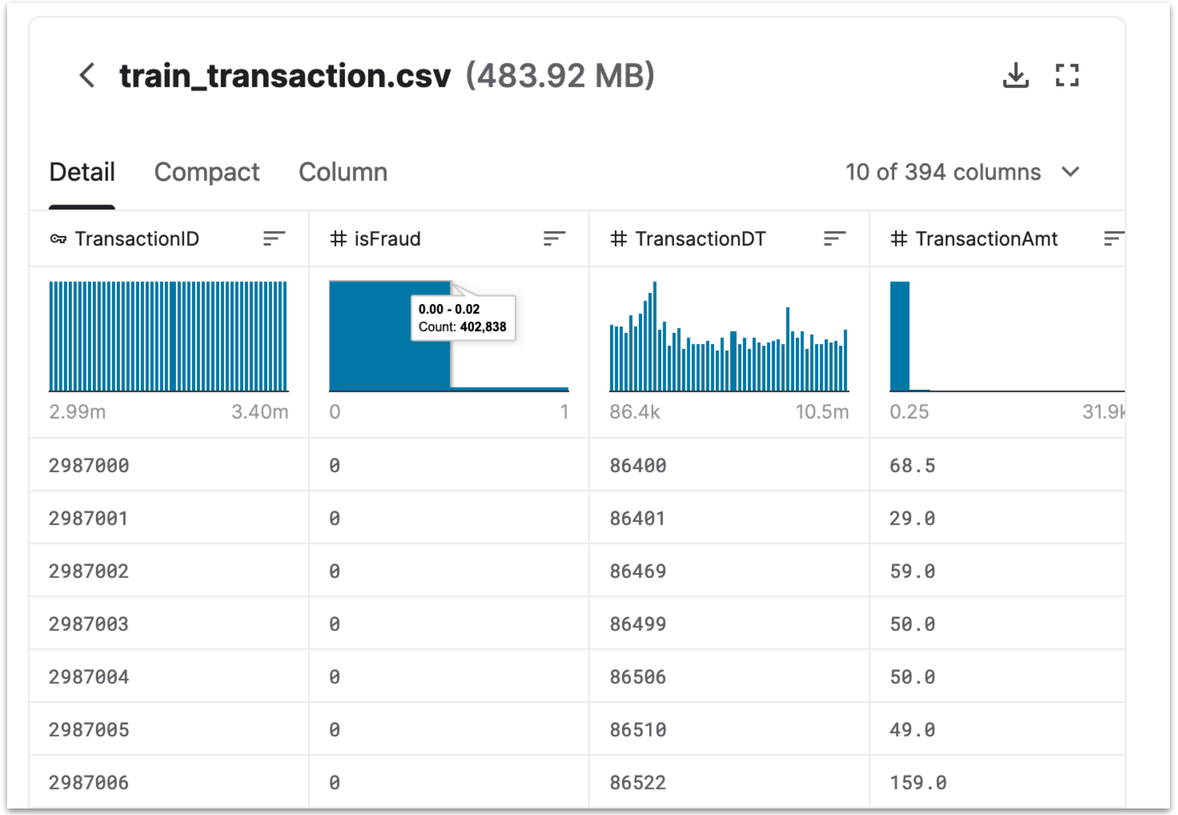
The field

Tabular data

Tabular data — two dimensional tables

rows ~ objects

columns ~ features



The field

Deep learning

Deep learning

~ more like a collection of methods

~ approach to building models

~ **very** successful in some fields

Transformers

The field

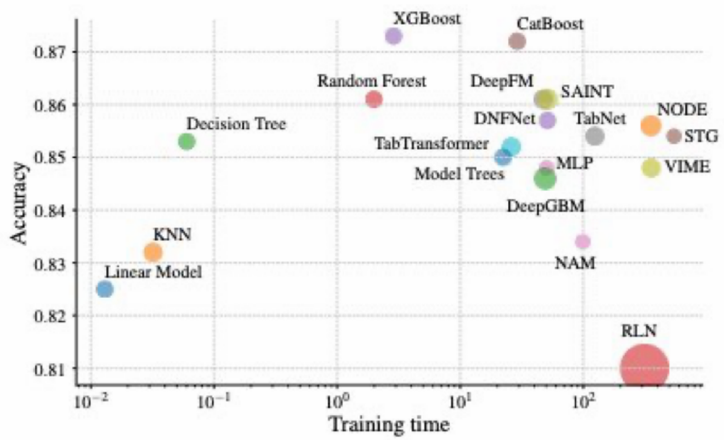
Tabular data + Deep learning =

Bojan Tunguz
@tunguz

XGBoost Is All You Need

Deep Neural Networks and Tabular Data: A Survey

arxiv.org/pdf/2110.01889...



Adult dataset 👀

Mark Tenenholtz
@marktenenholtz

Why is XGBoost so incredible?

It's not because:

- It's always the most accurate
- It's always the fastest
- It's always the most interpretable

It's because:

- It's always the fastest to a good (tabular) solution.

Yeah, I can beat it with deep learning. But time is money.

Timeline

2016/2017

Pre DL
Linear models all the way
XGBoost and CatBoost came out

2021

DL
Tabular DL papers
Our paper

Today+

Post DL (Tabular DL)
Our papers
Future work

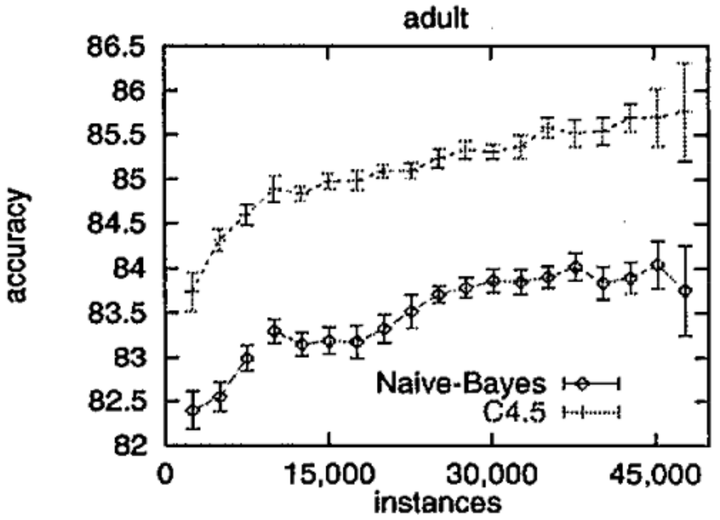
“Classical” ML algorithms

Scaling Up the Accuracy of Naive-Bayes Classifiers: a Decision-Tree Hybrid

Ron Kohavi
Data Mining and Visualization
Silicon Graphics, Inc.
2011 N. Shoreline Blvd
Mountain View, CA 94043-1389
ronnyk@sgi.com

3.5. Experimental results

To verify the effects of sample selection bias experimentally, we apply Naive Bayes, logistic regression, C4.5 and SVMLight (soft margin) (Joachims, 2000b) to the Adult dataset, available from the UCI Machine Learning repository (Blake & Merz, 1998). We assume that the original dataset is not biased and artificially simulate biasedness by generating a value for s for each



Adult dataset 👀

Deep learning is here

Tabular, but not intentionally

Self-Normalizing Neural Networks

Günter Klambauer **Thomas Unterthiner** **Andreas Mayr**

Sepp Hochreiter
LIT AI Lab & Institute of Bioinformatics,
Johannes Kepler University Linz
A-4040 Linz, Austria
{klambauer,unterthiner,mayr,hochreit}@bioinf.jku.at

$$\text{selu}(x) = \lambda \begin{cases} x & \text{if } x > 0 \\ \alpha e^x - \alpha & \text{if } x \leqslant 0 \end{cases}.$$

121 UCI Machine Learning Repository datasets. The benchmark comprises 121 classification datasets from the UCI Machine Learning repository [10] from diverse application areas, such as physics, geology, or biology. The size of the datasets ranges between 10 and 130,000 data points and the number of features from 4 to 250. In abovementioned work [10], there were methodological mistakes [37] which we avoided here. Each compared FNN method was optimized with respect to its architecture and hyperparameters on a validation set that was then removed from the subsequent

https://twitter.com/seluappendix

Deep learning is here

More intentional

Regularization Learning Networks: Deep Learning for Tabular Datasets

Ira Shavitt
Weizmann Institute of Science
irashavitt@gmail.com

Eran Segal
Weizmann Institute of Science
eran.segal@weizmann.ac.il

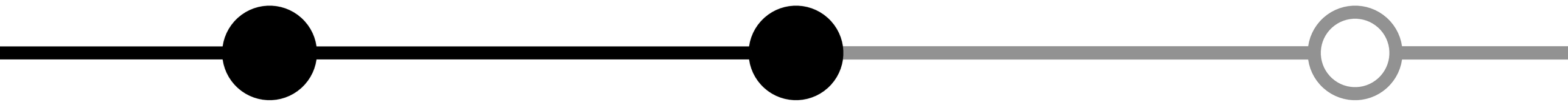
They introduce regularization hyperparameter for each weight in a NN

But optimize via SGD

Test on tabular DL problems

Conclusion: GBDT is still better

Timeline



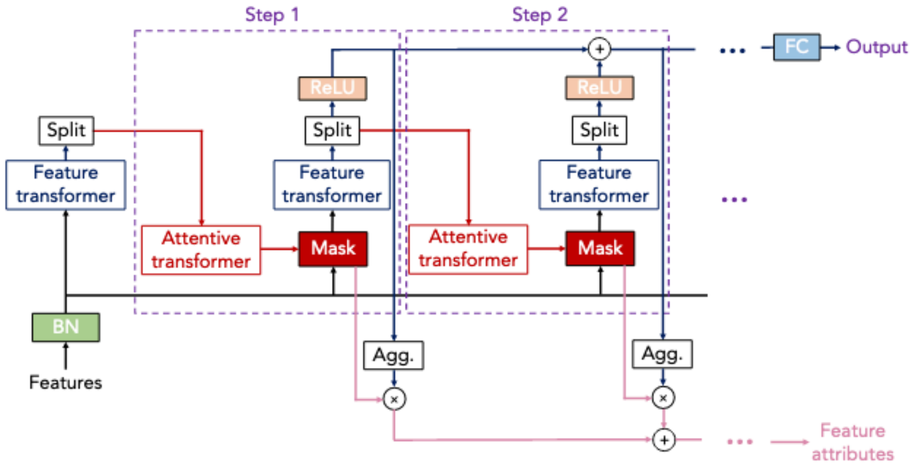
Pre DL
Linear models all the way
XGBoost and CatBoost came out

DL
Tabular DL papers
Our paper

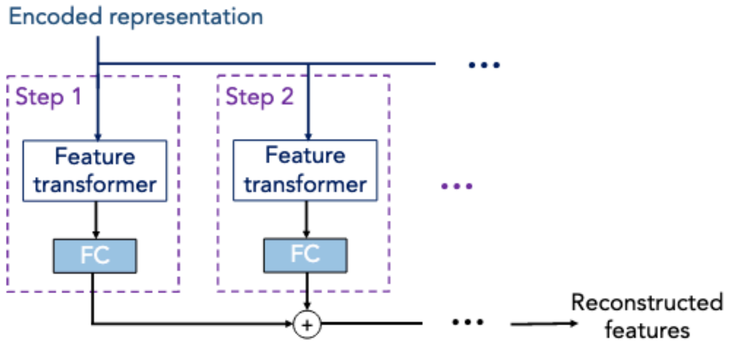
Post DL (Tabular DL)
Our papers
Future work

Tabnet

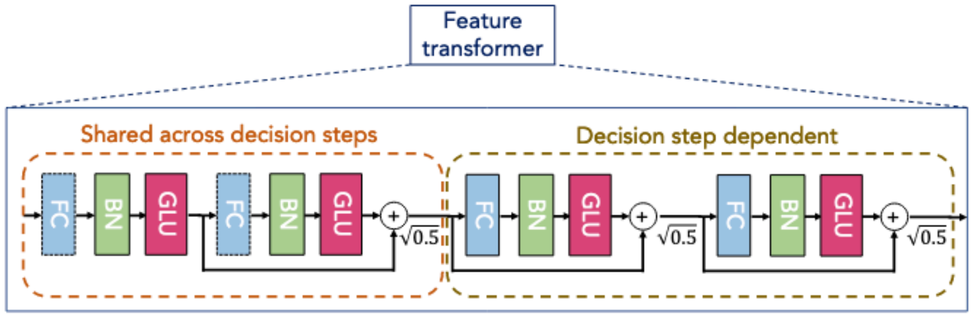
Architectures for tabular data



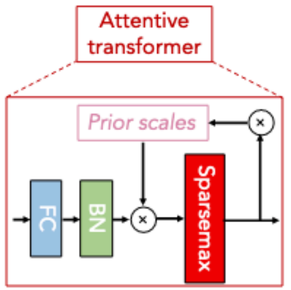
(a) TabNet encoder architecture



(b) TabNet decoder architecture



(c)



(d)

Tabnet

Architectures for tabular data

Adult dataset 👀

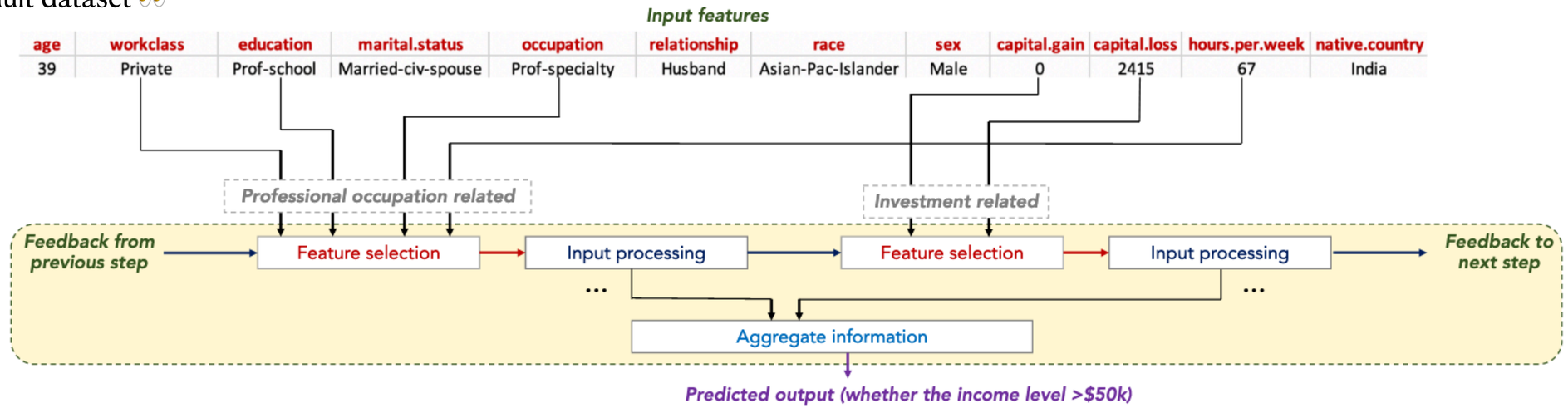


Figure 1: TabNet's sparse feature selection exemplified for Adult Census Income prediction (Dua and Graff 2017). Sparse feature selection enables interpretability and better learning as the capacity is used for the most salient features. TabNet employs multiple decision blocks that focus on processing a subset of input features for reasoning. Two decision blocks shown as examples process features that are related to professional occupation and investments, respectively, in order to predict the income level.

GrowNet

Architectures for tabular data

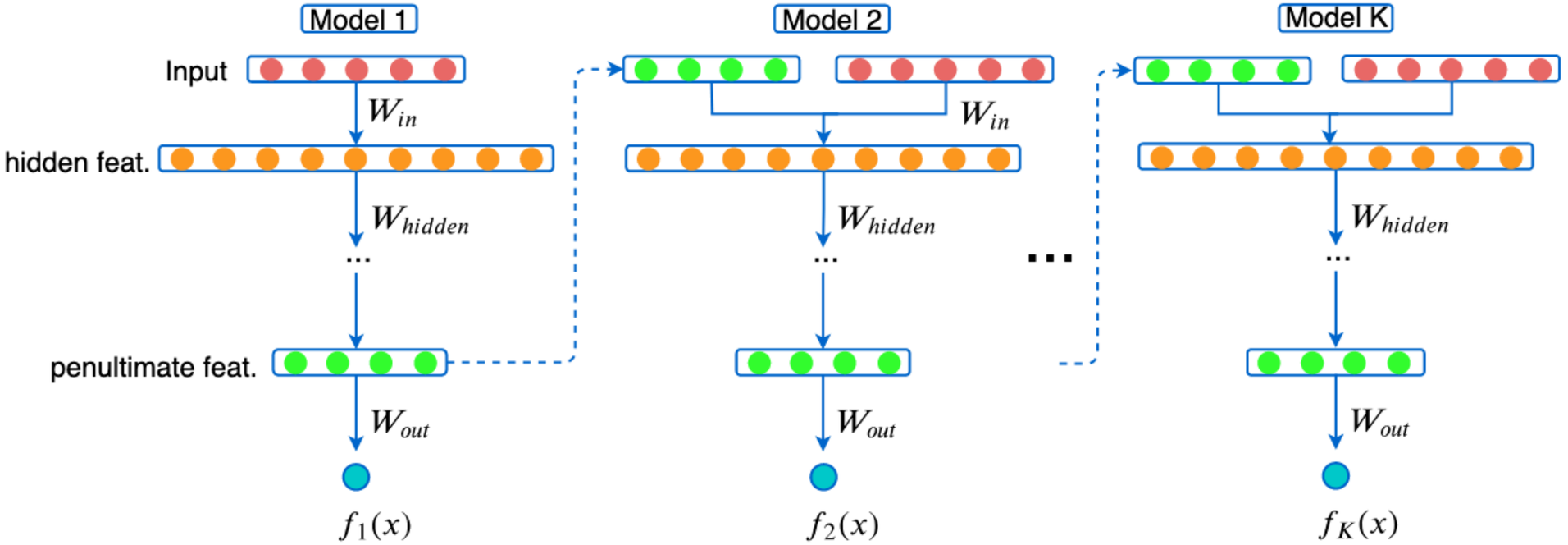


Figure 1: GrowNet architecture. After the first weak learner, each predictor is trained on combined features from original input and penultimate layer features from previous weak learner. The final output is the weighted sum of outputs from all predictors, $\sum_{k=1}^{k=K} \alpha_k f_k(x)$. Here Model K means weak learner K.

NODE

Architectures for tabular data

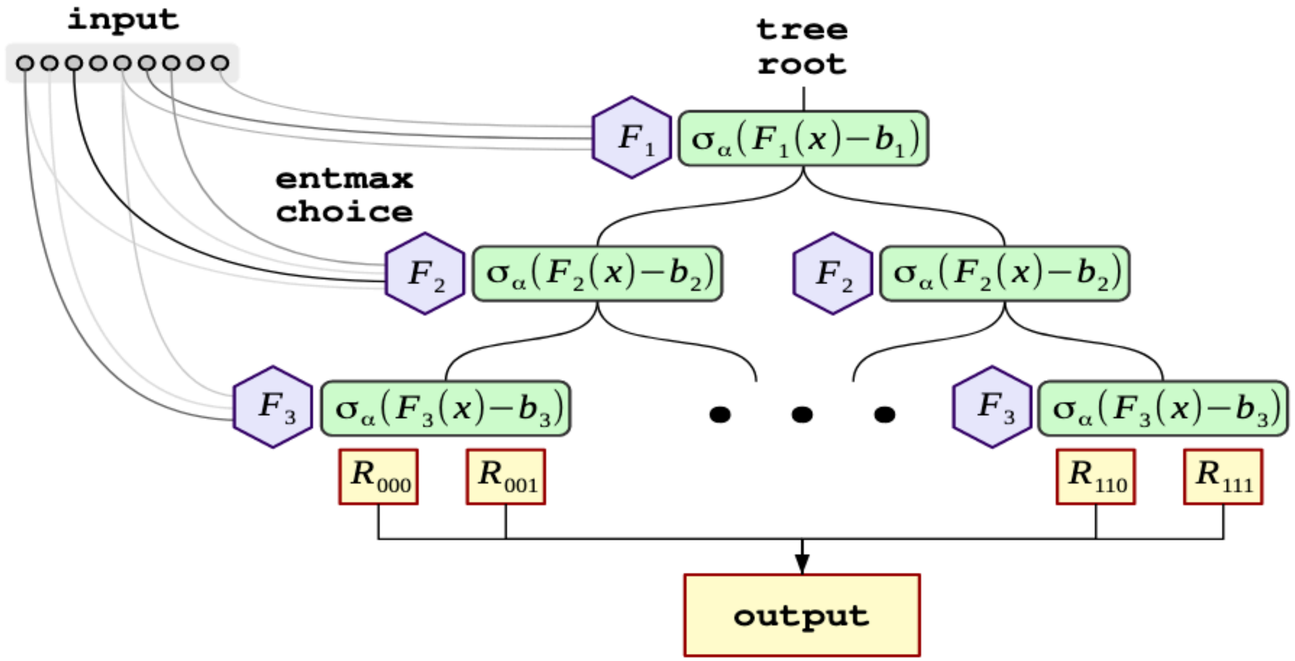


Figure 1: The single ODT inside the NODE layer. The splitting features and the splitting thresholds are shared across all the internal nodes of the same depth. The output is a sum of leaf responses scaled by the choice weights.

NODE

Architectures for tabular data

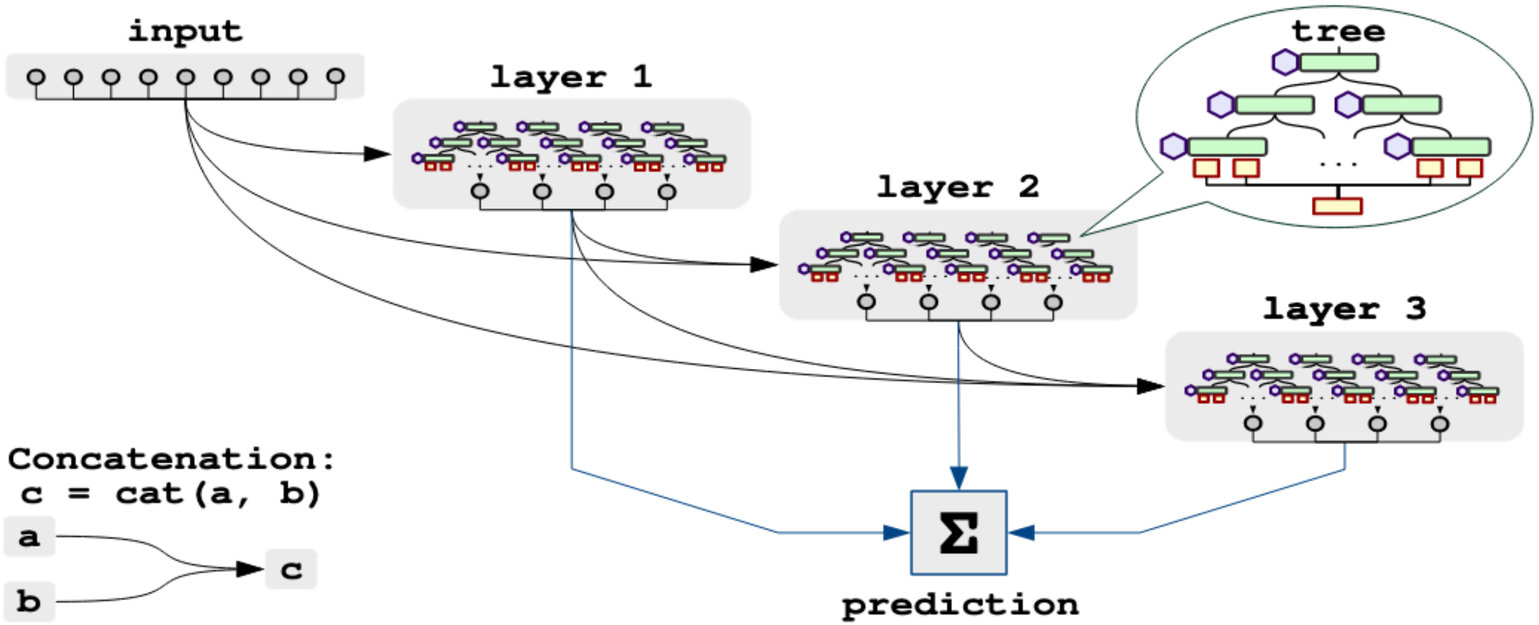
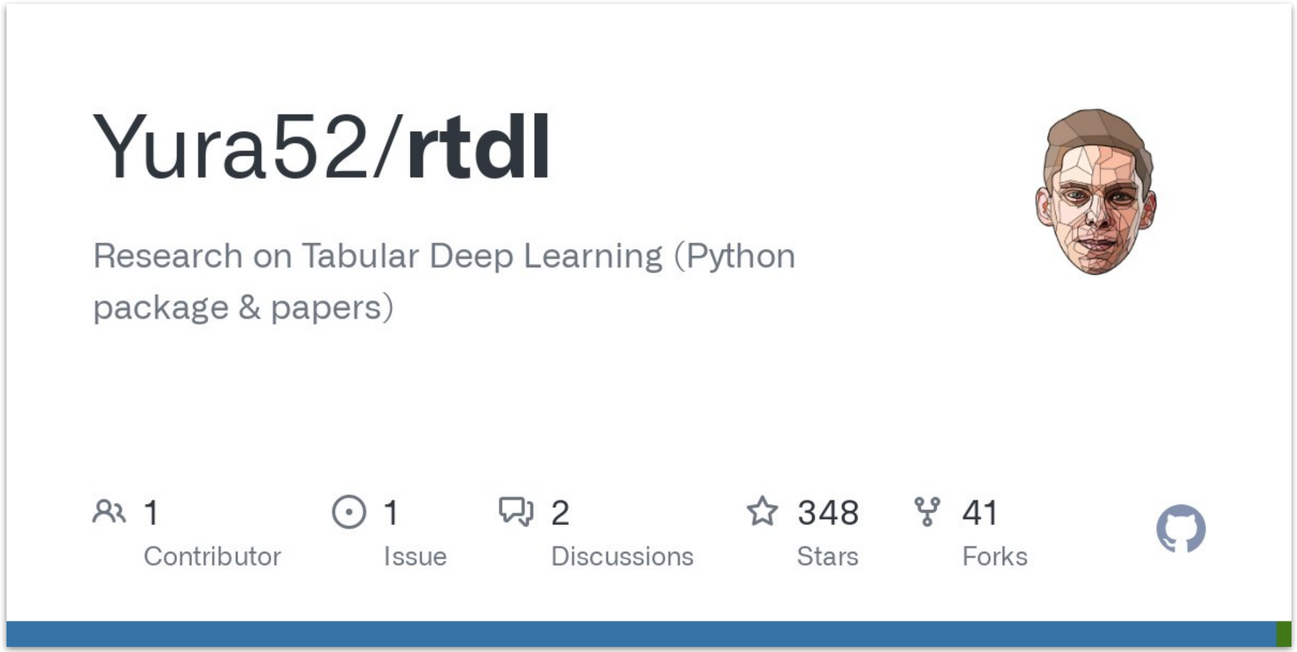


Figure 2: The NODE architecture, consisting of densely connected NODE layers. Each layer contains several trees whose outputs are concatenated and serve as input for the subsequent layer. The final prediction is obtained by averaging the outputs of all trees from all the layers.

Revisiting deep learning for tabular data

Our paper from NeurIPS 21

- But where are the results?
- Here: https://github.com/Yura52/tabular-dl-revisiting-models



Revisiting deep learning for tabular data

Our paper from NeurIPS 21

$$\begin{aligned} \texttt{MLP}(x) &= \texttt{Linear}\left(\texttt{MLPBlock}\left(\ldots\left(\texttt{MLPBlock}(x)\right)\right)\right) \\ \texttt{MLPBlock}(x) &= \texttt{Dropout}(\texttt{ReLU}(\texttt{Linear}(x))) \end{aligned} \tag{1}$$

$$\begin{aligned} \texttt{ResNet}(x) &= \texttt{Prediction}\left(\texttt{ResNetBlock}\left(\ldots\left(\texttt{ResNetBlock}\left(\texttt{Linear}(x)\right)\right)\right)\right) \\ \texttt{ResNetBlock}(x) &= x + \texttt{Dropout}(\texttt{Linear}(\texttt{Dropout}(\texttt{ReLU}(\texttt{Linear}(\texttt{BatchNorm}(x)))))) \\ \texttt{Prediction}(x) &= \texttt{Linear}\left(\texttt{ReLU}\left(\texttt{BatchNorm}\left(x\right)\right)\right) \end{aligned} \tag{2}$$

Revisiting deep learning for tabular data

Our paper from NeurIPS 21

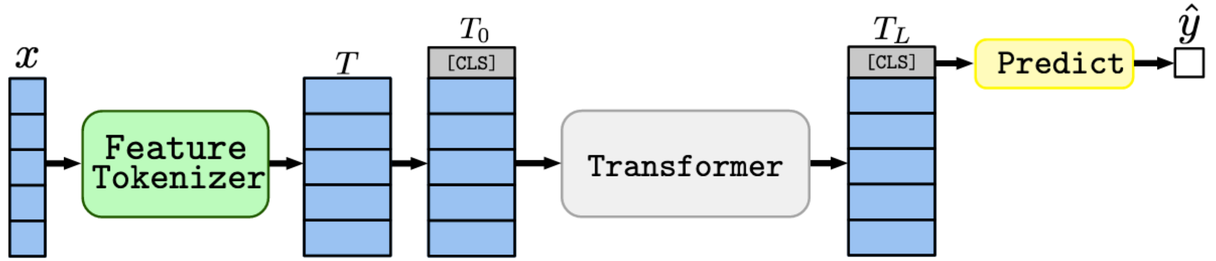
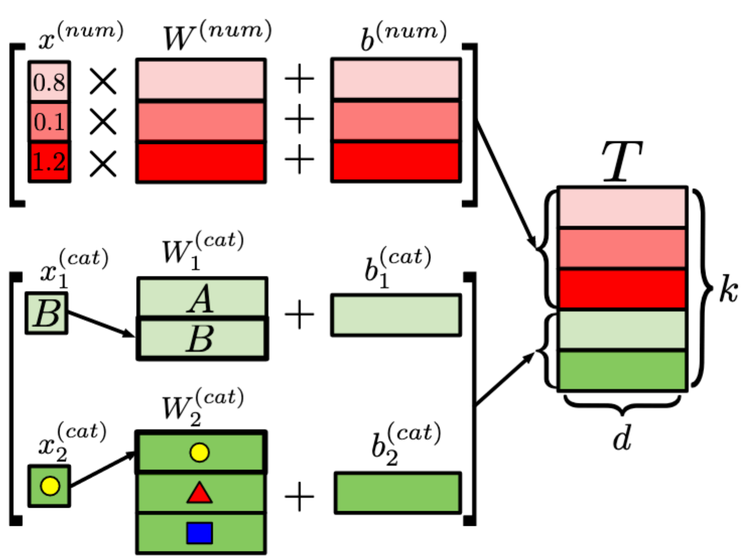
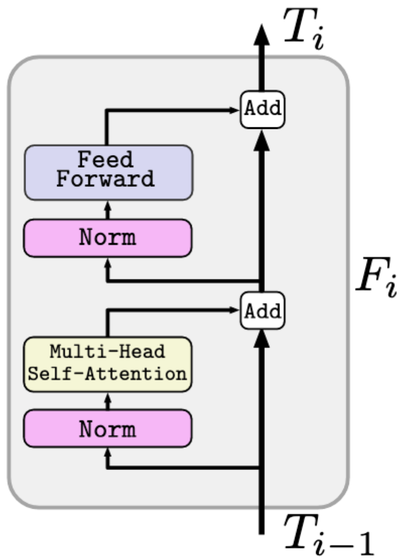


Figure 1: The FT-Transformer architecture. Firstly, Feature Tokenizer transforms features to embeddings. The embeddings are then processed by the Transformer module and the final representation of the [CLS] token is used for prediction.





Revisiting deep learning for tabular data

Our paper from NeurIPS 21

Adult dataset 👀

	CA ↓	AD ↑	HE ↑	JA ↑	HI ↑	AL ↑	EP ↑	YE ↓	CO ↑	YA ↓	MI ↓	rank (std)
TabNet	0.510	0.850	0.378	0.723	0.719	0.954	0.8896	8.909	0.957	0.823	0.751	7.5 (2.0)
SNN	0.493	0.854	0.373	0.719	0.722	0.954	0.8975	8.895	0.961	0.761	0.751	6.4 (1.4)
AutoInt	0.474	**0.859**	0.372	0.721	0.725	0.945	0.8949	8.882	0.934	0.768	0.750	5.7 (2.3)
GrowNet	0.487	**0.857**	–	–	0.722	–	0.8970	8.827	–	0.765	0.751	5.7 (2.2)
MLP	0.499	0.852	0.383	0.719	0.723	0.954	0.8977	8.853	0.962	0.757	0.747	4.8 (1.9)
DCN2	0.484	0.853	0.385	0.716	0.723	0.955	0.8977	8.890	0.965	0.757	0.749	4.7 (2.0)
NODE	0.464	**0.858**	0.359	0.727	0.726	0.918	0.8958	**8.784**	0.958	**0.753**	**0.745**	3.9 (2.8)
ResNet	0.486	0.854	**0.396**	0.728	0.727	**0.963**	0.8969	8.846	0.964	0.757	0.748	3.3 (1.8)
FT-T	**0.459**	**0.859**	0.391	**0.732**	**0.729**	0.960	**0.8982**	8.855	**0.970**	0.756	0.746	1.8 (1.2)

Revisiting deep learning for tabular data

Our paper from NeurIPS 21

Adult dataset 👀

	CA ↓	AD ↑	HE ↑	JA ↑	HI ↑	AL ↑	EP ↑	YE ↓	CO ↑	YA ↓	MI ↓
Default hyperparameters											
XGBoost	0.462	**0.874**	0.348	0.711	0.717	0.924	0.8799	9.192	0.964	0.761	0.751
CatBoost	**0.428**	0.873	0.386	0.724	0.728	0.948	0.8893	8.885	0.910	0.749	0.744
FT-Transformer	0.454	0.860	**0.395**	**0.734**	**0.731**	**0.966**	**0.8969**	**8.727**	**0.973**	**0.747**	**0.742**
Tuned hyperparameters											
XGBoost	0.431	0.872	0.377	0.724	0.728	–	0.8861	8.819	0.969	**0.732**	0.742
CatBoost	**0.423**	**0.874**	0.388	0.727	0.729	–	0.8898	8.837	0.968	0.740	**0.741**
ResNet	0.478	0.857	0.398	0.734	0.731	0.966	0.8976	8.770	0.967	0.751	0.745
FT-Transformer	0.448	0.860	**0.398**	**0.739**	**0.731**	**0.967**	**0.8984**	**8.751**	**0.973**	0.747	0.743

Concurrent work

Tabular DL go brrr (not really)

- Well tuned MLPs outperform GBDTs https://arxiv.org/abs/2106.11189
- SAINT: transformers for tabular data problems https://arxiv.org/abs/2106.01342

Timeline

2016/2017

Pre DL
Linear models all the way
XGBoost and CatBoost came out

2021

DL
Tabular DL papers
Our paper

Today+

Post DL (Tabular DL)
Our paper
Future work

Embeddings for numerical features

Our work

$$z_i = f_i\left(x_i^{(num)}\right) \in \mathbb{R}^{d_i}$$

$$\texttt{MLP}(z_1, \ldots, z_k) = \texttt{MLP}\left(\texttt{concat}[z_1, \ldots, z_k]\right)$$

$$\texttt{concat}[z_1, \ldots, z_k] \in \mathbb{R}^{d_1 + \ldots + d_k}$$

	GE ↑	CH ↑	EY ↑	CA ↓	HO ↓	AD ↑	OT ↑	HI ↑	FB ↓	SA ↑	CO ↑	MI ↓
MLP	0.632	0.856	0.615	0.495	3.204	0.854	0.818	0.720	5.686	0.912	**0.964**	0.747
MLP-L	**0.639**	**0.861**	0.635	0.475	3.123	0.856	**0.820**	0.723	5.684	0.916	0.963	0.748
MLP-LR	**0.642**	**0.860**	**0.660**	**0.471**	**3.084**	0.857	**0.819**	**0.726**	**5.625**	0.923	0.963	**0.746**
MLP-AutoDis	**0.649**	0.857	0.634	0.474	3.165	**0.859**	0.807	**0.725**	**5.670**	**0.924**	**0.963**	–

Embeddings for numerical features

Our work

$$z_i = f_i\left(x_i^{(num)}\right) \in \mathbb{R}^{d_i}$$

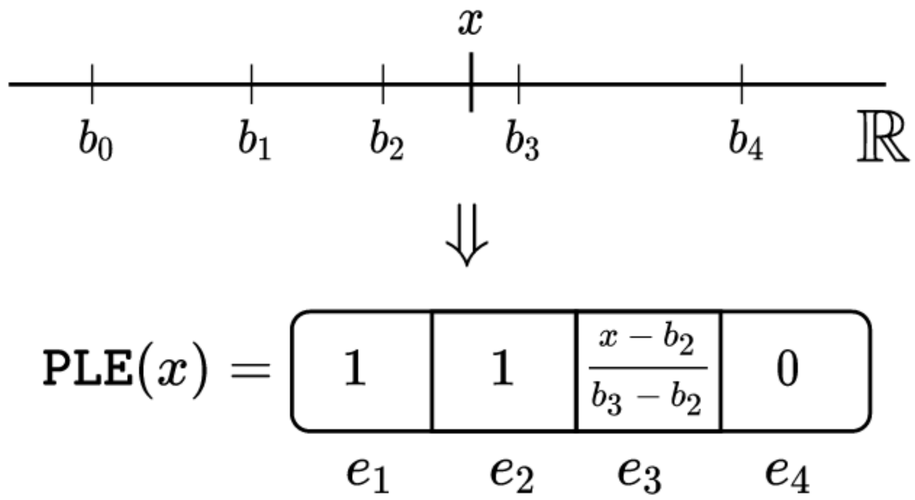
$$\texttt{MLP}(z_1, \ldots, z_k) = \texttt{MLP}\left(\texttt{concat}[z_1, \ldots, z_k]\right)$$

$$\texttt{concat}[z_1, \ldots, z_k] \in \mathbb{R}^{d_1 + \ldots + d_k}$$

	GE ↑	CH ↑	EY ↑	CA ↓	HO ↓	AD ↑	OT ↑	HI ↑	FB ↓	SA ↑	CO ↑	MI ↓
MLP	0.632	0.856	0.615	0.495	3.204	0.854	0.818	0.720	5.686	0.912	**0.964**	0.747
MLP-L	**0.639**	**0.861**	0.635	0.475	3.123	0.856	**0.820**	0.723	5.684	0.916	0.963	0.748
MLP-LR	**0.642**	**0.860**	**0.660**	**0.471**	**3.084**	0.857	**0.819**	**0.726**	**5.625**	0.923	0.963	**0.746**
MLP-AutoDis	**0.649**	0.857	0.634	0.474	3.165	**0.859**	0.807	**0.725**	**5.670**	**0.924**	**0.963**	–

Embeddings for numerical features

Our work



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

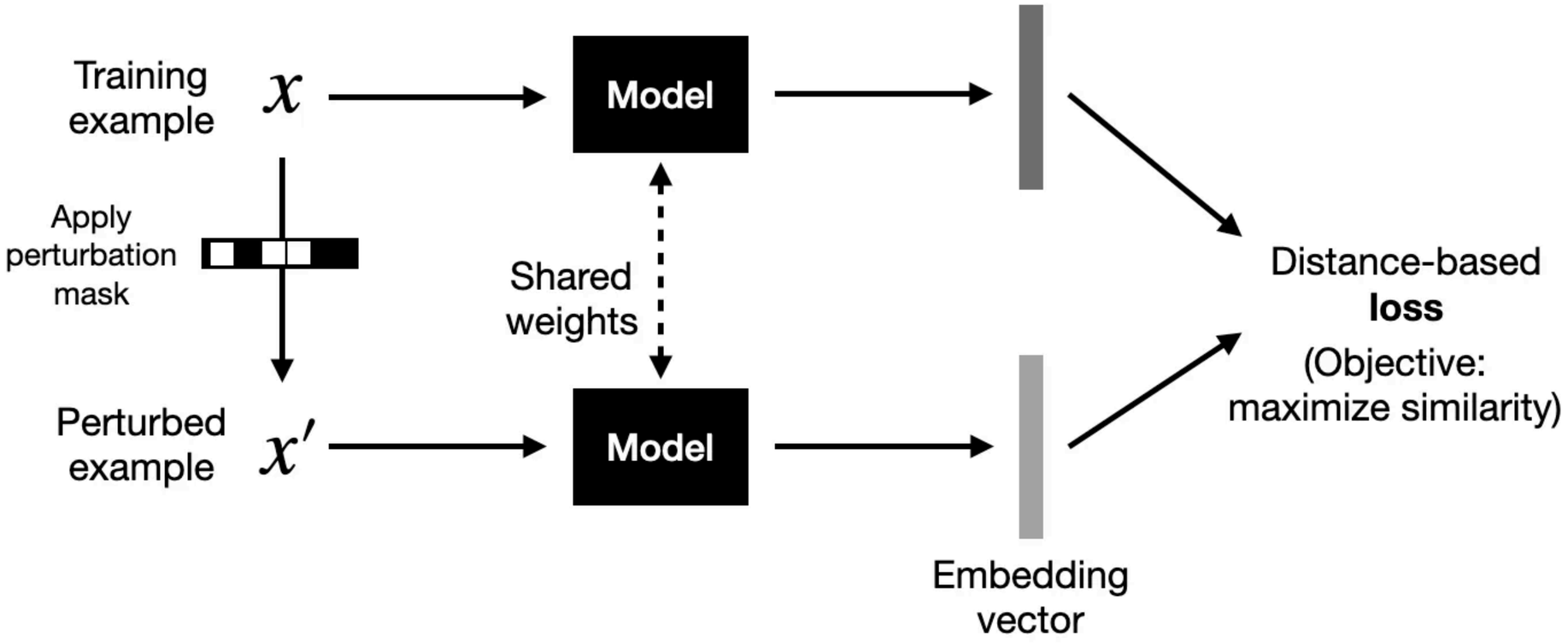
Embeddings for numerical features

Very sorry :.(

	GE ↑	CH ↑	EY ↑	CA ↓	HO ↓	AD ↑	OT ↑	HI ↑	FB ↓	SA ↑	CO ↑	MI ↓	Avg. Rank
CatBoost	0.692	0.861	0.757	0.430	3.093	0.873	0.825	0.727	5.226	0.924	0.967	**0.741**	6.8 ± 4.9
XGBoost	0.683	0.859	0.738	0.434	3.152	**0.875**	0.827	0.726	5.338	0.919	0.969	0.742	9.0 ± 5.7
MLP	0.665	0.856	0.637	0.486	3.109	0.856	0.822	0.727	5.616	0.913	0.968	0.746	15.6 ± 2.4
MLP-LR	0.679	0.861	0.694	0.463	3.012	0.859	0.826	0.731	5.477	0.924	0.972	0.744	10.2 ± 4.4
MLP-Q-LR	0.682	0.859	0.732	0.433	3.080	0.867	0.818	0.724	**5.144**	0.924	0.974	0.745	10.7 ± 4.6
MLP-T-LR	0.673	0.861	0.729	0.435	3.099	0.870	0.821	0.727	5.409	0.924	0.973	0.746	10.3 ± 3.8
MLP-PLR	**0.700**	0.858	0.968	0.453	**2.975**	0.874	**0.830**	**0.734**	5.388	**0.924**	0.975	0.743	4.9 ± 4.8
ResNet	0.690	0.861	0.667	0.483	3.081	0.856	0.821	0.734	5.482	0.918	0.968	0.745	12.1 ± 4.7
ResNet-LR	0.672	0.862	0.735	0.450	2.992	0.859	0.822	0.733	5.415	0.923	0.971	0.743	9.8 ± 4.3
ResNet-Q-LR	0.674	0.859	0.794	0.427	3.066	0.868	0.815	0.729	5.309	0.923	0.976	0.746	9.2 ± 4.8
ResNet-T-LR	0.683	0.862	0.817	**0.425**	3.030	0.872	0.822	0.731	5.471	0.923	0.975	0.744	7.8 ± 3.6
ResNet-PLR	0.691	0.861	0.925	0.443	3.040	**0.874**	0.825	0.734	5.400	0.924	0.975	0.743	5.2 ± 2.3
Transformer-L	0.668	0.861	0.769	0.455	3.188	0.860	0.824	0.727	5.434	0.924	0.973	0.743	10.6 ± 3.3
Transformer-LR	0.666	0.861	0.776	0.446	3.193	0.861	0.824	0.733	5.430	0.924	0.973	0.743	9.4 ± 4.1
Transformer-Q-LR	0.690	0.857	0.842	**0.425**	3.143	0.868	0.818	0.726	5.471	**0.924**	0.975	0.744	8.5 ± 5.5
Transformer-T-LR	0.686	0.862	0.833	**0.423**	3.149	0.871	0.823	0.733	5.515	0.924	**0.976**	0.744	7.2 ± 4.6
Transformer-PLR	0.686	**0.864**	**0.977**	0.449	3.091	0.873	0.823	0.734	5.581	**0.924**	0.975	0.743	6.0 ± 4.5

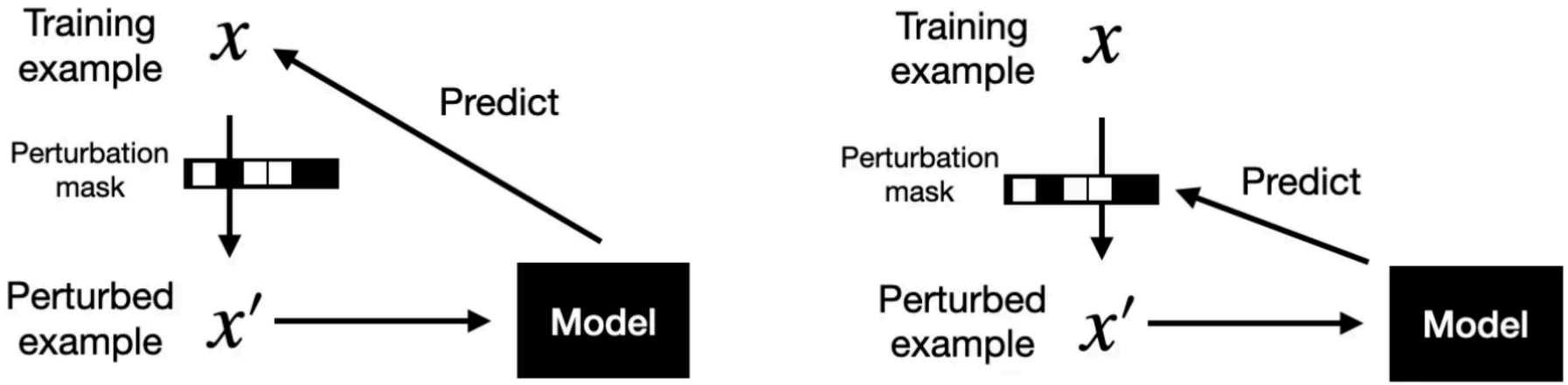
Pretraining in tabular deep-learning

https://magazine.sebastianraschka.com/p/ahead-of-ai-3-ainnouncements



Pretraining in tabular deep-learning

https://magazine.sebastianraschka.com/p/ahead-of-ai-3-ainnouncements



A simplified depiction of the two self-prediction processes. Left: the model predicts the original training example from the perturbed example. Right: the model predicts that perturbation mask given the perturbed example.

Pretraining in tabular deep-learning

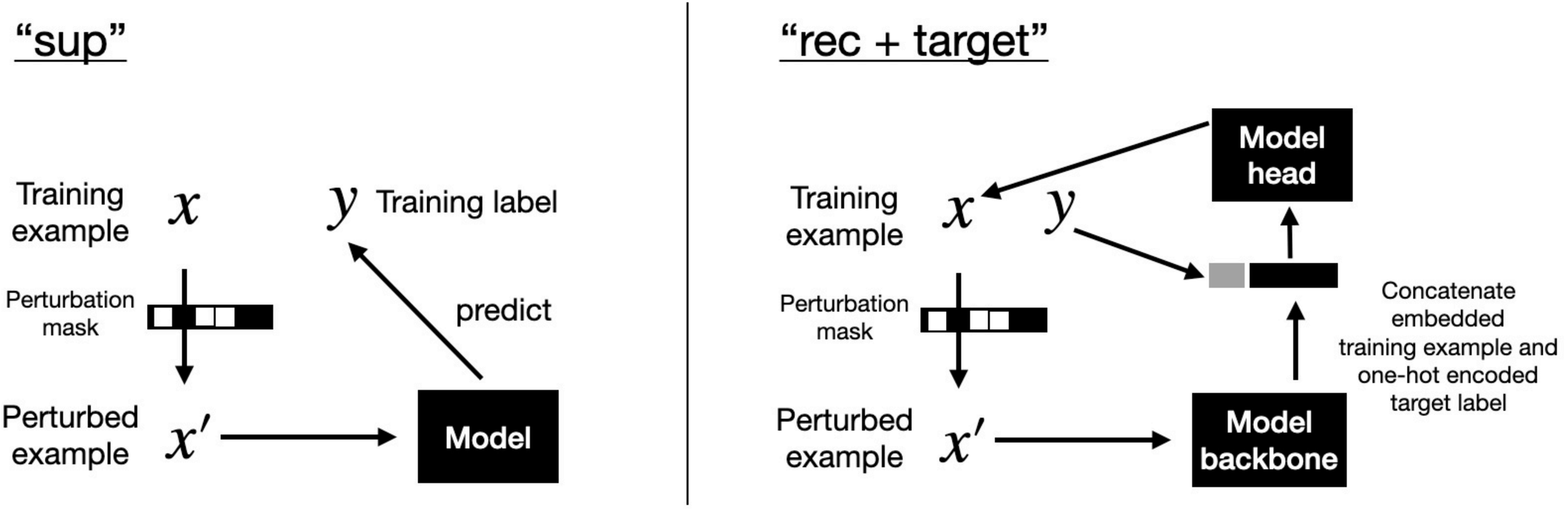
https://magazine.sebastianraschka.com/p/ahead-of-ai-3-ainnouncements

Table 2: Results for pretraining deep models with different objectives. We report metrics averaged over 15 seeds, bold entries correspond to results that are statistically significantly better (we use Tukey HSD test). The comparisons are separate for different models. ↑ corresponds to accuracy and ROC-AUC metrics, ↓ corresponds to RMSE and log-loss for OT. "no pretraining" stands for the supervised baseline, initialized with random weights

	GE ↑	CH ↑	CA ↓	HO ↓	OT ↓	HI ↑	FB ↓	AD ↑	WE ↓	CO ↑	MI ↓
MLP											
no pretraining	0.635	0.849	0.506	3.156	0.479	0.801	5.737	0.908	1.909	0.963	0.749
contrastive	0.672	**0.855**	0.455	**3.056**	0.469	**0.813**	5.697	0.910	1.881	0.960	0.748
rec	0.662	0.853	**0.445**	**3.044**	**0.466**	0.805	**5.641**	**0.910**	**1.875**	**0.965**	**0.746**
mask	**0.691**	**0.857**	0.454	3.113	0.472	**0.814**	**5.681**	**0.912**	1.883	0.964	0.748
MLP-PLR											
no pretraining	0.668	**0.858**	0.469	**3.008**	0.483	0.809	**5.608**	0.926	1.890	0.969	0.746
rec	0.667	0.852	**0.439**	**3.031**	**0.472**	0.808	**5.571**	**0.926**	**1.877**	**0.971**	**0.745**
mask	**0.685**	**0.863**	**0.434**	**3.007**	0.477	**0.818**	**5.586**	**0.927**	1.911	**0.970**	0.748
MLP-T-LR											
no pretraining	0.634	**0.866**	0.444	3.113	0.482	0.805	5.520	0.925	1.897	0.968	0.749
rec	**0.652**	0.857	**0.424**	3.109	**0.472**	0.808	**5.363**	0.924	**1.861**	**0.969**	**0.746**
mask	**0.654**	**0.868**	**0.424**	**3.045**	**0.472**	**0.818**	5.544	**0.926**	1.916	**0.969**	0.748

Pretraining in tabular deep-learning

https://magazine.sebastianraschka.com/p/ahead-of-ai-3-ainnouncements



Pretraining in tabular deep-learning

https://magazine.sebastianraschka.com/p/ahead-of-ai-3-ainnouncements

2 datasets where gradient boosting outperforms the deep learning-based methods

Table 4: Comparison of pretrained models to GBDT. Notation follows Table 2. Results represent ensembles of models. Bold entries correspond to the overall statistically significant best entries.

	GE ↑	CH ↑	CA ↓	HO ↓	OT ↓	HI ↑	FB ↓	AD ↑	WE ↓	CO ↑	MI ↓	Avg. Rank
CatBoost	0.692	**0.864**	0.430	3.093	0.450	0.807	5.226	0.928	1.801	0.967	**0.741**	2.6 ± 1.4
XGBoost	0.683	0.860	0.434	3.152	0.454	0.805	5.338	0.927	**1.782**	0.969	**0.742**	3.0 ± 1.5
MLP												
no pretraining	0.656	0.852	0.482	3.055	0.467	0.805	5.666	0.910	1.850	0.968	0.747	4.8 ± 1.1
mask + target	0.709	0.860	0.414	2.949	0.457	**0.828**	5.551	0.916	1.809	0.969	0.746	2.8 ± 1.2
rec + sup	0.709	0.859	0.419	2.951	**0.442**	0.817	5.531	0.913	1.801	0.973	0.745	2.5 ± 1.2
MLP-P-LR												
no pretraining	0.695	**0.864**	0.454	2.953	0.470	0.814	5.324	0.928	1.835	**0.974**	0.744	2.6 ± 1.2
mask + target	**0.719**	**0.866**	**0.407**	2.952	0.458	**0.828**	5.373	**0.930**	1.849	0.973	0.745	2.1 ± 1.2
rec + sup	**0.737**	0.862	0.424	2.964	0.449	0.811	**5.124**	**0.929**	1.813	**0.974**	0.744	2.0 ± 1.0
MLP-T-LR												
no pretraining	0.662	**0.868**	0.437	3.028	0.472	0.808	5.424	0.927	1.850	0.972	0.747	3.7 ± 1.1
mask + target	0.673	**0.868**	**0.410**	**2.894**	0.460	**0.827**	5.458	**0.930**	1.849	0.972	0.746	2.4 ± 1.4
rec + sup	0.705	**0.866**	0.425	3.057	**0.444**	0.814	5.422	0.927	1.811	**0.974**	0.746	2.5 ± 1.1

Diffusion models 🤯 and tabular data generation

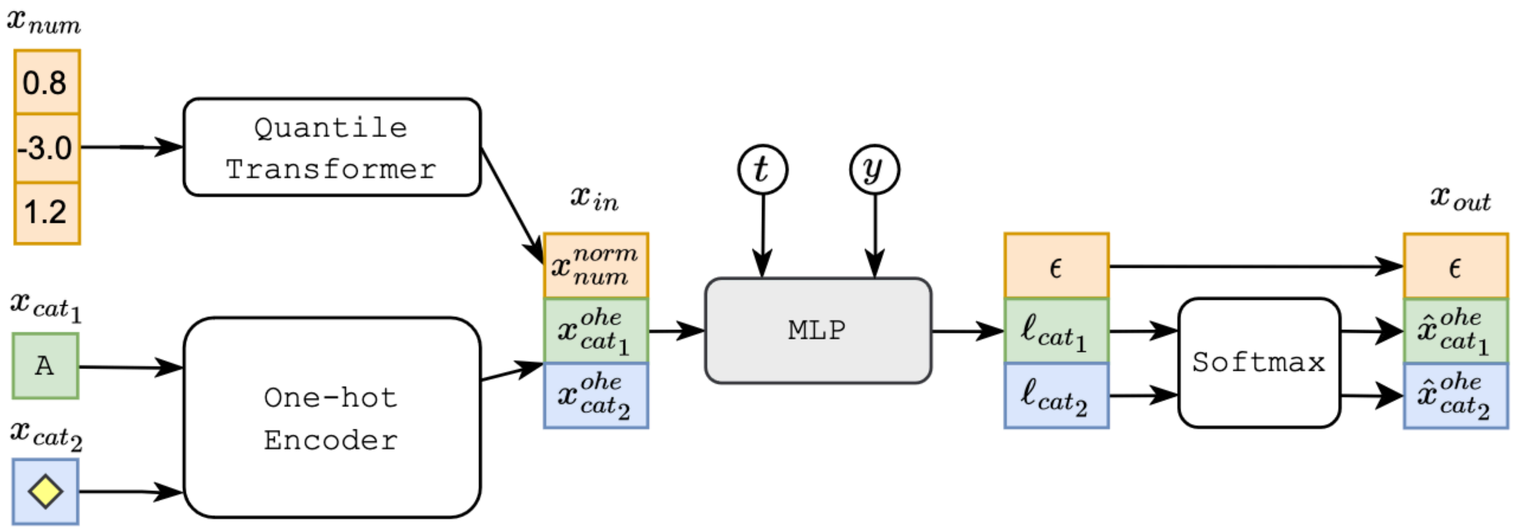


Figure 1: TabDDPM scheme for classification problems; t, y and ℓ denote a diffusion timestep, a class label, and logits, respectively.

Diffusion models 🤯 and tabular data generation

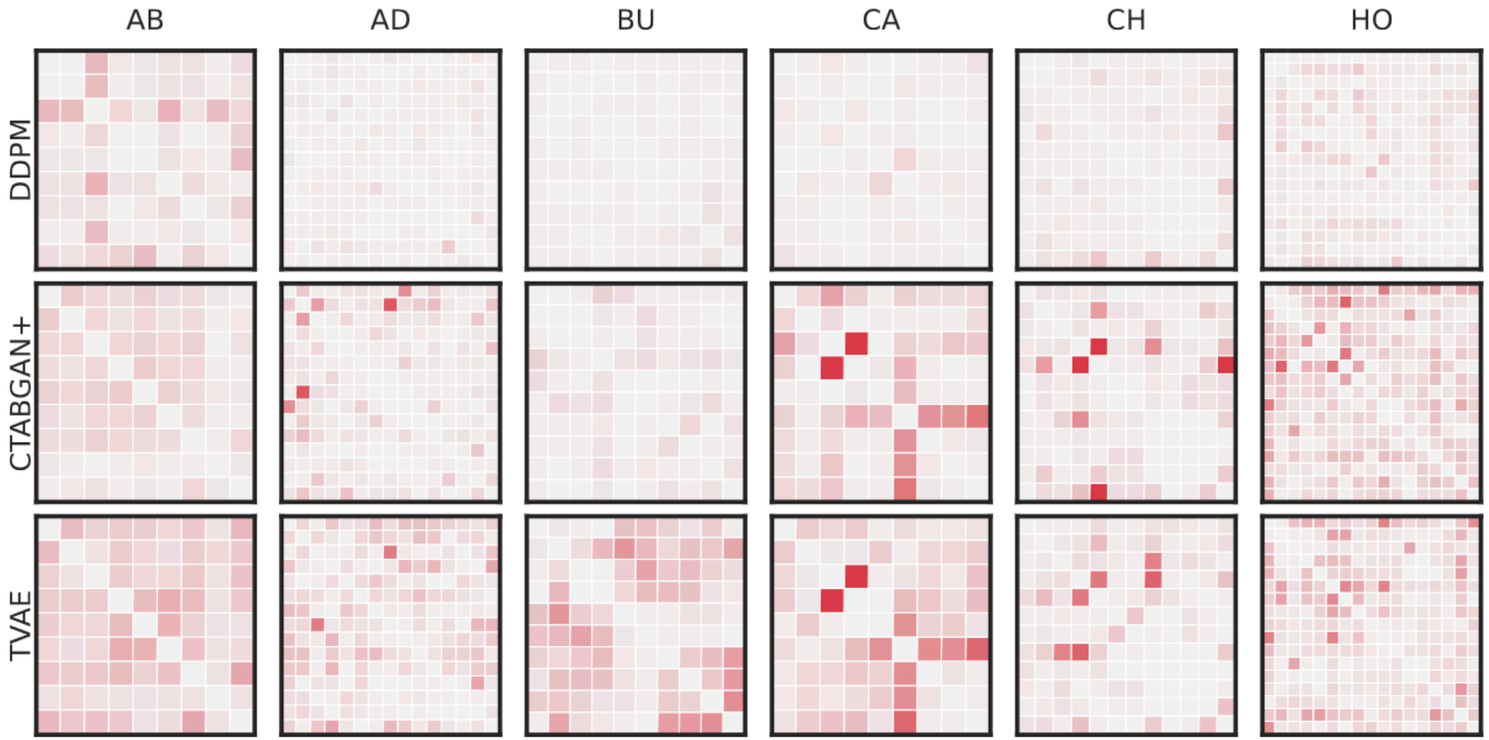


Figure 3: The absolute difference between correlation matrices computed on real and synthetic datasets. A more intensive red colour indicates a higher difference between the real and synthetic correlation values.

Diffusion models 🤯 and tabular data generation

	AB ($R2$)	AD ($F1$)	BU ($F1$)	CA ($R2$)	CAR ($F1$)	CH ($F1$)	DE ($F1$)	DI ($F1$)
TVAE	$0.238 \pm .012$	$0.742 \pm .001$	$0.779 \pm .004$	-13.0 ± 1.51	$0.693 \pm .002$	$0.684 \pm .003$	$0.643 \pm .003$	$0.712 \pm .010$
CTABGAN	–	$0.737 \pm .007$	$0.786 \pm .008$	–	$0.684 \pm .003$	$0.636 \pm .010$	$0.614 \pm .007$	$0.655 \pm .015$
CTABGAN+	$0.316 \pm .024$	$0.730 \pm .007$	$0.837 \pm .006$	$-7.59 \pm .645$	$\mathbf{0.708 \pm .002}$	$0.650 \pm .008$	$0.648 \pm .008$	$\mathbf{0.727 \pm .023}$
SMOTE	$\mathbf{0.400 \pm .009}$	$0.750 \pm .004$	$0.842 \pm .003$	$0.667 \pm .006$	$0.693 \pm .001$	$0.690 \pm .003$	$0.649 \pm .003$	$0.677 \pm .013$
TabDDPM	$\mathbf{0.392 \pm .009}$	$\mathbf{0.758 \pm .005}$	$\mathbf{0.851 \pm .003}$	$\mathbf{0.695 \pm .002}$	$0.696 \pm .001$	$\mathbf{0.693 \pm .003}$	$\mathbf{0.659 \pm .003}$	$0.675 \pm .011$
Real	$0.423 \pm .009$	$0.750 \pm .006$	$0.845 \pm .004$	$0.663 \pm .002$	$0.683 \pm .002$	$0.679 \pm .003$	$0.648 \pm .003$	$0.699 \pm .012$

	FB ($R2$)	GE ($F1$)	HI ($F1$)	HO ($R2$)	IN ($R2$)	KI ($R2$)	MI ($F1$)	WI ($F1$)
TVAE	$\ll 0$	$0.372 \pm .006$	$0.590 \pm .004$	$0.174 \pm .012$	$0.470 \pm .024$	$0.666 \pm .006$	$\mathbf{0.880 \pm .002}$	$0.497 \pm .001$
CTABGAN	–	$0.339 \pm .009$	$0.539 \pm .006$	–	–	–	$0.856 \pm .003$	$0.656 \pm .011$
CTABGAN+	$\ll 0$	$0.373 \pm .009$	$0.598 \pm .004$	$0.222 \pm .042$	$0.669 \pm .018$	$0.197 \pm .051$	$0.867 \pm .002$	$0.653 \pm .027$
SMOTE	$\mathbf{0.651 \pm .002}$	$\mathbf{0.478 \pm .005}$	$0.664 \pm .003$	$0.394 \pm .006$	$0.709 \pm .008$	$\mathbf{0.751 \pm .005}$	$0.860 \pm .001$	$\mathbf{0.793 \pm .004}$
TabDDPM	$0.527 \pm .005$	$0.462 \pm .005$	$\mathbf{0.670 \pm .002}$	$\mathbf{0.426 \pm .007}$	$\mathbf{0.734 \pm .007}$	$0.611 \pm .013$	$0.850 \pm .004$	$\mathbf{0.792 \pm .004}$
Real	$0.645 \pm .005$	$0.431 \pm .005$	$0.663 \pm .002$	$0.415 \pm .007$	$0.708 \pm .007$	$0.768 \pm .013$	$0.850 \pm .004$	$0.684 \pm .004$

Table 3: The values of machine learning efficiency computed with regards to five weak classification/regression models. Negative scores denote negative R2, which means that performance is worse than an optimal constant prediction.

Diffusion models 🤯 and tabular data generation

	AB ($R2$)	AD ($F1$)	BU ($F1$)	CA ($R2$)	CAR ($F1$)	CH ($F1$)	DE ($F1$)	DI ($F1$)
TVAE	0.433±.008	0.781±.002	0.864±.005	0.752±.001	0.717±.001	0.732±.006	0.656±.007	**0.714±.039**
CTABGAN	–	0.783±.002	0.855±.005	–	0.717±.001	0.688±.006	0.644±.011	**0.731±.022**
CTABGAN+	0.467±.004	0.772±.003	0.884±.005	0.525±.004	0.733±.001	0.702±.012	0.686±.004	**0.734±.020**
SMOTE	**0.549±.005**	0.791±.002	0.891±.003	**0.840±.001**	0.732±.001	0.743±.005	**0.693±.003**	0.683±.037
TabDDPM	**0.550±.010**	**0.795±.001**	**0.906±.003**	0.836±.002	**0.737±.001**	**0.755±.006**	**0.691±.004**	**0.740±.020**
Real	0.556±.004	0.815±.002	0.906±.002	0.857±.001	0.738±.001	0.740±.009	0.688±.003	0.785±.013

	FB ($R2$)	GE ($F1$)	HI ($F1$)	HO ($R2$)	IN ($R2$)	KI ($R2$)	MI ($F1$)	WI ($F1$)
TVAE	0.685±.003	0.434±.006	0.638±.003	0.493±.006	0.784±.010	0.824±.003	0.912±.001	0.501±.012
CTABGAN	–	0.392±.006	0.575±.004	–	–	–	0.889±.002	**0.906±.019**
CTABGAN+	0.509±.011	0.406±.009	0.664±.002	0.504±.005	0.797±.005	0.444±.014	0.892±.002	0.798±.021
SMOTE	**0.803±.002**	**0.658±.007**	**0.722±.001**	0.662±.004	**0.812±.002**	**0.842±.004**	0.932±.001	**0.913±.007**
TabDDPM	0.713±.002	0.597±.006	**0.722±.001**	**0.677±.010**	0.809±.002	**0.833±.014**	**0.936±.001**	**0.904±.009**
Real	0.837±.001	0.636±.007	0.724±.001	0.662±.003	0.814±.001	0.907±.002	0.934±.000	0.898±.006

Table 4: The values of machine learning efficiency computed with regards to the state-of-the-art tuned CatBoost model.

The end

Let’s talk

(If you think you might like doing research, apply for ML residency here:

https://yandex.com/jobs/vacancies/yandex-research-ml-residency-1740)

my contacts: @puhsuuu everywhere

Papers mentioned

- https://arxiv.org/abs/2106.11959 - Revisiting models
- https://arxiv.org/abs/2203.05556 - Numerical feature embeddings
- https://arxiv.org/abs/2207.03208 - Pre-training
- https://arxiv.org/abs/2209.15421 - Generating tabular data with TabDDPM