TabTransformer: Tabular Data Modeling Using Contextual Embeddings

Xin Huang,[1] **Ashish Khetan,** [1] **Milan Cvitkovic** [2] **Zohar Karnin** [1]

[1] Amazon AWS
[2] PostEra
xinxh@amazon.com, khetan@amazon.com, mwcvitkovic@gmail.com, zkarnin@amazon.com

Abstract

We propose TabTransformer, a novel deep tabular data modeling architecture for supervised and semi-supervised learning. The TabTransformer is built upon self-attention based Transformers. The Transformer layers transform the embeddings of categorical features into robust contextual embeddings to achieve higher prediction accuracy. Through extensive experiments on fifteen publicly available datasets, we show that the TabTransformer outperforms the state-of-the-art deep learning methods for tabular data by at least 1.0% on mean AUC, and matches the performance of tree-based ensemble models. Furthermore, we demonstrate that the contextual embeddings learned from TabTransformer are highly robust against both missing and noisy data features, and provide better interpretability. Lastly, for the semi-supervised setting we develop an unsupervised pre-training procedure to learn data-driven contextual embeddings, resulting in an average 2.1% AUC lift over the state-of-the-art methods.

1 Introduction

Tabular data is the most common data type in many real-world applications such as recommender systems (Cheng et al. 2016), online advertising (Song et al. 2019), and portfolio optimization (Ban, El Karoui, and Lim 2018). Many machine learning competitions such as Kaggle and KDD Cup are primarily designed to solve problems in tabular domain.

The state-of-the-art for modeling tabular data is tree-based ensemble methods such as the gradient boosted decision trees (GBDT) (Chen and Guestrin 2016; Prokhorenkova et al. 2018). This is in contrast to modeling image and text data where all the existing competitive models are based on deep learning (Sandler et al. 2018; Devlin et al. 2019). The tree-based ensemble models can achieve competitive prediction accuracy, are fast to train and easy to interpret. These benefits make them highly favourable among machine learning practitioners. However, the tree-based models have several limitations in comparison to deep learning models. (a) They are not suitable for continual training from streaming data, and do not allow efficient end-to-end learning of image/text encoders in presence of multi-modality along with tabular data. (b) In their basic form they are not suitable for state-of-the-art semi-supervised learning methods. This is due to the fact that the basic decision tree learner does not produce reliable probability estimation to its predictions (Tanha, Someren, and Afsarmanesh 2017). (c) The state-of-the-art deep learning methods (Devlin et al. 2019) to handle missing and noisy data features do not apply to them. Also, robustness of tree-based models has not been studied much in literature.

A classical and popular model that is trained using gradient descent and hence allows end-to-end learning of image/text encoders is multi-layer perceptron (MLP). The MLPs usually learn parametric embeddings to encode categorical data features. But due to their shallow architecture and context-free embeddings, they have the following limitations: (a) neither the model nor the learned embeddings are interpretable; (b) it is not robust against missing and noisy data (Section 3.2); (c) for semi-supervised learning, they do not achieve competitive performance (Section 3.4). Most importantly, MLPs do not match the performance of tree-based models such as GBDT on most of the datasets (Arik and Pfister 2019). To bridge this performance gap between MLP and GBDT, researchers have proposed various deep learning models (Song et al. 2019; Cheng et al. 2016; Arik and Pfister 2019; Guo et al. 2018). Although these deep learning models achieve comparable prediction accuracy, they do not address all the limitations of GBDT and MLP. Furthermore, their comparisons are done in a limited setting of a handful of datasets. In particular, in Section 3.3 we show that when compared to standard GBDT on a large collection of datasets, GBDT perform significantly better than these recent models.

In this paper, we propose TabTransformer to address the limitations of MLPs and existing deep learning models, while bridging the performance gap between MLP and GBDT. We establish performance gain of TabTransformer through extensive experiments on fifteen publicly available datasets.

The TabTransformer is built upon Transformers (Vaswani et al. 2017) to learn efficient contextual embeddings of categorical features. Different from tabular domain, the application of embeddings has been studied extensively in NLP. The use of embeddings to encode words in a dense low dimensional space is prevalent in natural language processing. Beginning from Word2Vec (Rong 2014) with the context-free word embeddings to BERT (Devlin et al. 2019) which pro-

Preprint.

vides the contextual word-token embeddings, embeddings have been widely studied and applied in practice in NLP. In comparison to context-free embeddings, the contextual embedding based models (Mikolov et al. 2011; Huang, Xu, and Yu 2015; Devlin et al. 2019) have achieved tremendous success. In particular, self-attention based Transformers (Vaswani et al. 2017) have become a standard component of NLP models to achieve state-of-the-art performance. The effectiveness and interpretability of contextual embeddings generated by Transformers have been also well studied (Coenen et al. 2019; Brunner et al. 2019).

Motivated by the successful applications of Transformers in NLP, we adapt them in tabular domain. In particular, TabTransformer applies a sequence of multi-head attention-based Transformer layers on parametric embeddings to transform them into contextual embeddings, bridging the performance gap between baseline MLP and GBDT models. We investigate the effectiveness and interpretability of the resulting contextual embeddings generated by the Transformers. We find that highly correlated features (including feature pairs in the same column and cross column) result in embedding vectors that are close together in Euclidean distance, whereas no such pattern exists in context-free embeddings learned in a baseline MLP model. We also study the robustness of the TabTransformer against random missing and noisy data. The contextual embeddings make them highly robust in comparison to MLPs.

Furthermore, many existing deep learning models for tabular data are designed for supervised learning scenario but few are for semi-supervised leanring (SSL). Unfortunately, the state-of-art SSL models developed in computer vision (Voulodimos et al. 2018; Kendall and Gal 2017) and NLP (Vaswani et al. 2017; Devlin et al. 2019) cannot be easily extended to tabular domain. Motivated by such challenges, we exploit pre-training methodologies from the language models and propose a semi-supervised learning approach for pre-training Transformers of our TabTransformer model using unlabeled data.

One of the key benefits of our proposed method for semi-supervised learning is the two independent training phases: a costly pre-training phase on unlabeled data and a lightweight fine-tuning phase on labeled data. This differs from many state-of-the-art semi-supervised methods (Chapelle, Scholkopf, and Zien 2009; Oliver et al. 2018; Stretcu et al. 2019) that require a single training job including both the labeled and unlabeled data. The separated training procedure benefits the scenario where the model needs to be pretrained once but fine-tuned multiple times for multiple target variables. This scenario is in fact quite common in the industrial setting as companies tend to have one large dataset (e.g. describing customers/products) and are interested in applying multiple analyses on this data. To summarize, we provide the following contributions:

1. We propose TabTransformer, an architecture that provides and exploits contextual embeddings of categorical features. We provide extensive empirical evidence showing TabTransformer is superior to both a baseline MLP and recent deep networks for tabular data while matching the performance of tree-based ensemble models (GBDT).
2. We investigate the resulting contextual embeddings and highlight their interpretability, contrasted to parametric context-free embeddings achieved by existing art.
3. We demonstrate the robustness of TabTransformer against noisy and missing data.
4. We provide and extensively study a two-phase pre-training then fine-tune procedure for tabular data, beating the state-of-the-art performance of semi-supervised learning methods.

2 The TabTransformer

The TabTransformer architecture comprises a column embedding layer, a stack of N Transformer layers, and a multi-layer perceptron. Each Transformer layer (Vaswani et al. 2017) consists of a multi-head self-attention layer followed by a position-wise feed-forward layer. The architecture of TabTransformer is shown below in Figure 1.

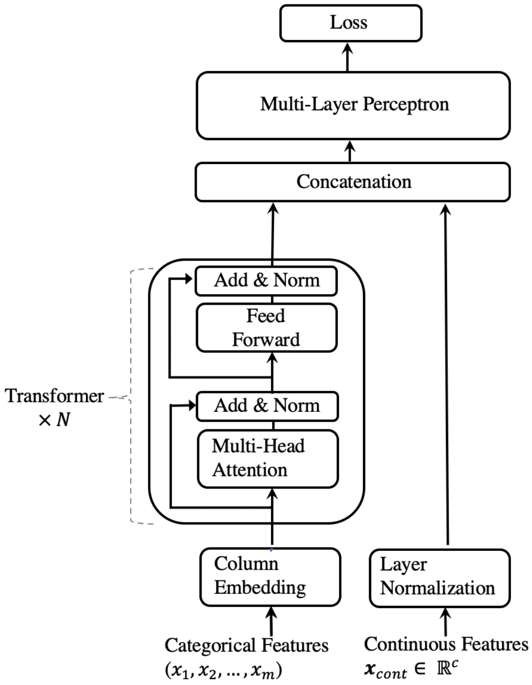

Figure 1: The architecture of TabTransformer.

Let $(\boldsymbol{x}, y)$ denote a feature-target pair, where $\boldsymbol{x} \equiv \{\boldsymbol{x}_{\text{cat}}, \boldsymbol{x}_{\text{cont}}\}$. The $\boldsymbol{x}_{\text{cat}}$ denotes all the categorical features and $\boldsymbol{x}_{\text{cont}} \in \mathbb{R}^c$ denotes all of the c continuous features. Let $\boldsymbol{x}_{\text{cat}} \equiv \{x_1, x_2, \cdots, x_m\}$ with each x_i being a categorical feature, for $i \in \{1, \cdots, m\}$.

We embed each of the x_i categorical features into a parametric embedding of dimension d using *Column embedding*, which is explained below in detail. Let $\boldsymbol{e}_{\phi_i}(x_i) \in \mathbb{R}^d$ for $i \in \{1, \cdots, m\}$ be the embedding of the x_i feature, and $\boldsymbol{E}_\phi(\boldsymbol{x}_{\text{cat}}) = \{\boldsymbol{e}_{\phi_1}(x_1), \cdots, \boldsymbol{e}_{\phi_m}(x_m)\}$ be the set of embeddings for all the categorical features.

Next, these parametric embeddings $\boldsymbol{E}_\phi(\boldsymbol{x}_{\text{cat}})$ are inputted to the first Transformer layer. The output of the first Transformer layer is inputted to the second layer Transformer, and so forth. Each parametric embedding is transformed into contextual embedding when outputted from the top layer Transformer, through successive aggregation of context from other embeddings. We denote the sequence of Transformer layers as a function f_θ. The function f_θ operates on parametric embeddings $\{\boldsymbol{e}_{\phi_1}(x_1), \cdots, \boldsymbol{e}_{\phi_m}(x_m)\}$ and returns the corresponding contextual embeddings $\{\boldsymbol{h}_1, \cdots, \boldsymbol{h}_m\}$ where $\boldsymbol{h}_i \in \mathbb{R}^d$ for $i \in \{1, \cdots, m\}$.

The contextual embeddings $\{\boldsymbol{h}_1, \cdots, \boldsymbol{h}_m\}$ are concatenated along with the continuous features $\boldsymbol{x}_{\text{cont}}$ to form a vector of dimension $(d \times m + c)$. This vector is inputted to an MLP, denoted by $g_{\boldsymbol{\psi}}$, to predict the target y. Let H be the cross-entropy for classification tasks and mean square error for regression tasks. We minimize the following loss function $\mathcal{L}(\boldsymbol{x}, y)$ to learn all the TabTransformer parameters in an end-to-end learning by the first-order gradient methods. The TabTransformer parameters include $\boldsymbol{\phi}$ for column embedding, $\boldsymbol{\theta}$ for Transformer layers, and $\boldsymbol{\psi}$ for the top MLP layer.

$$\mathcal{L}(\boldsymbol{x}, y) \equiv H(g_{\boldsymbol{\psi}}(f_{\boldsymbol{\theta}}(\boldsymbol{E}_\phi(\boldsymbol{x}_{\text{cat}})), \boldsymbol{x}_{\text{cont}}), y)\,. \tag{1}$$

Below, we explain the Transformer layers and column embedding.

Transformer. A Transformer (Vaswani et al. 2017) consists of a multi-head self-attention layer followed by a position-wise feed-forward layer, with element-wise addition and layer-normalization being done after each layer. A self-attention layer comprises three parametric matrices - Key, Query and Value. Each input embedding is projected on to these matrices, to generate their key, query and value vectors. Formally, let $K \in \mathbb{R}^{m \times k}$, $Q \in \mathbb{R}^{m \times k}$ and $V \in \mathbb{R}^{m \times v}$ be the matrices comprising key, query and value vectors of all the embeddings, respectively, and m be the number of embeddings inputted to the Transformer, k and v be the dimensions of the key and value vectors, respectively. Every input embedding attends to all other embeddings through a Attention head, which is computed as follows:

$$\text{Attention}(K, Q, V) = A \cdot V, \tag{2}$$

where $A = \text{softmax}((QK^T)/\sqrt{k})$. For each embedding, the attention matrix $A \in \mathbb{R}^{m \times m}$ calculates how much it attends to other embeddings, thus transforming the embedding into contextual one. The output of the attention head of dimension v is projected back to the embedding of dimension d through a fully connected layer, which in turn is passed through two position-wise feed-forward layers. The first layer expands the embedding to four times its size and the second layer projects it back to its original size.

Column embedding. For each categorical feature (column) i, we have an embedding lookup table $\boldsymbol{e}_{\phi_i}(.)$, for $i \in \{1, 2, ..., m\}$. For ith feature with d_i classes, the embedding table $\boldsymbol{e}_{\phi_i}(.)$ has $(d_i + 1)$ embeddings where the additional embedding corresponds to a missing value. The embedding for the encoded value $x_i = j \in [0, 1, 2, .., d_i]$ is $\boldsymbol{e}_{\phi_i}(j) = [\boldsymbol{c}_{\phi_i}, \boldsymbol{w}_{\phi_{ij}}]$, where $\boldsymbol{c}_{\phi_i} \in \mathbb{R}^\ell, \boldsymbol{w}_{\phi_{ij}} \in \mathbb{R}^{d-\ell}$. The dimension of $\boldsymbol{c}_{\phi_i}$, ℓ, is a hyper-parameter. The unique identifier $\boldsymbol{c}_{\phi_i} \in \mathbb{R}^\ell$ distinguishes the classes in column i from those in the other columns.

The use of unique identifier is new and is particularly designed for tabular data. Rather in language modeling, embeddings are element-wisely added with the positional encoding of the word in the sentence. Since, in tabular data, there is no ordering of the features, we do not use positional encodings. An ablation study on different embedding strategies is given in Appendix A. The strategies include both different choices for ℓ, d and element-wise adding the unique identifier and feature-value specific embeddings rather than concatenating them.

Pre-training the Embeddings. The contextual embeddings explained above are learned in end-to-end supervised training using labeled examples. For a scenario, when there are a few labeled examples and a large number of unlabeled examples, we introduce a pre-training procedure to train the Transformer layers using unlabeled data. This is followed by fine-tuning of the pre-trained Transformer layers along with the top MLP layer using the labeled data. For fine-tuning, we use the supervised loss defined in Equation (1).

We explore two different types of pre-training procedures, the masked language modeling (MLM) (Devlin et al. 2019) and the replaced token detection (RTD) (Clark et al. 2020). Given an input $\boldsymbol{x}_{\text{cat}} = \{x_1, x_2, ..., x_m\}$, MLM randomly selects $k\%$ features from index 1 to m and masks them as missing. The Transformer layers along with the column embeddings are trained by minimizing cross-entropy loss of a multi-class classifier that tries to predict the original features of the masked features, from the contextual embedding outputted from the top-layer Transformer.

Instead of masking features, RTD replaces the original feature by a random value of that feature. Here, the loss is minimized for a binary classifier that tries to predict whether or not the feature has been replaced. The RTD procedure as proposed in (Clark et al. 2020) uses auxiliary generator for sampling a subset of features that a feature should be replaced with. The reason they used an auxiliary encoder network as the generator is that there are tens of thousands of tokens in language data and a uniformly random token is too easy to detect. In contrast, (a) the number of classes within each categorical feature is typically limited; (b) a different binary classifier is defined for each column rather than a shared one, as each column has its own embedding lookup table. We name the two pre-training methods as TabTransformer-MLM and TabTransformer-RTD. In our experiments, the replacement value k is set to 30. An ablation study on k is given in Appendix A.

3 Experiments

Data. We evaluate TabTransformer and baseline models on 15 publicly available binary classification datasets from the UCI repository (Dua and Graff 2017), the AutoML Challenge (Guyon et al. 2019), and Kaggle (Kaggle, Inc. 2017) for both supervised and semi-supervised learning. Each dataset is divided into five cross-validation splits. The training/validation/testing proportion of the data for each split are $65/15/20\%$. The number of categorical features across dataset ranges from 2 to 136. In the semi-supervised experiments, for each dataset and split, the first p observations in the training data are marked as the labeled data and the remaining training data as the unlabeled set. The value of p is chosen as 50, 200, and 500, corresponding to 3 different scenarios. In the supervised experiments, each training dataset is fully labeled. Summary statistics of the all the datasets are provided in Table 8, 9 in Appendix C.

Setup. For the TabTransformer, the hidden (embedding) dimension, the number of layers and the number of attention heads are fixed to 32, 6, and 8 respectively. The MLP layer sizes are set to $\{4 \times l, 2 \times l\}$, where l is the size of its input. For hyperparameter optimization (HPO), each model is given 20 HPO rounds for each cross-validation split. For evaluation metrics, we use the Area under the curve (AUC) (Bradley 1997). Note, the pre-training is only applied in semi-supervised scenario. We do not find much benefit in using it when the entire data is labeled. Its benefit is evident when there is a large number of unlabeled examples and a few labeled examples. Since in this scenario the pre-training provides a representation of the data that could not have been learned based only on the labeled examples.

The experiment section is organized as follows. In Section 3.1, we first demonstrate the effectiveness of the attention-based Transformer by comparing our model with the one without the Transformers (equivalently an MLP model). In Section 3.2, we illustrate the robustness of TabTransformer against noisy and missing data. Finally, extensive evaluation on various methods are conducted in Section 3.3 for supervised learning, and in Section 3.4 for semi-supervised learning.

3.1 The Effectiveness of the Transformer Layers

First, a comparison between TabTransformers and the baseline MLP is conducted in a supervised learning scenario. We remove the Transformer layers $f_{\boldsymbol{\theta}}$ from the architecture, fix the rest of the components, and compare it with the original TabTransformer. The model without the attention-based Transformer layers is equivalently an MLP. The dimension of embeddings d for categorical features is set as 32 for both models. The comparison results over 15 datasets are presented in Table 1. The TabTransformer with the Transformer layers outperforms the baseline MLP on 14 out of 15 datasets with an average 1.0% gain in AUC.

Next, we take contextual embeddings from different layers of the Transformer and compute a t-SNE plot (Maaten and Hinton 2008) to visualize their similarity in function space. More precisely, for each dataset we take its test data, pass their categorical features into a trained TabTransformer, and extract all contextual embeddings (across all columns) from a certain layer of the Transformer. The t-SNE algorithm is then used to reduce each embedding to a 2D point in the t-SNE plot. Figure 2 (left) shows the 2D visualization of embeddings from the last layer of the Transformer for dataset *bank_marketing*. Each marker in the plot represents an average of 2D points over the test data points for a certain class. We can see that semantically similar classes are close with each other and form clusters in the embedding space. Each cluster is annotated by a set of labels. For example, we find that all of the client-based features (color markers) such as job, education level and martial status stay close in the center and non-client based features (gray markers) such as month (last contact month of the year), day (last contact day of the week) lie outside the central area; in the bottom cluster the embedding of owning a housing loan stays close with that of being default; over the left cluster, embeddings of being a student, martial status as single, not having a housing loan, and education level as tertiary get together; and in the right cluster, education levels are closely associated with the occupation types (Torpey and Watson 2014). In Figure 2, the center and right plots are t-SNE plots of embeddings before being passed through the Transformer and the context-free embeddings from MLP, respectively. For the embeddings before being passed into the Transformer, it starts to distinguish the non-client based features (gray markers) from the client-based features (color markers). For the embeddings from MLP, we do not observe such pattern and many categorical features which are not semantically similar are grouped together, as indicated by the annotation in the plot.

Table 1: Comparison between TabTransfomers and the baseline MLP. The evaluation metric is AUC in percentage.

Dataset	Baseline MLP	TabTransformer	Gain (%)
albert	74.0	75.7	**1.7**
1995_income	90.5	90.6	**0.1**
dota2games	63.1	63.3	**0.2**
hcdr_main	74.3	75.1	**0.8**
adult	72.5	73.7	**1.2**
bank_marketing	92.9	93.4	**0.5**
blastchar	83.9	83.5	-0.4
insurance_co	69.7	74.4	**4.7**
jasmine	85.1	85.3	**0.2**
online_shoppers	91.9	92.7	**0.8**
philippine	82.1	83.4	**1.3**
qsar_bio	91.0	91.8	**0.8**
seismicbumps	73.5	75.1	**1.6**
shrutime	84.6	85.6	**1.0**
spambase	98.4	98.5	**0.1**

In addition to prove the effectiveness of Transformer layers, on the test data we take all of the contextual embeddings from each Transformer layer of a trained TabTransformer, use the embeddings from each layer along with the continuous variables as features, and separately fit a linear model with target y. Since all of the experimental datasets are for binary classification, the linear model is logistic regression. The motivation for this evaluation is defining the success of a simple linear model as a measure of quality for the learned embeddings.

For each dataset and each layer, an average of CV-score in AUC on the test data is computed. The evaluation is conducted on the entire test data with number of data points over 9000. Figure 3 presents results for dataset *BankMarketing*, *Adult*, and *QSAR_Bio*. For each line, each prediction score is normalized by the "best score" from an end-to-end trained TabTransformer for the corresponding dataset. We also ex-

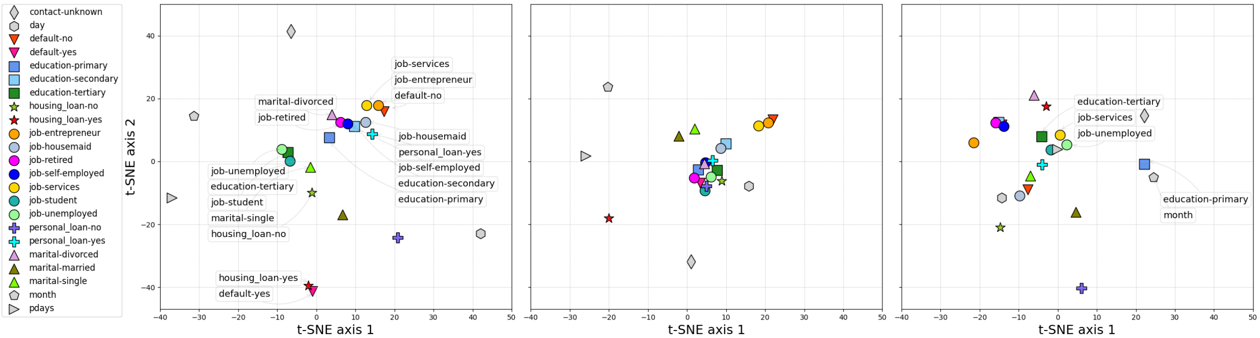

Figure 2: t-SNE plots of learned embeddings for categorical features on dataset *BankMarketing*. **Left**: TabTransformer-the embeddings generated from the last layer of the attention-based Transformer. **Center**: TabTransformer-the embeddings before being passed into the attention-based Transformer. **Right**: The embeddings learned from MLP.

plore the average and maximum pooling strategy (Howard and Ruder 2018) rather than concatenation of embeddings as the features for the linear model. The upward pattern clearly shows that embeddings becomes more effective as the Transformer layer progresses. In contrast, the embeddings from MLP (the single black markers) perform worse with a linear model. Furthermore, the last value in each line close to 1.0 indicates that a linear model with the last layer of embeddings as features can achieve reliable accuracy, which confirms our assumption.

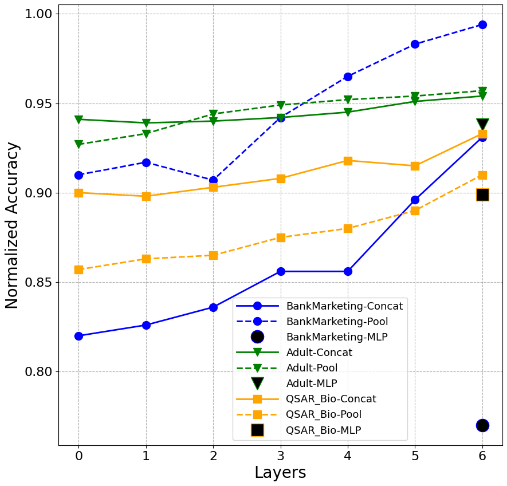

Figure 3: Predictions of liner models using features as the embeddings extracted from different Transformer layers in TabTransformer. Layer 0 corresponds to the embeddings before being passed into the Transformer layers. For each dataset, each prediction score is normalized by the "best score" from an end-to-end trained TabTransformer.

3.2 The Robustness of TabTransformer

We further demonstrate the robustness of TabTransformer on the noisy data and data with missing values, against the baseline MLP. We consider these two scenarios only on categorical features to specifically prove the robustness of contextual embeddings from the Transformer layers.

Noisy Data. On the test examples, we firstly contaminate the data by replacing a certain number of values by randomly generated ones from the corresponding columns (features). Next, the noisy data are passed into a trained TabTransformer to compute a prediction AUC score. Results on a set of 3 different dataets are presented in Figure 4. As the noisy rate increases, TabTransformer performs better in prediction accuracy and thus is more robust than MLP. In particular notice the *Blastchar* dataset where the performance is near identical with no noise, yet as the noise increases, TabTransformer becomes significantly more performant compared to the baseline. We conjecture that the robustness comes from the contextual property of the embeddings. Despite a feature being noisy, it draws information from the correct features allowing for a certain amount of correction.

Data with Missing Values. Similarly, on the test data we artificially select a number of values to be missing and send the data with missing values to a trained TabTransformer to compute the prediction score. There are two options to handle the embeddings of missing values: (1) Use the average learned embeddings over all classes in the corresponding column; (2) the embedding for the class of missing value, the additional embedding for each column mentioned in Section 2. Since the benchmark datasets do not contain enough missing values to effectively train the embedding in option (2), we use the average embedding in (1) for imputation. Results on the same 3 datasets are presented in Figure 5. We can see the same patterns of the noisy data case, i.e. that the TabTransformer shows better stability than MLP in handling missing values.

3.3 Supervised Learning

Here we compare the performance of TabTransformer against following four categories of methods: (a) Logistic

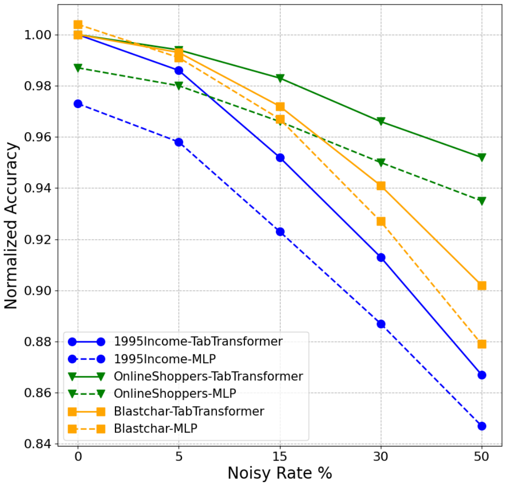

Figure 4: Performance of TabTransformer and MLP with noisy data. For each dataset, each prediction score is normalized by the score of TabTransformer at 0 noise.

Table 2: Model performance in supervised learning. The evaluation metric is mean ± standard deviation of AUC score over the 15 datasets for each model. Larger the number, better the result. The top 2 numbers are bold.

Model Name	Mean AUC (%)
TabTransformer	$\mathbf{82.8} \pm 0.4$
MLP	81.8 ± 0.4
GBDT	$\mathbf{82.9} \pm 0.4$
Sparse MLP	81.4 ± 0.4
Logistic Regression	80.4 ± 0.4
TabNet	77.1 ± 0.5
VIB	80.5 ± 0.4

regression and GBDT (b) MLP and a sparse MLP following (Morcos et al. 2019) (c) TabNet model of Arik and Pfister (2019) (d) and the Variational Information Bottleneck model (VIB) of Alemi et al. (2017).

Results are summarized in Table 2. TabTransformer, MLP, and GBDT are the top 3 performers. The TabTransformer outperforms the baseline MLP with an average 1.0% gain and perform comparable with the GBDT. Furthermore, the TabTransformer is significantly better than TabNet and VIB, the recent deep networks for tabular data. For experiment and model details, see Appendix B. The models' performances on each individual dataset are presented in Table 16 and 17 in Appendix C.

3.4 Semi-supervised Learning

Lastly, we evaluate the TabTransformer under the semi-supervised learning scenario where few labeled training examples are available together with a significant number of unlabeled samples. Specifically, we compare our pretrained and then fine-tuned TabTransformer-RTD/MLM against following semi-supervised models: (a) Entropy Regularization (ER) (Grandvalet and Bengio 2006) combined with MLP and TabTransformer (b) Pseudo Labeling (PL) (Lee 2013) combined with MLP, TabTransformer, and GBDT (Jain 2017) (c) MLP (DAE): an unsupervised pre-training method designed for deep models on tabular data: the swap noise Denoising AutoEncoder (Jahrer 2018).

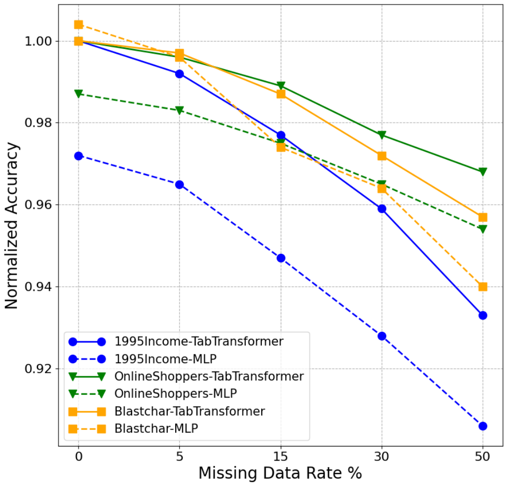

Figure 5: Performance of TabTransformer and MLP under missing data scenario. For each dataset, each prediction score is normalized by the score of TabTransformer trained without missing values.

The pre-training models TabTransformer-MLM, TabTransformer-RTD and MLP (DAE) are firstly pre-trained on the entire unlabeled training data and then fine-tuned on labeled data. The semi-supervised learning methods, Pseudo Labeling and Entropy Regularization, are trained on the mix of labeled and unlabeled training data. To better present results, we split the set of 15 datasets into two subsets. The first set includes 6 datasets with more than 30K data points and the second set includes remaining 9 datasets.

The results are presented in Table 3 and Table 4. When the number of unlabeled data is large, Table 3 shows that our TabTransformer-RTD and TabTransformer-MLM significantly outperform all the other competitors. Particularly, TabTransformer-RTD/MLM improves over all the other competitors by at least 1.2%, 2.0% and 2.1% on mean AUC for the scenario of 50, 200, and 500 labeled data points respectively. The Transformer-based semi-supervised learning methods TabTransformer (ER) and TabTransformer (PL) and the tree-based semi-supervised learning method GBDT (PL) perform worse than the average of all the models. When the number of unlabeled data becomes smaller, as shown in Table 4, TabTransformer-RTD still outperforms most of its

Table 3: Semi-supervised learning results for 8 datasets each with more than 30K data points, for different number of labeled data points. Evaluation metrics are mean AUC in percentage. Larger the number, better the result.

# Labeled data	50	200	500
TabTransformer-RTD	66.6 ± 0.6	70.9 ± 0.6	**73.1** ± 0.6
TabTransformer-MLM	**66.8** ± 0.6	**71.0** ± 0.6	72.9 ± 0.6
MLP (ER)	65.6 ± 0.6	69.0 ± 0.6	71.0 ± 0.6
MLP (PL)	65.4 ± 0.6	68.8 ± 0.6	71.0 ± 0.6
TabTransformer (ER)	62.7 ± 0.6	67.1 ± 0.6	69.3 ± 0.6
TabTransformer (PL)	63.6 ± 0.6	67.3 ± 0.7	69.3 ± 0.6
MLP (DAE)	65.2 ± 0.5	68.5 ± 0.6	71.0 ± 0.6
GBDT (PL)	56.5 ± 0.5	63.1 ± 0.6	66.5 ± 0.7

Table 4: Semi-supervised learning results for 12 datasets each with less than 30K data points, for different number of labeled data points. Evaluation metrics are mean AUC in percentage. Larger the number, better the result.

# Labeled data	50	200	500
TabTransformer-RTD	78.6 ± 0.6	**81.6** ± 0.5	**83.4** ± 0.5
TabTransformer-MLM	78.5 ± 0.6	81.0 ± 0.6	82.4 ± 0.5
MLP (ER)	**79.4** ± 0.6	81.1 ± 0.6	82.3 ± 0.6
MLP (PL)	79.1 ± 0.6	81.1 ± 0.6	82.0 ± 0.6
TabTransformer (ER)	77.9 ± 0.6	81.2 ± 0.6	82.1 ± 0.6
TabTransformer (PL)	77.8 ± 0.6	81.0 ± 0.6	82.1 ± 0.6
MLP (DAE)	78.5 ± 0.7	80.7 ± 0.6	82.2 ± 0.6
GBDT (PL)	73.4 ± 0.7	78.8 ± 0.6	81.3 ± 0.6

competitors but with a marginal improvement.

Furthermore, we observe that when the number of unlabeled data is small as shown in Table 4, TabTransformer-RTD performs better than TabTransformer-MLM, thanks to its easier pre-training task (a binary classification) than that of MLM (a multi-class classification). This is consistent with the finding of the ELECTRA paper (Clark et al. 2020). In Table 4, with only 50 labeled data points, MLP (ER) and MLP (PL) beat our TabTransformer-RTD/MLM. This can be attributed to the fact that there is room for improvement in our fine-tuning procedure. In particular, our approach allows to obtain informative embeddings but does not allow the weights of the classifier itself to be trained with unlabelled data. Since this issue does not occur for ER and PL, they obtain an advantage in extremely small labelled set. We point out however that this only means that the methods are complementary and mention that a possible follow up could combine the best of all approaches.

Both evaluation results, Table 3 and Table 4, show that our TabTransformer-RTD and Transformers-MLM models are promising in extracting useful information from unlabeled data to help supervised training, and are particularly useful when the size of unlabeled data is large. For model performance on each individual dataset see Table 10, 11, 12, 13, 14, 15 in Appendix C.

4 Related Work

Supervised learning. Standard MLPs have been applied to tabular data for many years (De Brébisson et al. 2015). For deep models designed specifically for tabular data, there are deep versions of factorization machines (Guo et al. 2018; Xiao et al. 2017), Transformers-based methods (Song et al. 2019; Li et al. 2020; Sun et al. 2019), and deep versions of decision-tree-based algorithms (Ke et al. 2019; Yang, Morillo, and Hospedales 2018). In particular, (Song et al. 2019) applies one layer of multi-head attention on embeddings to learn higher order features. The higher order features are concatenated and inputted to a fully connected layer to make the final prediction. (Li et al. 2020) use self-attention layers and track the attention scores to obtain feature importance scores. (Sun et al. 2019) combine the Factorization Machine model with transformer mechanism. All 3 papers are focused on recommendation systems making it hard to have a clear comparison with this paper. Other models have been designed around the purported properties of tabular data such as low-order and sparse feature interactions. These include Deep & Cross Networks (Wang et al. 2017), Wide & Deep Networks (Cheng et al. 2016), TabNets (Arik and Pfister 2019), and AdaNet (Cortes et al. 2016).

Semi-supervised learning. (Izmailov et al. 2019) give a semi-supervised method based on density estimation and evaluate their approach on tabular data. *Pseudo labeling* (Lee 2013) is a simple, efficient and popular baseline method. The Pseudo labeling uses the current network to infer pseudo-labels of unlabeled examples, by choosing the most confident class. These pseudo-labels are treated like human-provided labels in the cross entropy loss. *Label propagation* (Zhu and Ghahramani 2002), (Iscen et al. 2019) is a similar approach where a node's labels propagate to all nodes according to their proximity, and are used by the training model as if they were the true labels. Another standard method in semi-supervised learning is *entropy regularization* (Grandvalet and Bengio 2005; Sajjadi, Javanmardi, and Tasdizen 2016). It adds average per-sample entropy for the unlabeled examples to the original loss function for the labeled examples. Another classical approach of semi-supervised learning is co-training (Nigam and Ghani 2000). However, the recent approaches - entropy regularization and pseudo labeling - are typically better and more popular. A succinct review of semi-supervised learning methods in general can be found in (Oliver et al. 2019; Chappelle, Schölkopf, and Zien 2010).

5 Conclusion

We proposed TabTransformer, a novel deep tabular data modeling architecture for supervised and semi-supervised learning. We provide extensive empirical evidence showing TabTransformer significantly outperforms MLP and recent deep networks for tabular data while matching the performance of tree-based ensemble models (GBDT). We provide and extensively study a two-phase pre-training then fine-tune procedure for tabular data, beating the state-of-the-art performance of semi-supervised learning methods. TabTransformer shows promising results for robustness against noisy and missing data, and interpretability of the contextual embeddings. For future work, it would be interesting to investigate them in detail.

References

Alemi, A. A.; Fischer, I.; and Dillon, J. V. 2018. Uncertainty in the Variational Information Bottleneck. *arXiv:1807.00906 [cs, stat]* URL http://arxiv.org/abs/1807.00906. ArXiv: 1807.00906.

Alemi, A. A.; Fischer, I.; Dillon, J. V.; and Murphy, K. 2017. Deep Variational Information Bottleneck. *International Conference on Learning Representations* abs/1612.00410. URL https://arxiv.org/abs/1612.00410.

Arik, S. O.; and Pfister, T. 2019. TabNet: Attentive Interpretable Tabular Learning. *arXiv preprint arXiv:1908.07442* URL https://arxiv.org/abs/1908.07442.

Ban, G.-Y.; El Karoui, N.; and Lim, A. E. 2018. Machine learning and portfolio optimization. *Management Science* 64(3): 1136–1154.

Bradley, A. P. 1997. The use of the area under the ROC curve in the evaluation of machine learning algorithms. *Pattern recognition* 30(7): 1145–1159.

Brunner, G.; Liu, Y.; Pascual, D.; Richter, O.; and Wattenhofer, R. 2019. On the validity of self-attention as explanation in transformer models. *arXiv preprint arXiv:1908.04211* .

Chapelle, O.; Scholkopf, B.; and Zien, A. 2009. Semi-supervised learning). *IEEE Transactions on Neural Networks* 20(3): 542–542.

Chappelle, O.; Schölkopf, B.; and Zien, A. 2010. Semi-supervised learning. Adaptive Computation and Machine Learning.

Chen, T.; and Guestrin, C. 2016. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, 785–794.

Cheng, H.-T.; Koc, L.; Harmsen, J.; Shaked, T.; Chandra, T.; Aradhye, H.; Anderson, G.; Corrado, G.; Chai, W.; Ispir, M.; et al. 2016. Wide & deep learning for recommender systems. In *Proceedings of the 1st workshop on deep learning for recommender systems*, 7–10.

Clark, K.; Luong, M.-T.; Le, Q. V.; and Manning, C. D. 2020. ELECTRA: Pre-training Text Encoders as Discriminators Rather Than Generators. In *International Conference on Learning Representations*. URL https://openreview.net/forum?id=r1xMH1BtvB.

Coenen, A.; Reif, E.; Yuan, A.; Kim, B.; Pearce, A.; Viégas, F.; and Wattenberg, M. 2019. Visualizing and measuring the geometry of bert. *arXiv preprint arXiv:1906.02715* .

Cortes, C.; Gonzalvo, X.; Kuznetsov, V.; Mohri, M.; and Yang, S. 2016. AdaNet: Adaptive Structural Learning of Artificial Neural Networks.

De Brébisson, A.; Simon, E.; Auvolat, A.; Vincent, P.; and Bengio, Y. 2015. Artificial Neural Networks Applied to Taxi Destination Prediction. In *Proceedings of the 2015th International Conference on ECML PKDD Discovery Challenge - Volume 1526*, ECMLPKDDDC'15, 40–51. Aachen, DEU: CEUR-WS.org.

Devlin, J.; Chang, M.-W.; Lee, K.; and Toutanova, K. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *NAACL-HLT*.

Dua, D.; and Graff, C. 2017. UCI Machine Learning Repository. URL http://archive.ics.uci.edu/ml.

Grandvalet, Y.; and Bengio, Y. 2005. Semi-supervised learning by entropy minimization. In *Advances in neural information processing systems*, 529–536.

Grandvalet, Y.; and Bengio, Y. 2006. Entropy regularization. *Semi-supervised learning* 151–168.

Guo, H.; Tang, R.; Ye, Y.; Li, Z.; He, X.; and Dong, Z. 2018. DeepFM: An End-to-End Wide & Deep Learning Framework for CTR Prediction. *arXiv:1804.04950 [cs, stat]* URL http://arxiv.org/abs/1804.04950. ArXiv: 1804.04950.

Guyon, I.; Sun-Hosoya, L.; Boullé, M.; Escalante, H. J.; Escalera, S.; Liu, Z.; Jajetic, D.; Ray, B.; Saeed, M.; Sebag, M.; Statnikov, A.; Tu, W.; and Viegas, E. 2019. Analysis of the AutoML Challenge series 2015-2018. In *AutoML*, Springer series on Challenges in Machine Learning. URL https://www.automl.org/wp-content/uploads/2018/09/chapter10-challenge.pdf.

Howard, J.; and Ruder, S. 2018. Universal language model fine-tuning for text classification. *arXiv preprint arXiv:1801.06146* .

Huang, Z.; Xu, W.; and Yu, K. 2015. Bidirectional LSTM-CRF models for sequence tagging. *arXiv preprint arXiv:1508.01991* .

Iscen, A.; Tolias, G.; Avrithis, Y.; and Chum, O. 2019. Label propagation for deep semi-supervised learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 5070–5079.

Izmailov, P.; Kirichenko, P.; Finzi, M.; and Wilson, A. G. 2019. Semi-Supervised Learning with Normalizing Flows. *arXiv:1912.13025 [cs, stat]* URL http://arxiv.org/abs/1912.13025. ArXiv: 1912.13025.

Jahrer, M. 2018. Porto Seguro's Safe Driver Prediction. URL https://kaggle.com/c/porto-seguro-safe-driver-prediction.

Jain, S. 2017. Introduction to Pseudo-Labelling : A Semi-Supervised learning technique. https://www.analyticsvidhya.com/blog/2017/09/pseudo-labelling-semi-supervised-learning-technique/.

Kaggle, Inc. 2017. The State of ML and Data Science 2017. URL https://www.kaggle.com/surveys/2017.

Ke, G.; Meng, Q.; Finley, T.; Wang, T.; Chen, W.; Ma, W.; Ye, Q.; and Liu, T.-Y. 2017. LightGBM: A highly efficient gradient boosting decision tree. In *Advances in Neural Information Processing Systems*, 3146–3154. URL https://papers.nips.cc/paper/6907-lightgbm-a-highly-efficient-gradient-boosting-decision-tree.pdf.

Ke, G.; Zhang, J.; Xu, Z.; Bian, J.; and Liu, T.-Y. 2019. TabNN: A Universal Neural Network Solution for Tabular Data. URL https://openreview.net/forum?id=r1eJssCqY7.

Kendall, A.; and Gal, Y. 2017. What uncertainties do we need in bayesian deep learning for computer vision? In *Advances in neural information processing systems*, 5574–5584.

Klambauer, G.; Unterthiner, T.; Mayr, A.; and Hochreiter, S. 2017. Self-normalizing neural networks. In *Advances in neural information processing systems*, 971–980.

Lee, D.-H. 2013. Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks. In *Workshop on challenges in representation learning, ICML*, volume 3, 2.

Li, Z.; Cheng, W.; Chen, Y.; Chen, H.; and Wang, W. 2020. Interpretable Click-Through Rate Prediction through Hierarchical Attention. In *Proceedings of the 13th International Conference on Web Search and Data Mining*, 313–321. Houston TX USA: ACM. ISBN 978-1-4503-6822-3. doi:10.1145/3336191.3371785. URL http://dl.acm.org/doi/10.1145/3336191.3371785.

Loshchilov, I.; and Hutter, F. 2017. Decoupled Weight Decay Regularization. In *International Conference on Learning Representations*. URL https://arxiv.org/abs/1711.05101.

Maaten, L. v. d.; and Hinton, G. 2008. Visualizing data using t-SNE. *Journal of machine learning research* 9(Nov): 2579–2605.

Mikolov, T.; Kombrink, S.; Burget, L.; Černockỳ, J.; and Khudanpur, S. 2011. Extensions of recurrent neural network language model. In *2011 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, 5528–5531. IEEE.

Morcos, A. S.; Yu, H.; Paganini, M.; and Tian, Y. 2019. One ticket to win them all: generalizing lottery ticket initializations across datasets and optimizers. *arXiv:1906.02773 [cs, stat]* URL http://arxiv.org/abs/1906.02773. ArXiv: 1906.02773.

Nigam, K.; and Ghani, R. 2000. Analyzing the effectiveness and applicability of co-training. In *Proceedings of the ninth international conference on Information and knowledge management*, 86–93.

Oliver, A.; Odena, A.; Raffel, C.; Cubuk, E. D.; and Goodfellow, I. J. 2019. Realistic Evaluation of Deep Semi-Supervised Learning Algorithms. *arXiv:1804.09170 [cs, stat]* URL http://arxiv.org/abs/1804.09170. ArXiv: 1804.09170.

Oliver, A.; Odena, A.; Raffel, C. A.; Cubuk, E. D.; and Goodfellow, I. 2018. Realistic evaluation of deep semi-supervised learning algorithms. In *Advances in Neural Information Processing Systems*, 3235–3246.

Paszke, A.; Gross, S.; Massa, F.; Lerer, A.; Bradbury, J.; Chanan, G.; Killeen, T.; Lin, Z.; Gimelshein, N.; Antiga, L.; Desmaison, A.; Kopf, A.; Yang, E.; DeVito, Z.; Raison, M.; Tejani, A.; Chilamkurthy, S.; Steiner, B.; Fang, L.; Bai, J.; and Chintala, S. 2019. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In Wallach, H.; Larochelle, H.; Beygelzimer, A.; d'Alché Buc, F.; Fox, E.; and Garnett, R., eds., *Advances in Neural Information Processing Systems 32*, 8024–8035. Curran Associates, Inc. URL http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf.

Prokhorenkova, L.; Gusev, G.; Vorobev, A.; Dorogush, A. V.; and Gulin, A. 2018. CatBoost: unbiased boosting with categorical features. In *Advances in neural information processing systems*, 6638–6648.

Rong, X. 2014. word2vec parameter learning explained. *arXiv preprint arXiv:1411.2738* .

Sajjadi, M.; Javanmardi, M.; and Tasdizen, T. 2016. Regularization with stochastic transformations and perturbations for deep semi-supervised learning. In *Advances in neural information processing systems*, 1163–1171.

Sandler, M.; Howard, A.; Zhu, M.; Zhmoginov, A.; and Chen, L.-C. 2018. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 4510–4520.

Song, W.; Shi, C.; Xiao, Z.; Duan, Z.; Xu, Y.; Zhang, M.; and Tang, J. 2019. AutoInt: Automatic Feature Interaction Learning via Self-Attentive Neural Networks. *Proceedings of the 28th ACM International Conference on Information and Knowledge Management - CIKM '19* 1161–1170. doi: 10.1145/3357384.3357925. URL http://arxiv.org/abs/1810.11921. ArXiv: 1810.11921.

Stretcu, O.; Viswanathan, K.; Movshovitz-Attias, D.; Platanios, E.; Ravi, S.; and Tomkins, A. 2019. Graph Agreement Models for Semi-Supervised Learning. In *Advances in Neural Information Processing Systems 32*, 8713–8723. Curran Associates, Inc. URL http://papers.nips.cc/paper/9076-graph-agreement-models-for-semi-supervised-learning.pdf.

Sun, Q.; Cheng, Z.; Fu, Y.; Wang, W.; Jiang, Y.-G.; and Xue, X. 2019. DeepEnFM: Deep neural networks with Encoder enhanced Factorization Machine URL https://openreview.net/forum?id=SJlyta4YPS.

Tanha, J.; Someren, M.; and Afsarmanesh, H. 2017. Semi-supervised self-training for decision tree classifiers. *International Journal of Machine Learning and Cybernetics* 8: 355–370.

Torpey, E.; and Watson, A. 2014. *Education level and jobs: Opportunities by state*. URL https://www.bls.gov/careeroutlook/2014/article/education-level-and-jobs.htm.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, Ł.; and Polosukhin, I. 2017. Attention is all you need. In *Advances in neural information processing systems*, 5998–6008.

Voulodimos, A.; Doulamis, N.; Doulamis, A.; and Protopapadakis, E. 2018. Deep learning for computer vision: A brief review. *Computational intelligence and neuroscience* 2018.

Wang, R.; Fu, B.; Fu, G.; and Wang, M. 2017. Deep & Cross Network for Ad Click Predictions. In *ADKDD@KDD*.

Xiao, J.; Ye, H.; He, X.; Zhang, H.; Wu, F.; and Chua, T.-S. 2017. Attentional Factorization Machines: Learning the Weight of Feature Interactions via Attention Networks. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence*, 3119–3125. Melbourne, Australia: International Joint Conferences on Artificial Intelligence Organization. ISBN 978-0-9992411-0-3. doi:10.24963/ijcai.2017/435. URL https://www.ijcai.org/proceedings/2017/435.

Yang, Y.; Morillo, I. G.; and Hospedales, T. M. 2018. Deep neural decision trees. *arXiv preprint arXiv:1806.06988* .

Zhu, X.; and Ghahramani, Z. 2002. Learning from labeled and unlabeled data with label propagation .

A Appendix: Ablation Studies

We perform a number of ablation studies on various architectural choices and pre-training approaches for our TabTransformer. The first ablation study is on the choice of column embedding. The second and third ablation studies focus on the pre-training approach. Specifically, they are on the replacement value k and dynamic versus static replacement strategy. For the pre-training approach, we use TabTransformer-RTD as our model. That is, the loss in the pre-training is RTD loss. For TabTransformer, the hidden (embedding) dimension, the number of layers and the number of attention heads in the Transformer are set to 32, 6, and 8 respectively. The MLP layer sizes are set to $\{4 \times l, 2 \times l\}$, where l is the size of its input. To better present the result, we introduce an additional evaluation metric, the *relative AUC*. More precisely, for each dataset and cross-validation split, the *relative AUC* for a model is the relative change of its AUC against the mean AUC over all competing models.

Column Embedding. The first study is on the choice of column embedding – shared parameters $\boldsymbol{c}_{\phi_i}$ across the embeddings of multiple classes in column i for $i \in \{1, 2, ..., m\}$. In particular, we study the optimal dimension of $\boldsymbol{c}_{\phi_i}$, ℓ. An alternative choice is to element-wisely add the unique identifier $\boldsymbol{c}_{\phi_i}$ and feature-value specific embeddings $\boldsymbol{w}_{\phi_{ij}}$ rather than concatenating them. In that case, both the dimension of $\boldsymbol{c}_{\phi_i}$ and $\boldsymbol{w}_{\phi_{ij}}$ are equal to the dimension of embedding d. The goal of having column embedding is to enable the model to distinguish the classes in one column from those in the other columns. A baseline approach is to not have any shared embedding. Results are presented in Table 5 where "Col Embed-Concat-1/X" indicates that the dimension ℓ is set as d/X. The relative AUC score is calculated over all the models that appear in the rows and columns in the table, which explains why negative scores appear in some of the entries. Results show that not having the shared column embedding performs worst and our concatenation column embedding gives an average better performance.

The replacement value k. The second ablation study is on the replacement value k in pre-training approach. We run experiments for three different choices of k – $\{15, 30, 50\}$ on three different datasets, namely – *Adult*, *BankMarketing*, and *1995_income*. The TabTransformer is firstly pre-trained with a value of k on unlabeled data and then fine-tuned on labeled data. The number of labeled data is set as 256. The final fine-tuning accuracy is not much sensitive to the value of k, as shown in Table 6. The pre-training curves of training and validation accuracy for the three different replacement value k is shown in Figure 6. Note, that a constant prediction model would achieve 85% accuracy for the 15% replacement value.

Dynamic versus Static Replacement. The third ablation study is on dynamic vs static replacement in the pre-training approach. In dynamic replacement, we randomly replace feature values during pre-training over the epochs. That is the replacement is different in each epoch. Whereas in static replacement, the random replacement is chosen once, and then the same replacement is used in all the epochs. We combine this study with another ablation on shared RTD binary classifier (predictor) vs. different classifiers for different columns. Results in Table 7 show that our choice of dynamic replacement and un-shared RTD classifiers perform better than static replacement and shared RTD classifiers. Figure 7 shows the pre-training curves of training and validation accuracy for the three choices – dynamic replacement, static replacement, and static replacement with a shared RTD classifier.

B Appendix: Experiment and Model Details

In this section, we discuss the experiments and model details. First, we go through the experiments details and hyper parameters search space for HPO in Section B.1. Next, we discuss the feature engineering in Section B.2.

B.1 Experiments Details and Hyper Parameters

Setup. All experiments were run on an Ubuntu Linux machine with 8 CPUs and 60GB memory, with all models using a single NVIDIA V100 Tensor Core GPU. For the competing models mentioned in the experiment, we re-implemented all of them for consistency of pre-processing. In cases where there exist published results for a model, our tested results are close to the published records. The GBDT model is implemented using the `LightGBM` library (Ke et al. 2017). All the other models are implemented using the `PyTorch` library (Paszke et al. 2019). To reproduce our experiment results, the models' implementations and the exact values for all hyper-parameters can be found in another supplemental material, Code and Data Appendix.

For each dataset, all of the cross-validation splits, labeled, and unlabeled training data are obtained with a fixed random seed such that every model tested receives exactly the same training and testing conditions.

As all the datasets are for binary classification, the cross entropy loss was used for both supervised and semi-supervised training (for pre-training, the problem is binary classification in RTD and multi-class classification in MLM). For all deep models, the AdamW optimizer (Loshchilov and Hutter 2017) was used to update the model parameters, and a constant learning rate was applied throughout each training job. All models used early stopping based on the performance on the validation set and the early stopping patience (the number of epochs) is set as 15.

Hyper-parameters Search Space. The hyper-parameters tuned for the GBDT model were the number of leaves in the trees with a search space $\{x \in \mathbb{Z} | 5 \leq x \leq 50\}$, the minimum number of datapoints required to split a leaf in the trees with a search space $\{x \in \mathbb{Z} | 1 \leq x \leq 100\}$, the boosting learning rate with a search space $\{x = 5 \cdot 10^u, u \in \mathbb{U} | -3 \leq x \leq -1\}$, and the number of trees used for boosting with a search space $\{x \in \mathbb{Z} | 10 \leq x \leq 1000\}$.

For all of the deep models, the common hyper-parameters include the weight decay factor with a search space $\{x = 10^u, u \in \mathbb{U} | -6 \leq u \leq -1\}$, the learning rate with a search space $\{x = 10^u, u \in \mathbb{U} | -6 \leq u \leq -3\}$, the dropout probability with a search space $\{0, 0.1, 0.2, ...0.5\}$, and whether to

Table 5: Performance of TabTransformer with no column embedding, concatenation column embedding, and addition column embedding. The evaluation metric is mean ± standard deviation of relative AUCs (in percentage) over all 15 datasets. Larger value means better performance. The best model is bold for each row.

# of Transformers Layers	No Col Embed	Col Embed-Concat-1/4	Col Embed-Concat-1/8	Col Embed-Add
1	-0.59 ± 0.33	-2.01 ± 1.33	**-0.27 ± 0.21**	-1.11 ± 0.77
2	-0.59 ± 0.22	-0.37 ± 0.20	-0.14 ± 0.19	**0.34 ± 0.27**
3	-0.37 ± 0.19	0.04 ± 0.18	-0.02 ± 0.21	**0.21 ± 0.23**
6	0.54 ± 0.22	0.53 ± 0.24	**0.70 ± 0.17**	0.25 ± 0.23
12	0.66 ± 0.21	**1.05 ± 0.31**	0.73 ± 0.58	0.42 ± 0.39

Table 6: Fine-tuning performance of TabTransformer-RTD for different pre-training replacement value k. The number of labeled data points is 256. The evaluation metrics are mean ± standard deviation of (1) AUC score over 5 cross-validation splits for each dataset (in percentage); (2) relative AUCs over the 3 datasets (in percentage). Larger value means better performance. The best model is bold for each column.

Replacement value $k\%$	Adult	BankMarketing	1995_income	relative AUC (%)
15	58.1 ± 3.52	85.9 ± 1.62	**86.8 ± 1.35**	0.02 ± 0.10
30	**58.1 ± 3.15**	**86.1 ± 1.58**	86.7 ± 1.41	**0.08 ± 0.10**
50	57.9 ± 3.21	85.7 ± 1.93	86.7 ± 1.38	-0.10 ± 0.11

Table 7: Fine-tuning performance of TabTransformer-RTD for dynamic replacement, static replacement, and static replacement with a shared classifier. The number of labeled data points is 256. The evaluation metrics are mean ± standard deviation of (1) AUC score over 5 cross-validation splits for each dataset (in percentage) ; (2) relative AUCs over the 3 datasets (in percentage). Larger value means better performance. The best model is bold for each column.

	Adult	BankMarketing	1995_income	relative AUC (%)
Dynamic Replacement (Un-shared RTD classifiers)	**58.1 ± 3.52**	**85.9 ± 1.62**	**86.8 ± 1.35**	**0.81 ± 0.19**
Static Replacement (Un-shared RTD classifiers)	57.9 ± 2.93	83.9 ± 1.18	85.9 ± 1.60	-0.33 ± 0.15
Static Replacement (Shared RTD Classifiers)	57.5 ± 2.74	84.2 ± 1.46	86.0 ± 1.69	-0.49 ± 0.11

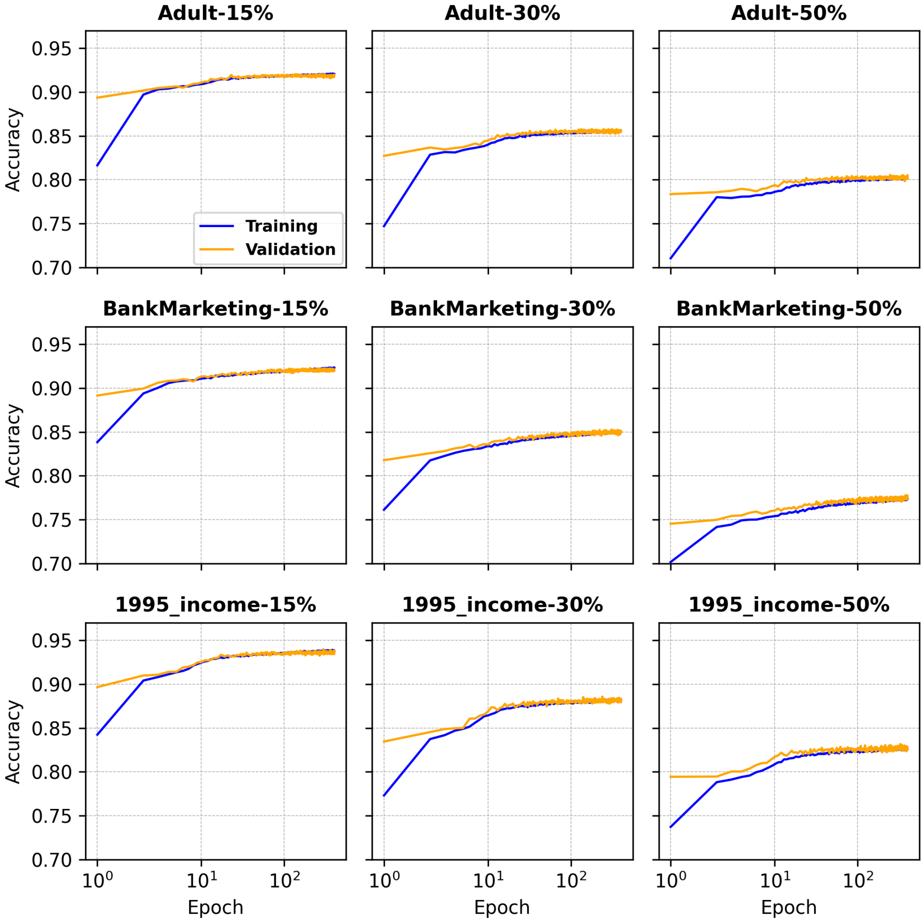

Figure 6: The pre-training curves of training and validation accuracy for the three different replacement value k for dataset *Adult*, *BankMarketing*, and *1995_income*.

one-hot encode categorical variables or train learnable embeddings.

For MLPs, they all used SELU activations (Klambauer et al. 2017) followed by batch normalization in each layer, and set the number of hidden layers as 2. The model-specific hyper-parameters tuned were the first hidden layer with a search space $\{x = m * l, m \in \mathbb{Z} | 1 \leq m \leq 8\}$ where l is the input size, and the second hidden layer with a search space $\{x = m * l, m \in \mathbb{Z} | 1 \leq m \leq 3\}$.

For TabTransformer, the hidden (embedding) dimension, the number of layers and the number of attention heads in the Transformer were fixed to 32, 6, and 8 respectively during the experiments. The MLP layer sizes were fixed to $\{4 \times l, 2 \times l\}$, where l was the size of its input. However, these parameters were optimally selected based on 50 rounds of HPO run on 5 datasets. The search spaces were the number of attention heads $\{2, 4, 8\}$, the hidden dimension $\{32, 64, 128, 256\}$, and the number of layers $\{1, 2, 3, 6, 12\}$. The search spaces of the first and second hidden layer in MLP are exactly the same as those in MLP model setting. The dimension of $\boldsymbol{c}_{\phi_i}$, ℓ was chosen as $d/8$ based on the ablation study in Appendix A.

For Sparse MLP (Prune), its implementation was the same as the MLP except that at every k epochs during training the fraction p of weights with the smallest magnitude were permanently set to zero. The model-specific hyper-parameters tuned were the fraction p with a search space $\{x = 5 \cdot 10^u, u \in \mathbb{U} | -2 \leq u \leq -1\}$. The number of layers and layer sizes are exactly the same as the setting in MLP. The parameter k is set as 10.

For TabNet model, we implemented exactly as described in Arik and Pfister (2019), though we also added the option to use a softmax attention instead of a sparsemax attention, and did not include the sparsification term in the loss func-

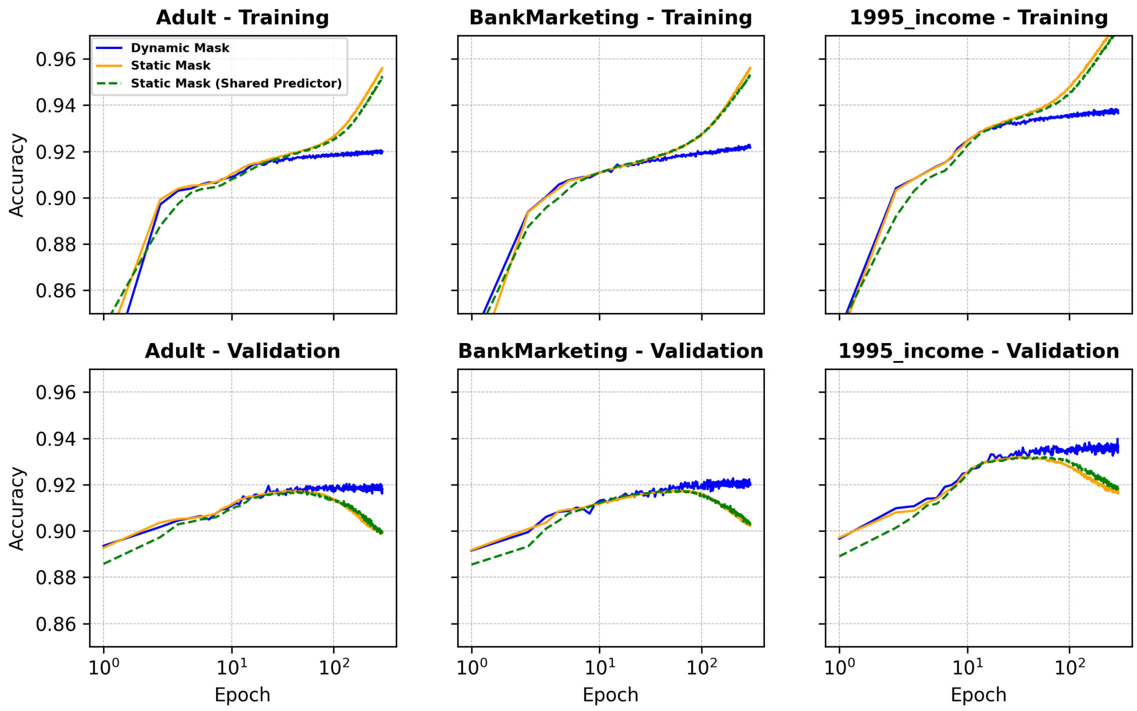

Figure 7: The pre-training curves of training and validation accuracy for dynamic mask, static mask, and static mask with a shared predictor (classifier) for dataset *Adult*, *BankMarketing*, and *1995_income*.

tion. The model-specific hyper-parameters tuned were the number of layers with a search space $\{x \in \mathbb{Z} | 3 \leq x \leq 10\}$, the hidden dimension $\{x \in \mathbb{Z} | 8 \leq x \leq 128\}$, and the sparse coefficient with a search space $\{x = 10^u, u \in \mathbb{U} | -6 \leq u \leq -2\}$.

For VIB model, we implemented it as described in Alemi, Fischer, and Dillon (2018). We used a diagonal covariance, with 10 samples from the variational distribution during training and 20 during testing. The model-specific hyper-parameters tuned were the number of hidden layers and layer sizes, with exactly the same search spaces as MLP, and the number of mixture components in the mixture of gaussians used in the marginal distribution with a search space $\{x \in \mathbb{Z} | 3 \leq x \leq 10\}$.

For MLP (DAE), its pre-training used swap noise as described in Jahrer (2018). The model-specific hyper-parameters were exactly the same as MLP.

For Pseudo Labeling (Lee 2013), since this method was combined with deep models such as MLP, TabTransformer and GBDT, the model-specific hyper-parameters were exactly the same as the corresponding deep models mentioned above. The unsupervised coefficient α is chosen as $\alpha_f = 3, T_1 = 30, T_2 = 70$.

For Entropy Regularization (Grandvalet and Bengio 2006), it is the same as Pseudo Labeling. The additional model-specific hyper-parameter was the positive Lagrange multiplier λ with a search space $\{0.1, 0.2, ..., 0.9\}$.

B.2 Feature Engineering

For categorical variables, the processing options include whether to one-hot encode versus learn a parametric embedding, what embedding dimension to use, and how to apply dropout regularization (whether to drop vector elements or whole embeddings). In our experiments we found that learned embeddings nearly always improved performance as long as the cardinality of the categorical variable is significantly less than the number of data points, otherwise the feature is merely a means for the model to overfit.

For scalar variables, the processing options include how to re-scale the variable (via quantiles, normalization, or log scaling) or whether to quantize the feature and treat it like a categorical variable. While we have not explored this idea fully, the best strategy is likely to use all the different types of encoding in parallel, turning each scalar feature into three re-scaled features and one categorical feature. Unlike learning embeddings for high-cardinality categorical features, adding potentially-redundant encodings for scalar variables should not lead to overfitting, but can make the difference between a feature being useful or not.

For text variables, we simply encodes the number of words and character in the text.

C Appendix: Benchmark Dataset Information and Experiment Results

Table 8: Benchmark datasets. All datasets are binary classification tasks. Positive Class% is the fraction of data points that belongs to the positive class.

Dataset Name	N Datapoints	N Features	Positive Class%
1995_income	32561	14	24.1
adult	34190	25	85.4
albert	425240	79	50.0
bank_marketing	45211	16	11.7
blastchar	7043	20	26.5
dota2games	92650	117	52.7
fabert	8237	801	11.3
hcdr_main	307511	120	8.1
htru2	17898	8	9.2
insurance_co	5822	85	6.0
jannis	83733	55	2.0
jasmine	2984	145	50.0
online_shoppers	12330	17	15.5
philippine	5832	309	50.0
qsar_bio	1055	41	33.7
seismicbumps	2583	18	6.6
shrutime	10000	11	20.4
spambase	4601	57	39.4
sylvine	5124	20	50.0
volkert	58310	181	12.7

Table 9: Benchmark Dataset Links.

Dataset Name	URL
1995_income	https://www.kaggle.com/lodetomasi1995/income-classification
adult	http://automl.chalearn.org/data
albert	http://automl.chalearn.org/data
bank_marketing	https://archive.ics.uci.edu/ml/datasets/bank+marketing
blastchar	https://www.kaggle.com/blastchar/telco-customer-churn
dota2games	https://archive.ics.uci.edu/ml/datasets/Dota2+Games+Results
fabert	http://automl.chalearn.org/data
hcdr_main	https://www.kaggle.com/c/home-credit-default-risk
htru2	https://archive.ics.uci.edu/ml/datasets/HTRU2
insurance_co	https://archive.ics.uci.edu/ml/datasets/Insurance+Company+Benchmark+%28COIL+2000%29
jannis	http://automl.chalearn.org/data
jasmine	http://automl.chalearn.org/data
online_shoppers	https://archive.ics.uci.edu/ml/datasets/Online+Shoppers+Purchasing+Intention+Dataset
philippine	http://automl.chalearn.org/data
qsar_bio	https://archive.ics.uci.edu/ml/datasets/QSAR+biodegradation
seismicbumps	https://archive.ics.uci.edu/ml/datasets/seismic-bumps
shrutime	https://www.kaggle.com/shrutimechlearn/churn-modelling
spambase	https://archive.ics.uci.edu/ml/datasets/Spambase
sylvine	http://automl.chalearn.org/data
volkert	http://automl.chalearn.org/data

Table 10: AUC score for semi-supervised learning models on all datasets with **50** fine-tune data points. Values are the mean over 5 cross-validation splits, plus or minus the standard deviation. Larger values means better result.

Dataset	N Datapoints	N Features	Positive Class%	Best Model	TabTransformer-RTD	TabTransformer-MLM	MLP (ER)
albert	425240	79	50.0	TabTransformer-MLM	0.644 ± 0.015	0.647 ± 0.019	0.612 ± 0.017
hcdr_main	307511	120	8.1	MLP (DAE)	0.592 ± 0.047	0.596 ± 0.047	0.602 ± 0.033
dota2games	92650	117	52.7	TabTransformer-MLM	0.526 ± 0.009	0.538 ± 0.011	0.519 ± 0.007
jannis	83733	55	2.0	TabTransformer-RTD	0.684 ± 0.055	0.665 ± 0.056	0.621 ± 0.022
volkert	58310	181	1.0	TabTransformer-RTD	0.693 ± 0.046	0.689 ± 0.042	0.657 ± 0.028
bank_marketing	45211	16	11.7	MLP (PL)	0.771 ± 0.046	0.735 ± 0.040	0.792 ± 0.039
adult	34190	25	85.4	MLP (DAE)	0.580 ± 0.012	0.613 ± 0.014	0.609 ± 0.005
1995_income	32561	14	24.1	TabTransformer-MLM	0.840 ± 0.029	0.862 ± 0.018	0.839 ± 0.034
htru2	17898	8	9.2	MLP (DAE)	0.956 ± 0.007	0.958 ± 0.009	0.969 ± 0.012
online_shoppers	12330	17	15.5	MLP (DAE)	0.790 ± 0.013	0.780 ± 0.024	0.855 ± 0.019
shrutime	10000	11	20.4	TabTransformer-RTD	0.752 ± 0.019	0.741 ± 0.019	0.725 ± 0.032
fabert	8237	801	11.3	MLP (PL)	0.535 ± 0.027	0.525 ± 0.019	0.572 ± 0.019
blastchar	7043	20	26.5	TabTransformer-MLM	0.806 ± 0.018	0.822 ± 0.009	0.803 ± 0.021
philippine	5832	309	50.0	TabTransformer-RTD	0.739 ± 0.027	0.729 ± 0.035	0.722 ± 0.031
insurance_co	5822	85	6.0	MLP (PL)	0.601 ± 0.056	0.573 ± 0.077	0.575 ± 0.063
sylvine	5124	20	50.0	MLP (PL)	0.872 ± 0.031	0.898 ± 0.030	0.930 ± 0.015
spambase	4601	57	39.4	MLP (ER)	0.949 ± 0.005	0.945 ± 0.011	0.957 ± 0.008
jasmine	2984	145	50.0	TabTransformer-MLM	0.821 ± 0.019	0.837 ± 0.019	0.830 ± 0.022
seismicbumps	2583	18	6.6	TabTransformer (ER)	0.740 ± 0.088	0.738 ± 0.068	0.712 ± 0.074
qsar_bio	1055	41	33.7	MLP (DAE)	0.875 ± 0.028	0.869 ± 0.036	0.880 ± 0.022

Table 11: (Continued) AUC score for semi-supervised learning models on all datasets with **50** fine-tune data points. Values are the mean over 5 cross-validation splits, plus or minus the standard deviation. Larger values means better result.

Dataset	MLP (PL)	TabTransformer (ER)	TabTransformer (PL)	MLP (DAE)	GBDT (PL)
albert	0.607 ± 0.013	0.580 ± 0.017	0.587 ± 0.012	0.612 ± 0.014	0.547 ± 0.032
hcdr_main	0.599 ± 0.038	0.581 ± 0.023	0.570 ± 0.031	0.620 ± 0.028	0.531 ± 0.024
dota2games	0.520 ± 0.006	0.516 ± 0.009	0.519 ± 0.008	0.516 ± 0.004	0.505 ± 0.008
jannis	0.623 ± 0.035	0.582 ± 0.035	0.604 ± 0.013	0.626 ± 0.023	0.519 ± 0.047
volkert	0.653 ± 0.035	0.635 ± 0.024	0.639 ± 0.040	0.629 ± 0.019	0.525 ± 0.018
bank_marketing	0.805 ± 0.036	0.744 ± 0.063	0.767 ± 0.058	0.786 ± 0.055	0.688 ± 0.057
adult	0.605 ± 0.021	0.568 ± 0.012	0.582 ± 0.024	0.616 ± 0.010	0.519 ± 0.024
1995_income	0.819 ± 0.042	0.813 ± 0.045	0.822 ± 0.048	0.811 ± 0.042	0.685 ± 0.084
htru2	0.970 ± 0.012	0.955 ± 0.007	0.951 ± 0.009	0.973 ± 0.003	0.919 ± 0.021
online_shoppers	0.848 ± 0.021	0.816 ± 0.036	0.818 ± 0.028	0.858 ± 0.019	0.818 ± 0.032
shrutime	0.715 ± 0.044	0.748 ± 0.035	0.739 ± 0.034	0.683 ± 0.055	0.651 ± 0.093
fabert	0.577 ± 0.027	0.504 ± 0.020	0.516 ± 0.020	0.552 ± 0.013	0.534 ± 0.016
blastchar	0.799 ± 0.025	0.799 ± 0.013	0.792 ± 0.025	0.817 ± 0.016	0.729 ± 0.053
philippine	0.725 ± 0.022	0.689 ± 0.046	0.703 ± 0.050	0.717 ± 0.022	0.628 ± 0.085
insurance_co	0.601 ± 0.057	0.575 ± 0.066	0.592 ± 0.080	0.522 ± 0.052	0.560 ± 0.081
sylvine	0.939 ± 0.013	0.891 ± 0.022	0.904 ± 0.027	0.925 ± 0.010	0.914 ± 0.021
spambase	0.951 ± 0.010	0.947 ± 0.006	0.948 ± 0.006	0.949 ± 0.012	0.899 ± 0.039
jasmine	0.819 ± 0.021	0.825 ± 0.024	0.819 ± 0.018	0.812 ± 0.029	0.755 ± 0.016
seismicbumps	0.678 ± 0.106	0.745 ± 0.080	0.713 ± 0.090	0.724 ± 0.049	0.601 ± 0.071
qsar_bio	0.875 ± 0.015	0.851 ± 0.041	0.835 ± 0.053	0.888 ± 0.022	0.804 ± 0.057

Table 12: AUC score for semi-supervised learning models on all datasets with **200** fine-tune data points. Values are the mean over 5 cross-validation splits, plus or minus the standard deviation. Larger values means better result.

Dataset	N Datapoints	N Features	Positive Class%	Best Model	TabTransformer-RTD	TabTransformer-MLM	MLP (ER)
albert	425240	79	50.0	TabTransformer-MLM	0.699 ± 0.011	0.701 ± 0.014	0.642 ± 0.020
hcdr_main	307511	120	8.1	TabTransformer-MLM	0.655 ± 0.040	0.668 ± 0.028	0.639 ± 0.027
dota2games	92650	117	52.7	TabTransformer-MLM	0.536 ± 0.012	0.549 ± 0.008	0.527 ± 0.012
jannis	83733	55	2.0	TabTransformer-RTD	0.713 ± 0.037	0.692 ± 0.024	0.665 ± 0.024
volkert	58310	181	12.7	TabTransformer-RTD	0.753 ± 0.022	0.742 ± 0.023	0.696 ± 0.033
bank_marketing	45211	16	11.7	MLP (PL)	0.854 ± 0.020	0.838 ± 0.010	0.860 ± 0.008
adult	34190	25	85.4	MLP (ER)	0.596 ± 0.023	0.614 ± 0.012	0.623 ± 0.017
1995_income	32561	14	24.1	TabTransformer-MLM	0.866 ± 0.014	0.875 ± 0.011	0.868 ± 0.007
htru2	17898	8	9.2	MLP (DAE)	0.961 ± 0.008	0.963 ± 0.009	0.974 ± 0.007
online_shoppers	12330	17	15.5	MLP (ER)	0.834 ± 0.015	0.838 ± 0.024	0.876 ± 0.019
shrutime	10000	11	20.4	TabTransformer-RTD	0.805 ± 0.017	0.783 ± 0.024	0.773 ± 0.013
fabert	8237	801	11.3	MLP (ER)	0.556 ± 0.023	0.561 ± 0.028	0.600 ± 0.046
blastchar	7043	20	26.5	TabTransformer-MLM	0.831 ± 0.010	0.841 ± 0.014	0.829 ± 0.010
philippine	5832	309	50.0	TabTransformer-RTD	0.757 ± 0.017	0.754 ± 0.016	0.732 ± 0.024
insurance_co	5822	85	6.0	TabTransformer (ER)	0.667 ± 0.062	0.640 ± 0.043	0.601 ± 0.059
sylvine	5124	20	50.0	MLP (PL)	0.939 ± 0.008	0.948 ± 0.006	0.957 ± 0.008
spambase	4601	57	39.4	MLP (ER)	0.957 ± 0.006	0.955 ± 0.010	0.968 ± 0.009
jasmine	2984	145	50.0	TabTransformer-RTD	0.843 ± 0.016	0.843 ± 0.028	0.831 ± 0.019
seismicbumps	2583	18	6.6	TabTransformer-RTD	0.738 ± 0.063	0.708 ± 0.083	0.694 ± 0.088
qsar_bio	1055	41	33.7	TabTransformer-RTD	0.896 ± 0.018	0.889 ± 0.030	0.895 ± 0.026

Table 13: (Continued) AUC score for semi-supervised learning models on all datasets with **200** fine-tune data points. Values are the mean over 5 cross-validation splits, plus or minus the standard deviation. Larger values means better result.

Dataset	MLP (PL)	TabTransformer (ER)	TabTransformer (PL)	MLP (DAE)	GBDT (PL)
albert	0.638 ± 0.024	0.630 ± 0.025	0.630 ± 0.021	0.646 ± 0.023	0.628 ± 0.015
hcdr_main	0.631 ± 0.019	0.611 ± 0.030	0.605 ± 0.021	0.636 ± 0.027	0.579 ± 0.039
dota2games	0.527 ± 0.014	0.528 ± 0.017	0.525 ± 0.011	0.528 ± 0.012	0.506 ± 0.008
jannis	0.667 ± 0.036	0.619 ± 0.024	0.637 ± 0.026	0.659 ± 0.020	0.525 ± 0.030
volkert	0.693 ± 0.028	0.694 ± 0.002	0.689 ± 0.015	0.672 ± 0.015	0.612 ± 0.042
bank_marketing	0.866 ± 0.008	0.853 ± 0.016	0.858 ± 0.009	0.863 ± 0.009	0.802 ± 0.012
adult	0.616 ± 0.014	0.582 ± 0.026	0.584 ± 0.017	0.611 ± 0.027	0.572 ± 0.040
1995_income	0.869 ± 0.009	0.848 ± 0.024	0.852 ± 0.015	0.865 ± 0.011	0.822 ± 0.020
htru2	0.974 ± 0.007	0.955 ± 0.007	0.954 ± 0.007	0.974 ± 0.010	0.946 ± 0.022
online_shoppers	0.873 ± 0.030	0.857 ± 0.014	0.853 ± 0.017	0.873 ± 0.021	0.846 ± 0.019
shrutime	0.774 ± 0.018	0.803 ± 0.022	0.803 ± 0.024	0.763 ± 0.018	0.750 ± 0.050
fabert	0.595 ± 0.048	0.530 ± 0.027	0.522 ± 0.024	0.580 ± 0.020	0.573 ± 0.026
blastchar	0.829 ± 0.011	0.823 ± 0.011	0.823 ± 0.011	0.832 ± 0.013	0.783 ± 0.017
philippine	0.733 ± 0.018	0.736 ± 0.018	0.739 ± 0.024	0.720 ± 0.020	0.729 ± 0.024
insurance_co	0.616 ± 0.045	0.715 ± 0.038	0.680 ± 0.034	0.612 ± 0.024	0.630 ± 0.087
sylvine	0.961 ± 0.004	0.951 ± 0.009	0.950 ± 0.010	0.955 ± 0.009	0.957 ± 0.005
spambase	0.965 ± 0.008	0.962 ± 0.006	0.960 ± 0.008	0.964 ± 0.009	0.957 ± 0.013
jasmine	0.839 ± 0.013	0.824 ± 0.024	0.841 ± 0.016	0.842 ± 0.014	0.826 ± 0.013
seismicbumps	0.684 ± 0.071	0.723 ± 0.080	0.727 ± 0.081	0.673 ± 0.070	0.603 ± 0.023
qsar_bio	0.892 ± 0.033	0.871 ± 0.036	0.876 ± 0.032	0.891 ± 0.018	0.855 ± 0.035

Table 14: AUC score for semi-supervised learning models on all datasets with **500** fine-tune data points. Values are the mean over 5 cross-validation splits, plus or minus the standard deviation. Larger values means better result.

Dataset	N Datapoints	N Features	Positive Class%	Best Model	TabTransformer-RTD	TabTransformer-MLM	MLP (ER)
albert	425240	79	50.0	TabTransformer-RTD	0.711 ± 0.004	0.707 ± 0.006	0.666 ± 0.008
hcdr_main	307511	120	8.1	TabTransformer-MLM	0.690 ± 0.038	0.698 ± 0.033	0.653 ± 0.019
dota2games	92650	117	52.7	TabTransformer-MLM	0.548 ± 0.008	0.557 ± 0.003	0.543 ± 0.008
jannis	83733	55	2.0	TabTransformer-RTD	0.747 ± 0.015	0.720 ± 0.018	0.707 ± 0.036
volkert	58310	181	12.7	TabTransformer-RTD	0.771 ± 0.016	0.760 ± 0.015	0.723 ± 0.016
bank_marketing	45211	16	11.7	TabTransformer-RTD	0.879 ± 0.012	0.866 ± 0.016	0.869 ± 0.012
adult	34190	25	85.4	MLP (PL)	0.625 ± 0.011	0.647 ± 0.008	0.644 ± 0.015
1995_income	32561	14	24.1	MLP (DAE)	0.874 ± 0.008	0.880 ± 0.007	0.878 ± 0.002
htru2	17898	8	9.2	MLP (DAE)	0.964 ± 0.009	0.966 ± 0.009	0.973 ± 0.010
online_shoppers	12330	17	15.5	MLP (ER)	0.859 ± 0.009	0.861 ± 0.014	0.888 ± 0.012
shrutime	10000	11	20.4	TabTransformer-RTD	0.831 ± 0.017	0.815 ± 0.004	0.793 ± 0.017
fabert	8237	801	11.3	MLP (ER)	0.618 ± 0.014	0.609 ± 0.019	0.621 ± 0.032
blastchar	7043	20	26.5	TabTransformer-RTD	0.840 ± 0.013	0.839 ± 0.015	0.829 ± 0.013
philippine	5832	309	50.0	TabTransformer-MLM	0.769 ± 0.028	0.772 ± 0.017	0.734 ± 0.024
insurance_co	5822	85	6.0	TabTransformer (ER)	0.688 ± 0.039	0.642 ± 0.029	0.659 ± 0.023
sylvine	5124	20	50.0	MLP (PL)	0.955 ± 0.007	0.959 ± 0.006	0.967 ± 0.003
spambase	4601	57	39.4	MLP (ER)	0.966 ± 0.007	0.968 ± 0.008	0.975 ± 0.004
jasmine	2984	145	50.0	TabTransformer-RTD	0.847 ± 0.016	0.844 ± 0.011	0.837 ± 0.019
seismicbumps	2583	18	6.6	TabTransformer-RTD	0.758 ± 0.081	0.729 ± 0.069	0.682 ± 0.123
qsar_bio	1055	41	33.7	MLP (DAE)	0.909 ± 0.024	0.889 ± 0.038	0.918 ± 0.023

Table 15: (Continued) AUC score for semi-supervised learning models on all datasets with **500** fine-tune data points. Values are the mean over 5 cross-validation splits, plus or minus the standard deviation. Larger values means better result.

Dataset	MLP (PL)	TabTransformer (ER)	TabTransformer (PL)	MLP (DAE)	GBDT (PL)
albert	0.662 ± 0.007	0.664 ± 0.011	0.643 ± 0.029	0.666 ± 0.006	0.653 ± 0.011
hcdr_main	0.645 ± 0.022	0.623 ± 0.036	0.636 ± 0.031	0.657 ± 0.033	0.607 ± 0.035
dota2games	0.544 ± 0.010	0.538 ± 0.009	0.541 ± 0.010	0.542 ± 0.012	0.505 ± 0.005
jannis	0.698 ± 0.033	0.662 ± 0.007	0.660 ± 0.024	0.693 ± 0.024	0.521 ± 0.045
volkert	0.722 ± 0.012	0.712 ± 0.016	0.705 ± 0.021	0.712 ± 0.016	0.705 ± 0.016
bank_marketing	0.876 ± 0.017	0.863 ± 0.008	0.868 ± 0.016	0.874 ± 0.012	0.838 ± 0.019
adult	0.651 ± 0.012	0.618 ± 0.023	0.618 ± 0.021	0.654 ± 0.016	0.647 ± 0.030
1995_income	0.880 ± 0.003	0.868 ± 0.008	0.869 ± 0.007	0.882 ± 0.001	0.839 ± 0.013
htru2	0.974 ± 0.007	0.960 ± 0.010	0.960 ± 0.008	0.976 ± 0.006	0.949 ± 0.007
online_shoppers	0.885 ± 0.021	0.861 ± 0.011	0.860 ± 0.013	0.885 ± 0.019	0.865 ± 0.011
shrutime	0.800 ± 0.015	0.825 ± 0.013	0.822 ± 0.016	0.804 ± 0.015	0.788 ± 0.019
fabert	0.596 ± 0.046	0.573 ± 0.048	0.578 ± 0.033	0.617 ± 0.042	0.585 ± 0.025
blastchar	0.833 ± 0.013	0.834 ± 0.013	0.832 ± 0.011	0.833 ± 0.012	0.795 ± 0.021
philippine	0.740 ± 0.023	0.746 ± 0.020	0.735 ± 0.015	0.739 ± 0.017	0.749 ± 0.026
insurance_co	0.646 ± 0.048	0.710 ± 0.040	0.666 ± 0.060	0.612 ± 0.013	0.672 ± 0.037
sylvine	0.968 ± 0.003	0.958 ± 0.005	0.958 ± 0.003	0.967 ± 0.003	0.967 ± 0.006
spambase	0.973 ± 0.005	0.968 ± 0.007	0.967 ± 0.006	0.972 ± 0.006	0.972 ± 0.005
jasmine	0.833 ± 0.009	0.833 ± 0.021	0.838 ± 0.018	0.842 ± 0.011	0.838 ± 0.022
seismicbumps	0.677 ± 0.103	0.687 ± 0.100	0.735 ± 0.081	0.696 ± 0.112	0.666 ± 0.063
qsar_bio	0.914 ± 0.032	0.894 ± 0.036	0.895 ± 0.035	0.925 ± 0.034	0.908 ± 0.024

Table 16: AUC score for supervised learning models on all datasets. Values are the mean over 5 cross-validation splits, plus or minus the standard deviation. Larger values means better result.

Dataset	N Datapoints	N Features	Positive Class%	Best Model	Logistic Regression	GBDT
ds_name						
albert	425240	79	50.0	GBDT	0.726 ± 0.001	0.763 ± 0.001
hcdr_main	307511	120	8.1	GBDT	0.747 ± 0.004	0.756 ± 0.004
dota2games	92650	117	52.7	Logistic Regression	0.634 ± 0.003	0.621 ± 0.004
bank_marketing	45211	16	11.7	TabTransformer	0.911 ± 0.005	0.933 ± 0.003
adult	34190	25	85.4	GBDT	0.721 ± 0.010	0.756 ± 0.011
1995_income	32561	14	24.1	TabTransformer	0.899 ± 0.002	0.906 ± 0.002
online_shoppers	12330	17	15.5	GBDT	0.908 ± 0.015	0.930 ± 0.008
shrutime	10000	11	20.4	GBDT	0.828 ± 0.013	0.859 ± 0.009
blastchar	7043	20	26.5	GBDT	0.844 ± 0.010	0.847 ± 0.016
philippine	5832	309	50.0	TabTransformer	0.725 ± 0.022	0.812 ± 0.013
insurance_co	5822	85	6.0	TabTransformer	0.736 ± 0.023	0.732 ± 0.022
spambase	4601	57	39.4	GBDT	0.947 ± 0.008	0.987 ± 0.005
jasmine	2984	145	50.0	GBDT	0.846 ± 0.017	0.862 ± 0.008
seismicbumps	2583	18	6.6	GBDT	0.749 ± 0.068	0.756 ± 0.084
qsar_bio	1055	41	33.7	TabTransformer	0.847 ± 0.037	0.913 ± 0.031

Table 17: (Continued) AUC score for supervised learning models on all datasets. Values are the mean over 5 cross-validation splits, plus or minus the standard deviation. Larger values means better result.

	MLP	Sparse MLP	TabTransformer	TabNet	VIB
ds_name					
albert	0.740 ± 0.001	0.741 ± 0.001	0.757 ± 0.002	0.705 ± 0.005	0.737 ± 0.001
hcdr_main	0.743 ± 0.004	0.753 ± 0.004	0.751 ± 0.004	0.711 ± 0.006	0.745 ± 0.005
dota2games	0.631 ± 0.002	0.633 ± 0.004	0.633 ± 0.002	0.529 ± 0.025	0.628 ± 0.003
bank_marketing	0.929 ± 0.003	0.926 ± 0.007	0.934 ± 0.004	0.885 ± 0.017	0.920 ± 0.005
adult	0.725 ± 0.010	0.740 ± 0.007	0.737 ± 0.009	0.663 ± 0.016	0.733 ± 0.009
1995_income	0.905 ± 0.003	0.904 ± 0.004	0.906 ± 0.003	0.875 ± 0.006	0.904 ± 0.003
online_shoppers	0.919 ± 0.010	0.922 ± 0.011	0.927 ± 0.010	0.888 ± 0.020	0.907 ± 0.012
shrutime	0.846 ± 0.013	0.828 ± 0.007	0.856 ± 0.005	0.785 ± 0.024	0.833 ± 0.011
blastchar	0.839 ± 0.010	0.842 ± 0.015	0.835 ± 0.014	0.816 ± 0.014	0.842 ± 0.012
philippine	0.821 ± 0.020	0.764 ± 0.018	0.834 ± 0.018	0.721 ± 0.008	0.757 ± 0.018
insurance_co	0.697 ± 0.027	0.705 ± 0.054	0.744 ± 0.009	0.630 ± 0.061	0.647 ± 0.028
spambase	0.984 ± 0.004	0.980 ± 0.009	0.985 ± 0.005	0.975 ± 0.008	0.983 ± 0.004
jasmine	0.851 ± 0.015	0.856 ± 0.013	0.853 ± 0.015	0.816 ± 0.017	0.847 ± 0.017
seismicbumps	0.735 ± 0.028	0.699 ± 0.074	0.751 ± 0.096	0.701 ± 0.051	0.681 ± 0.084
qsar_bio	0.910 ± 0.037	0.916 ± 0.036	0.918 ± 0.038	0.860 ± 0.038	0.914 ± 0.028