A Broader impact of this work

Tabular data is the most common data format used in practical data analysis including healthcare, finance, business, advertising, manufacturing, etc. Regardless of its importance, much more effort was made to play deep learning with other domains like vision, language, and audio. As a result, non-deep algorithms especially tree-based still dominate tabular data analysis. This paper opens the door to leverage the power of transfer learning in the tabular domain because `TransTab` is capable of dealing with variable-column input tables. This property also alleviates the workload required for data preprocessing because `TransTab` permits missing features and make good predictions based on the remaining features. This advance is expected to bring tremendous savings of time and money from the data engineering which often takes up 80% of efforts for data science projects [72]. It also sheds light on developing foundation models for the tabular domain because of the success of pretraining on scale.

The potential negative effect would be that it requires more effort on solving data privacy issues because `TransTab` works better when features are represented by descriptive texts compared with discretized indices. However, this can be alleviated by mapping features to machine-readable tokens in advance by a private codebook. Besides, `TransTab` needs more resources than simple models like MLP and Trees. The main cause is the use of full attention in multihead attention modules and the tokenization in the featurizing process. The former problem can be alleviated by replacing transformers with MLP-based blocks like gated attention units. The latter can be circumvented by pre-tokenization before the model training.

B Baseline architecture and implementation

In this section, we introduce the implementation details of baselines in experiments.

- LR: Use the default setting of the package scikit-learn[4] except the max iterations is set 1000.
- XGBoost: We set the maximum number of estimators in $\{50, 100, 500\}$ and the max depth in $\{4, 6, 8\}$. Implemented based on the XGBoost package[5].
- MLP & SNN: We keep the same architecture for both except for their activations: three dense layers with hidden dimensions 256, 256, 1; dropout with rate of 0.1 is used. They are trained with batch size $\in \{16, 32, 64, 128\}$, learning rate $\in$ {5e-5, 1e-4, 5e-4, 1e-3}, and early stopping patience of 5 with 100 maximum epochs.
- TabNet: Use the official implementation with the default recommended parameters[6]. Trained with batch size $\in \{16, 32, 64, 128\}$, learning rate $\in$ {1e-4, 1e-3, 2e-2}, $n_a, n_d \in \{8, 16, 64, 128\}$, $\gamma \in \{1.3, 1.5, 1.8\}$, categorical embedding dimension $\in \{1, 8, 16\}$ and early stopping patience of 5 with 100 maximum epochs.
- DCN: Use the implementation by Deep-CTR[7]. The number of cross is 2; dropout rate for feed-forward component is 0.1; MLP part has two dense layers of dimension 256, 128; Trained with batch size $\in \{16, 32, 64, 128\}$, learning rate $\in$ {5e-5, 1e-4, 1e-3}, and early stopping patience of 10 in 100 maximum epochs.
- AutoInt: Use the implementation by Deep-CTR. Attention layer number is set 2; Attention head number is set 2; MLP part has two dense layers of dimension 256, 128; dropout deactivated; Trained with batch size $\in \{16, 32, 64, 128\}$, learning rate $\in$ {5e-5, 1e-4, 1e-3}, and early stopping patience of 10 in 100 maximum epochs.
- TabTransformer: Use the official implementation[8]. Feed-forward component has 128 dimension; 2 transformer layers are used; The number of heads of attention is $\in \{2, 4, 8\}$; Dropout rate is 0.1; ReLU activation is used; Trained with batch size $\in \{16, 32, 64, 128\}$, learning rate $\in$ {5e-5,1e-4,1e-3}, and early stopping patience of 10 in 100 maximum epochs.
- FT-Transformer: Use the official implementation[9]. Feed-forward component has 128 dimension; 2 transformer layers are used; The number of heads of attention is $\in \{2, 4, 8\}$; Dropout rate is 0.1;

[4] sklearn.linear_model.LogisticRegression
[5] xgboost.sklearn
[6] https://github.com/dreamquark-ai/tabnet
[7] https://github.com/shenweichen/DeepCTR-Torch
[8] https://github.com/lucidrains/tab-transformer-pytorch
[9] https://github.com/Yura52/rtdl

Table 7: Statistics of the used public datasets. All are binary classification tasks. Positive ratio means the ratio of data points belong the positive class. Source links are available at Table 13.

Name	N Datapoints	Categorical	Binary	Numerical	Positive ratio
credit-g (CG)	1000	11	2	7	0.7
credit-approval (CA)	690	6	3	6	0.56
dresses-sales (DS)	500	11	0	1	0.42
adult (AD)	48842	12	0	2	0.24
cylinder-bands (CB)	540	13	4	18	0.58
blastchar (BL)	7043	11	5	3	0.27
insurance-co (IO)	5822	2	0	83	0.06
1995-income (IC)	32561	8	0	6	0.24

Table 8: Test AUROC results on public datasets the under **supervised learning** setting. The dataset name is abbreviated referring to Table 7.

Methods	CG	CA	DS	AD	CB	BL	IO	IC	Rank(Std)
LR	0.720	0.836	0.557	0.851	0.748	0.801	0.769	0.860	9.88(1.90)
XGBoost	0.726	**0.895**	0.587	0.912	**0.892**	0.821	0.758	**0.925**	5.12(3.86)
MLP	0.643	0.832	0.568	0.904	0.613	0.832	0.779	0.893	9.25(2.07)
SNN	0.641	0.880	0.540	0.902	0.621	0.834	0.794	0.892	8.00(3.32)
TabNet	0.585	0.800	0.478	0.904	0.680	0.819	0.742	0.896	10.75(1.49)
DCN	0.739	0.870	**0.674**	0.913	0.848	0.840	0.768	0.915	4.12(1.69)
AutoInt	0.744	0.866	0.672	0.913	0.808	0.844	0.762	0.916	4.62(2.52)
TabTrans	0.718	0.860	0.648	**0.914**	0.855	0.820	0.794	0.882	6.50(3.12)
FT-Trans	0.739	0.859	0.657	0.913	0.862	0.841	0.793	0.915	3.94(1.35)
VIME	0.735	0.852	0.485	0.912	0.769	0.837	0.786	0.908	6.06(2.83)
SCARF	0.733	0.861	0.663	0.911	0.719	0.833	0.758	0.905	6.75(2.12)
`TransTab`	**0.768**	0.881	0.643	0.907	0.851	**0.845**	**0.822**	0.919	**3.00(1.93)**

ReLU activation is used; Trained with batch size $\in \{16, 32, 64\}$, learning rate $\in$ {5e-5,1e-4,1e-3}, and early stopping patience of 10 in 100 maximum epochs.

- VIME: We reproduce it by PyTorch [73] based on the original official implementation[10] where its encoders, mask estimators, decoders are all one dense layer with the hidden dimension same as the input features. During the pretraining phase, we train the model on all training data taking mask rate 0.3, batch size 128, learning rate 1e-4, and 10 epochs; during the fine-tuning phase, we add a classifier after the encoder with three dense layers of 100 dimension and ReLU activations. Trained with batch size $\in \{16, 32, 64, 128\}$, learning rate {5e-5,1e-4,1e-3}, and early stopping patience of 10 in 100 maximum epochs.
- SCARF: Since no official code is found, we reproduce it by PyTorch based on the descriptions in the original paper. A 4-layer encoder and a 2-layer decoder with ReLU activations are used. Hidden dimensions of their intermediate layers are all 256. During the pretraining phase, we train the model on all trainign data taking mask rate 0.5, InfoNCE loss [74] with learning rate 1e-3, batch size 128, and 10 epochs; during the fine-tuning phase, we add a classifier after the encoder with two dense layers with 256 hidden dimensions. Trained with batch size $\in \{16, 32, 64, 128\}$, learning rate $\in$ {5e-5,1e-4,1e-3}, and early stopping patience of 10 in 100 maximum epochs.

C Preprocessing of clinical trial datasets

The raw real-world de-identified patient-level clinical records are obtained from Project Data Sphere. All clinical trial data used in this paper are available under registration for the data platform. For

[10] https://github.com/jsyoon0823/VIME

Table 9: Test AUROC results on public datasets the under **feature incremental learning** setting. The dataset name is abbreviated referring to Table 7.

Methods	CG	CA	DS	AD	CB	BL	IO	IC	Rank(Std)
LR	0.670	0.773	0.475	0.832	0.727	0.806	0.655	0.825	8.88(2.80)
XGBoost	0.608	0.817	0.527	0.891	0.778	0.816	0.692	**0.898**	5.50(3.13)
MLP	0.586	0.676	0.516	0.890	0.631	0.825	0.626	0.885	9.50(2.07)
SNN	0.583	0.738	0.442	0.888	0.644	0.818	0.643	0.881	9.69(1.28)
TabNet	0.573	0.689	0.419	0.886	0.571	0.837	0.680	0.882	9.19(3.34)
DCN	0.674	0.835	0.578	0.893	0.778	0.840	0.660	0.891	3.38(2.01)
AutoInt	0.671	0.825	0.563	0.893	0.769	0.836	0.676	0.887	4.75(1.25)
TabTrans	0.653	0.732	0.584	0.856	0.784	0.792	0.674	0.828	7.62(3.78)
FT-Trans	0.662	0.824	0.626	0.892	0.768	0.840	0.645	0.889	4.81(2.59)
VIME	0.621	0.697	0.571	0.892	0.769	0.803	0.683	0.881	7.00(3.01)
SCARF	0.651	0.753	0.556	0.891	0.703	0.829	0.680	0.887	6.56(1.32)
`TransTab`	**0.741**	**0.879**	**0.665**	**0.894**	**0.791**	**0.841**	**0.739**	0.897	**1.12(0.35)**

Table 10: Test AUROC results on public datasets under **transfer learning** across tables.

Methods	CG		CA		DS		AD		CB		BL		IO		IC		Rank(Std)
	set1	set2	set1	set2	set1	set2	set1	set2	set1	set2	set1	set2	set1	set2	set1	set2	
LR	0.69	0.69	0.81	0.82	0.47	0.56	0.81	0.81	0.68	0.78	0.77	0.82	0.71	0.81	0.81	0.84	8.57(2.83)
XGB	0.72	0.71	0.85	0.87	0.46	0.63	0.88	0.89	**0.80**	0.81	0.76	0.82	0.65	0.74	**0.92**	**0.91**	5.57(3.62)
MLP	0.67	0.70	0.82	0.86	0.53	**0.67**	0.89	0.90	0.73	**0.82**	0.79	0.83	0.70	0.78	0.90	0.90	5.00(2.86)
SNN	0.66	0.63	0.85	0.83	0.54	0.42	0.87	0.88	0.57	0.54	0.77	0.82	0.69	0.78	0.87	0.88	8.67(2.92)
TabNet	0.60	0.47	0.66	0.68	0.54	0.53	0.87	0.88	0.58	0.62	0.75	0.83	0.62	0.71	0.88	0.89	9.87(2.92)
DCN	0.69	0.70	0.83	0.85	0.51	0.58	0.88	0.74	0.79	0.78	0.79	0.76	0.70	0.71	0.91	0.90	6.67(2.94)
AutoInt	0.70	0.70	0.82	0.86	0.49	0.55	0.88	0.74	0.77	0.79	**0.79**	0.76	0.71	0.72	0.91	0.90	6.03(3.23)
TabTrans	0.72	0.72	0.84	0.86	0.54	0.57	0.88	0.90	0.73	0.79	0.78	0.81	0.67	0.71	0.88	0.88	6.03(2.93)
FT-Trans	0.72	0.71	0.83	0.85	0.53	0.64	**0.89**	0.90	0.76	0.79	0.78	0.84	0.68	0.78	0.91	0.91	4.97(1.95)
VIME	0.59	0.70	0.79	0.76	0.45	0.53	0.88	0.90	0.65	0.81	0.58	0.83	0.67	0.70	0.90	0.90	8.83(3.24)
SCARF	0.69	0.72	0.82	0.85	**0.55**	0.64	0.88	0.89	0.77	0.73	0.78	0.83	0.71	0.75	0.90	0.89	5.47(2.42)
`TransTab`	**0.74**	**0.76**	**0.87**	**0.89**	0.55	0.66	0.88	**0.90**	0.80	0.80	0.79	**0.84**	**0.73**	**0.82**	0.91	0.91	**2.33(2.10)**

each trial, we manage to extract patient's baseline information as the features, including demographic information, medical history, medication history, lab test, vital signs, adverse events, etc. We draw the target labels from the survival analysis section where censoring is considered as "alive" and other events are tagged "mortality" so as to transform the datasets into binary prediction tasks.

D Establishment of subsets

For experiments in §3.2, §3.3, §3.4, we create subsets randomly with a fixed seed, respectively. That is, subsets vary across these experiments.

- Feature incremental learning. The columns are splitted into three distinct parts v_1, v_2, v_3. Set1 contains v_1, set2 contains v_1, v_2, and set3 has v_1, v_2, v_3. Three sets have the equal number of samples.
- Transfer learning. The columns are splitted into two parts v_1, v_2 where v_1 and v_2 have 50% of elements overlapped. Two sets have the equal number of samples.
- Zeroshot learning. The columns are splitted into three distinct parts v_1, v_2, v_3. Set1 contains v_1, set2 contains v_2, set3 contains v_3. Three sets have the equal number of samples.

Table 11: Test AUROC results on public datasets under **zero-shot learning** setting.

TransTab	CG	CA	DS	AD	CB	BL	IO	IC
Supervised	0.581	0.635	0.571	0.898	0.733	0.822	0.702	0.875
Transfer	0.719	0.758	0.561	0.900	0.854	0.831	0.761	0.880
Zero-shot	0.685	0.721	0.538	0.892	0.710	0.804	0.742	0.874

Table 12: Test AUROC on public datasets under the **across-table pretraining plus finetuning** setting. *Supervised*: baseline supervised model; *Transfer*: vanilla supervised transfer learning. Red shows the one worse than the baseline *Supervised*.

TransTab	CG	CA	DS	AD	CB	BL	IO	IC
Supervised	0.763	0.858	0.630	0.907	0.841	0.844	0.821	0.919
Transfer	**0.786**	**0.861**	**0.653**	0.907	0.819	0.843	0.813	0.918
Self-VPCL	0.777	0.837	0.626	0.907	0.819	0.843	**0.823**	0.919
VPCL	0.776	0.858	0.637	**0.907**	**0.862**	**0.844**	0.819	**0.919**

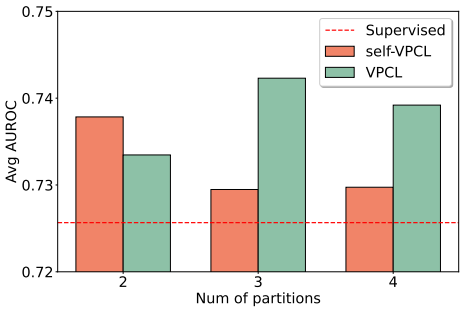

Figure 4: Analysis of the number of partitions for VPCL and self-VPCL on the **clinical trial datasets**.

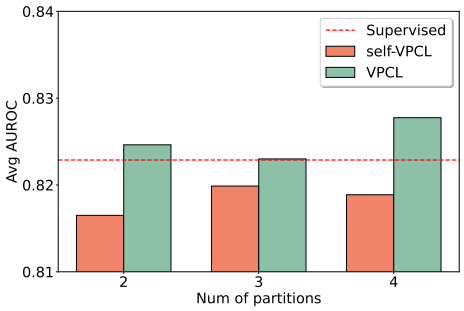

Figure 5: Analysis of the number of partitions for VPCL and self-VPCL on the **public datasets**

Table 13: Benchmark dataset links.

Dataset	URL
credit-g	https://www.openml.org/search?type=data&status=active&id=31
credit-approval	https://archive.ics.uci.edu/ml/datasets/credit+approval
dress-sales	https://www.openml.org/search?type=data&status=active&id=23381
adult	https://www.openml.org/search?type=data&status=active&id=1590
cylinder-bands	https://www.openml.org/search?type=data&status=active&id=6332
blastchar	https://www.kaggle.com/datasets/blastchar/telco-customer-churn
insurance-co	https://archive.ics.uci.edu/ml/datasets/Insurance+Company+Benchmark+%28COIL+2000%29
1995-income	https://www.kaggle.com/datasets/lodetomasi1995/income-classification

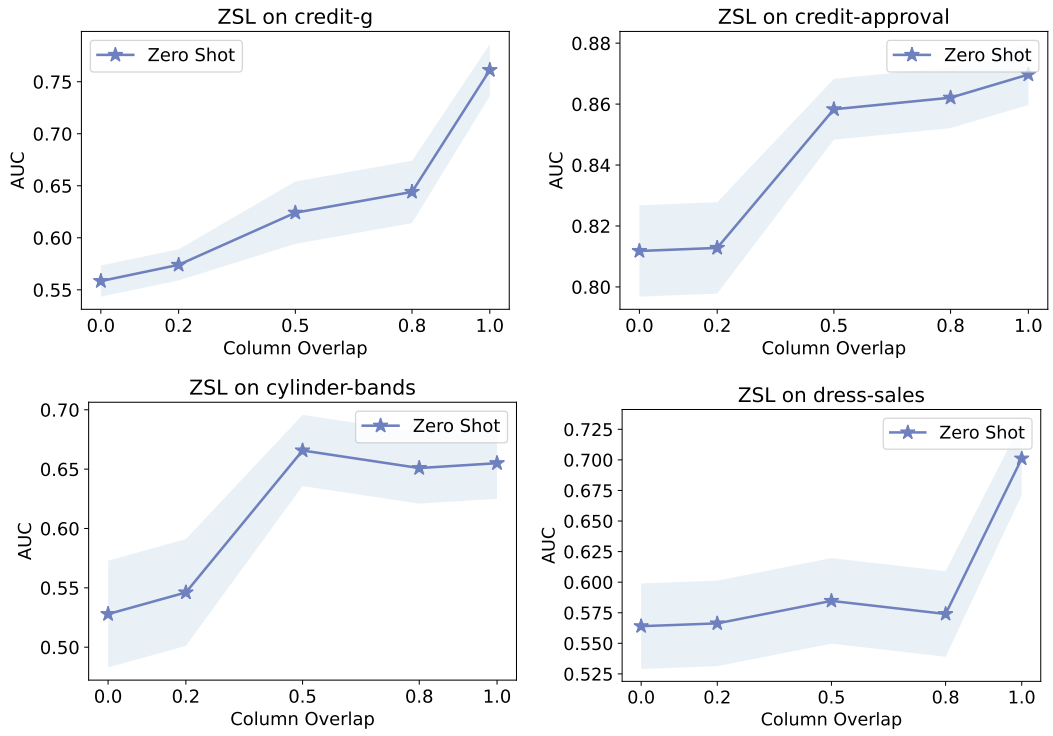

Figure 6: Evaluate how the overlap ratio of two tables' columns influences zero-shot learning (ZSL) performance of `TransTab`, on public data CG, CA, CB, and DS. x-axis: the ratio of test table columns exist in the training table (0: no test table column appears in training table; 1: all test table columns in training table); y-axis: the test AUC when making ZSL.