Binning as a Pretext Task: Improving Self-Supervised Learning in Tabular Domains

Kyungeun Lee [1] Ye Seul Sim [1] Hye-Seung Cho [1] Moonjung Eo [1] Suhee Yoon [1] Sanghyu Yoon [1] Woohyung Lim [1]

Abstract

The ability of deep networks to learn superior representations hinges on leveraging the proper inductive biases, considering the inherent properties of datasets. In tabular domains, it is critical to effectively handle heterogeneous features (both categorical and numerical) in a unified manner and to grasp irregular functions like piecewise constant functions. To address the challenges in the self-supervised learning framework, we propose a novel pretext task based on the classical *binning* method. The idea is straightforward: *reconstructing the bin indices* (either orders or classes) rather than the original values. This pretext task provides the encoder with an inductive bias to capture the irregular dependencies, mapping from continuous inputs to discretized bins, and mitigates the feature heterogeneity by setting all features to have category-type targets. Our empirical investigations ascertain several advantages of binning: capturing the irregular function, compatibility with encoder architecture and additional modifications, standardizing all features into equal sets, grouping similar values within a feature, and providing ordering information. Comprehensive evaluations across diverse tabular datasets corroborate that our method consistently improves tabular representation learning performance for a wide range of downstream tasks. The codes are available in `https://github.com/kyungeun-lee/tabularbinning`.

1. Introduction

Tabular datasets are ubiquitous across diverse applications from financial markets and healthcare diagnostics to e-commerce personalization and manufacturing process automation. These datasets are structured with rows representing individual samples and columns representing heterogeneous features—a combination of categorical and numerical features—and they serve as the foundation for myriad analyses. Despite the wide applicability of tabular data, research into leveraging deep networks to harness the inherent properties of such datasets is still in its nascent stage. In contrast, tree-based machine learning algorithms like XGBoost (Chen & Guestrin, 2016) and CatBoost (Prokhorenkova et al., 2018) have consistently demonstrated prowess in discerning the nuances of tabular domains, outperforming deep networks even those with a larger model capacity and specialized modules (Arik & Pfister, 2021; Gorishniy et al., 2021; Grinsztajn et al., 2022; Rubachev et al., 2022). The consistent edge held by tree models fuels the exploration of how their advantageous biases can be adapted for deep networks.

Recently, the quest to boost the performance of deep networks on tabular data has gained momentum. A fundamental challenge is the inherent heterogeneity of tabular datasets, encompassing both categorical and numerical features (Popov et al., 2019; Borisov et al., 2022; Yan et al., 2023). To mitigate the feature discrepancies in deep networks, previous studies proposed using an additional module like a feature tokenizer (Gorishniy et al., 2021) and an abstract layer (Chen et al., 2022). Concurrently, some research has explored ways to infuse the proven strengths of tree-based models into deep networks. For instance, Grinsztajn et al. (2022) observed that deep networks tend to prefer overly smooth solutions and struggle with modeling irregularities like piecewise constant functions, in contrast to the tree-based models. To address this challenge, Gorishniy et al. (2022) introduced a novel approach combining piecewise linear encoding during preprocessing and periodic activation functions. Although these advancements have led to enhanced performance on several tabular data problems, they have predominantly been explored within a supervised

[1]LG AI Research, Seoul, Republic of Korea. Correspondence to: Woohyung Lim <w.lim@lgresearch.ai>.

Proceedings of the 41st International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024. Copyright 2024 by the author(s).

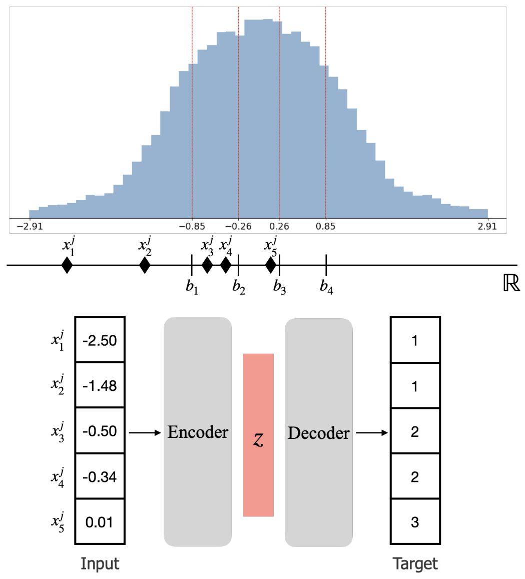


Figure 1: Binning as a pretext task. Bins are determined based on the distribution of the training dataset for each feature. The inputs are passed into the encoder network, then the decoder network predicts the bin indices which can be ordinal when the pretext task is the regression or nominal when the pretext task is the classification.

learning framework, where they still fall short of outperforming simple tree-based methods (Gorishniy et al., 2021; Grinsztajn et al., 2022; McElfresh et al., 2023).

In this study, we address the challenge of *unsupervised* tabular deep learning where the tree-based methods are fundamentally inapplicable. To this end, we propose a novel pretext task based on the classical *binning* method for auto-encoding-based self-supervised learning (SSL). Our approach is straightforward: *reconstructing bin indices* rather than reconstructing the raw values, as illustrated in Figure 1. Once numerical features are discretized into bins based on the quantiles of the training dataset, we optimize the encoder and decoder networks to accurately predict the bin indices given original inputs. Despite its simplicity, binning as a pretext task offers several advantages for tabular deep learning. By setting the discretized bins as targets for the pretext task, we can employ the inductive bias of capturing the irregular functions and mitigating the discrepancy between features. The binning procedure allows grouping the nearby samples based on the distribution of the training dataset, so the learned representations should be robust to the minor errors that can yield spurious patterns. It also facilitates standardizing all features into equal sets, thereby preventing any uninformative features from dominating during SSL. Furthermore, our approach is compatible with any other modifications, including the choice of deep architectures and input transformation functions.

Based on the extensive experiments on 25 public datasets, we found that the binning task consistently improves the SSL performance on diverse downstream tasks, even though we simply changed the targets during SSL from the continuous to the discretized bins. Finally, we found that the binning task can be not only an effective objective function for fully unsupervised learning but also beneficial as the pretraining strategy to achieve state-of-the-art performance, surpassing both tree-based and other supervised deep learning methods across a wide range of tabular data problems.

Our main contributions can be summarized as follows. First, we suggest binning as a new pretext task for SSL in tabular domains, compatible with any modifications. Second, we conduct extensive experiments on 25 public tabular datasets focusing on the various input transformation methods and SSL objectives. Finally, we consistently achieve the best performance both in unsupervised and supervised learning frameworks. The codes are available in `https://github.com/kyungeun-lee/tabularbinning`.

2. Related Works

Tabular deep learning: In recent years, there has been a large number of deep learning research on a tabular domain: developing new deep architectures (Popov et al., 2019; Badirli et al., 2020; Huang et al., 2020; Wang et al., 2021; Arik & Pfister, 2021; Gorishniy et al., 2021; Chen et al., 2022; Hollmann et al., 2022; Zhu et al., 2023; Kotelnikov et al., 2023; Chen et al., 2023a); or representing the heterogeneous nature of tabular features as the graphs (Yan et al., 2023; Chen et al., 2023b); or adopting new activation function (Gorishniy et al., 2022). Despite these advancements, ensembles of decision trees, such as GBDTs (Gradient Boosting Decision Trees), continue to serve as competitive baselines (Arik & Pfister, 2021; Gorishniy et al., 2021; Grinsztajn et al., 2022; Rubachev et al., 2022; McElfresh et al., 2023; Beyazit et al., 2023). In this paper, our goal is to suggest a new pretext task for self-supervised learning in tabular domains, so we focus on architectures directly inspired by classic deep models, in particular MLPs and FT-Transformers (Gorishniy et al., 2021), in addition to the state-of-the-art tabular deep learning model, such as T2G-Former (Yan et al., 2023).

Self-supervised learning in tabular domains: Self-supervised learning (SSL) aims to learn desirable representations without making use of annotation information. Recently, contrastive learning and auto-encoding have been two major choices in the tabular domain. Contrastive learning aims to model the similarity between two or more augmented views from the same sample, corresponding to the positive samples, and the dissimilarity between other samples, corresponding to the negative samples. Bahri et al. (2021); Ucar et al. (2021) have optimized contrastive loss after defining the positive and negative samples based on the data augmentation function, such as masking or cropping in feature dimension. Auto-encoding aims to reconstruct the original sample given its corrupted observation (Vincent et al., 2008). Compared to contrastive learning, auto-encoders can handle a mix of data types which can be beneficial for tasks involving heterogeneous datasets, like tabular data. Yoon et al. (2020); Huang et al. (2020); Majmundar et al. (2022) adopted the auto-encoding methods optimizing the reconstruction loss with or without the additional losses, such as corruption detection. In this study, we suggest a new SSL pretext task based on the auto-encoding approach.

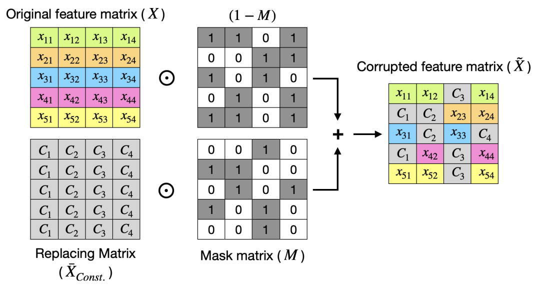
3. Backgrounds

In this section, we delve into the auto-encoding-based self-supervised learning framework in tabular domains focusing on two factors: transformation methods to tabular inputs and the objective functions in the auto-encoding-based SSL framework.

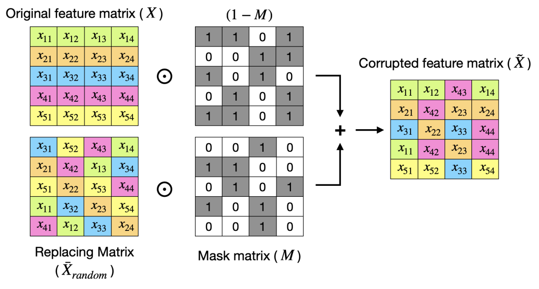
Input transformation: To ensure the encoder network does not simply learn an identity function, we employ transformation functions on the input that preserve the label-related information. For tabular datasets, only a few transformation functions have been suggested like masking (Yoon et al., 2020; Ucar et al., 2021; Majmundar et al., 2022) as illustrated in Figure 2 because all individual values can play a key role in determining the semantics and small changes can affect the context. Given a sample $x_i \in \mathbb{R}^d$ in dataset $\mathcal{D}$ where d is the number of features, $i \in [1, N]$, and N is the batch size, we randomly generate the masking vector m_i with the same size of x_i. Each element of the masking vector m_i is independently sampled from a Bernoulli distribution with probability $p_m \in [0, 1]$. To replace the masked values, the replacing vector $\bar{x}_i$ should be defined. In this study, we utilize two methods suggested in the previous studies (Yoon et al., 2020; Ucar et al., 2021; Majmundar et al., 2022).

- Constant (Figure 2a): $\bar{x}_{i,k}$ is set as the pre-determined constant value for all i. In this study, we use the average for each feature k in the training dataset.
- Random (Figure 2b): $\bar{x}_{i,k}$ is sampled from the other in-batch samples for a given feature. In other words, to replace the k-th feature of the i-th sample in the batch, we use the k-th feature of the i'-th sample in the same batch, and i' is sampled from the uniform distribution $\mathcal{U}\left(\frac{1}{N}\right)$.

Finally, the corrupted sample $\tilde{x}_i$ is formulated as $\tilde{x}_i = (\mathbf{1} - m_i) \odot x_i + m_i \odot \bar{x}_i$ where $\mathbf{1}$ is all-ones vector with the same size of x_i. The transformation procedure is stochastic and it provides randomness during training. When $p_m = 0$, m_i becomes the zero matrix, and the uncorrupted input $\tilde{x}_i = x_i$ is used for training.


Original feature matrix (X)
$(1-M)$
Corrupted feature matrix ($\tilde{X}$)
Replacing Matrix ($\bar{X}_{Const.}$)
Mask matrix (M)

(a) Replacing value = Constant


Original feature matrix (X)
$(1-M)$
Corrupted feature matrix ($\tilde{X}$)
Replacing Matrix ($\bar{X}_{random}$)
Mask matrix (M)

(b) Replacing value = Random

Figure 2: An illustration of two methods to generate the replacing vectors for masked features.

SSL objectives: Following the convention of SSL, the encoder f_e first transforms the corrupted sample $\tilde{x}_i$ to a representation z_i, then the decoder f_d will be introduced to learn the informative representation by optimizing the unsupervised loss $\mathcal{L}$. We can leverage which representation should be learned by introducing the specific pretext task. As a baseline, we consider two pretext tasks used in Yoon et al. (2020); Huang et al. (2020); Majmundar et al. (2022).

- Reconstructing the original values: One common approach is to reconstruct uncorrupted samples from their corrupted counterparts (Vincent et al., 2008). In this setup, the encoder attempts to impute the masked features by leveraging the correlations present in the non-masked features. The learned representations will involve the semantic-level information that is invariant to corruption. To this end, the decoder network is defined as $f_d^{\text{recon}} : Z \to \hat{X}$, and the corresponding loss is formulated as $\mathcal{L}_{\text{ValueRecon}} := \frac{1}{N} \sum_{i=1}^{N} ||x_i - f_d^{\text{recon}}(z_i)||_2^2$.
- Detecting the masked features: An auxiliary task that can facilitate the pretext task of reconstruction is predicting which features have been masked during the corruption

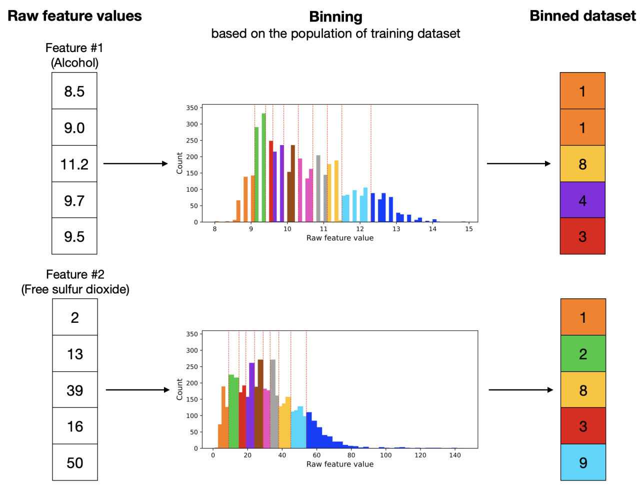


Figure 3: An example of binning (Dataset: Wine Quality (Cortez et al., 2009)). In the example, we set T as 10. For each feature, we implement the binning to include the same number of observations based on the training dataset. Finally, we use the binning indices as the targets for auto-encoding-based SSL. When we regard the bin indices as the classes without order information, the binning indices are converted into the one-hot vectors.

process of the input sample (Yoon et al., 2020). In this setup, the encoder attempts to leverage the inconsistency between feature values to identify the masked features, resulting in learned representations that capture abnormal patterns for a given input. Specifically, the method employs a binary cross-entropy loss, which can be formulated as $\mathcal{L}_{\text{MaskXent}} := -\frac{1}{N}\sum_{i=1}^{N} m_i \log f_d^{\text{mask}}(z_i) + (\mathbf{1} - m_i)\log(\mathbf{1} - f_d^{\text{mask}}(z_i))$ where the decoder network is defined as $f_d^{\text{mask}} : Z \to \hat{M}$.

We can optimize several loss functions simultaneously if we train several decoders that utilize z as the inputs. For example, Yoon et al. (2020) utilized the weighted sum of $\mathcal{L}_{\text{ValueRecon}}$ and $\mathcal{L}_{\text{MaskXent}}$.

4. Methods: Binning as a Pretext Task for Tabular SSL

Binning is a classical data preprocessing technique that quantizes a given numerical feature $x^j \in \mathbb{R}^{|\mathcal{D}|}$ into T discrete intervals, known as *bins* $B_t^j = [b_{t-1}^j, b_t^j)$ where $t \in [1, T]$ and $b_t^j \in \mathbb{R}$ is the bin boundaries. Binning is effective in transforming continuous features into discrete ones, mitigating minor errors in datasets like noise and outliers, and making the data distribution more manageable (Dougherty et al., 1995; Han et al., 2022).

In this study, we implement binning to establish targets for auto-encoding-based SSL. We anticipate the representations will be robust to the minor input variation in the same bins. Also, the deep networks can capture the irregularities akin to the decision-making process of tree-based models, which assign discrete leaves to each continuous sample because the pretext task corresponds to mapping continuous inputs to discretized bins. In addition, the binning approach helps mitigate feature heterogeneity by treating the targets for all features as the same category type during SSL.

Müller et al. (2021) proposed a similar approach in the context of Bayesian inference, addressing the well-known challenge that neural networks face in modeling continuous distributions. To overcome this issue, they pivoted towards utilizing discretized continuous distributions for accurately modeling posterior probability distributions. Their research revealed that integrating discretization into the objective function of deep networks not only enhances the effectiveness of the training process but is also theoretically established as a versatile technique capable of modeling any distribution. Similarly, recent researches (Stewart et al., 2023; Wu et al., 2023) underscore the utility of binning across various domains in deep learning. These studies highlight the substantial potential of binning in augmenting the capabilities of neural networks.

The binning procedure is described in Figure 3. We first determine the number of bins T as the design parameter. Then, we split the value range into the disjoint set of T intervals, $\left\{B_1^j, \ldots, B_T^j\right\}$, considering the number of observations in the training dataset $\mathcal{D}_{\text{train}}$ for each j-th feature x^j. Specifically, the bin boundaries b_t^j are determined according to the quantiles of $\frac{t}{T}$. (Alternative binning strategies are also discussed in Supplementary D.1.) When the number of unique values for x^j in the training dataset is less than T, each distinct value is assigned its own bin. Finally, we place each numerical feature x_i^j into the bin B_t^j, and we substitute the original values with the corresponding bin indices $t_i^j \in [1, T]$. Thus, we use the grouped ranks (or classes) instead of the raw values. We call the binned dataset as X_{Bin}.

The bin index of i-th sample and j-th feature, t_i^j, can be expressed as ordinal values or nominal classes. When we utilize the bin indices as ordinal values, we set the pretext task as reconstructing the bin indices based on the continuous inputs, and the corresponding *BinRecon loss* is defined as

$$\mathcal{L}_{\text{BinRecon}} := \frac{1}{N}\sum_{i=1}^{N} \left\| t_i - f_d^{\text{BinRecon}}(z_i) \right\|_2^2$$
$$\text{where } f_d^{\text{BinRecon}} : Z \to \hat{X}_{\text{Bin}}.$$

When we utilize the bin indices as nominal classes, we convert the bin index t_i^j into the one-hot vector $\mathbf{u}_i^j = [u_1, u_2, \ldots, u_T]$ where $u_v = 1$ when $v = t_i^j$ and $u_v = 0$

otherwise. Then, we set the pretext task as predicting the bin indices as classes by optimizing the *BinXent loss*, defined as the multi-class cross-entropy loss for each feature.

$$\mathcal{L}_{\text{BinXent}} := -\frac{1}{Nd}\sum_{i=1}^{N}\sum_{j=1}^{d}\mathbf{u}_i^j \log f_d^{\text{BinXent}}(z_i^j)$$

In this case, the predictions for each sample should be in $\mathbb{R}^{d\times T}$. As a simple implementation, we add the 1x1 convolutional layer at the end of $f_d^{\text{BinXent}}(\cdot) : Z \to \hat{U}$ where $U \in \mathbb{R}^{N\times d\times T}$ represents the one-hot encoded binned dataset.

We outline the benefits of utilizing the binning task in SSL as follows. Empirical evidence on how each item is advantageous for tabular data problems will be provided in subsequent sections.

- *Capturing the irregular function*: We explicitly make deep networks learn the function that maps from continuous inputs to discrete targets during SSL. It effectively provides beneficial inductive bias for tabular learning and mitigates the discrepancy between heterogeneous features. (Section 5.2, 6.3, 6.4)
- *Compatibility with other modifications*: The binning task is agnostic to modifications such as changes in encoder architecture, input transformation functions, and additional objectives. Thus, it can be utilized independently or in conjunction with other options. (Section 5.1, 5.2)
- *Standardizing all features into equal sets*[1]: After binning, all features lie on the uniform distribution with identical elements. Unlike the conventional normalization schemes, it largely simplifies the dataset to include only T distinct values, and this ensures all features become equal sets, thereby preventing any uninformative features from dominating during training. (Section 6.1)
- *Grouping similar values in each feature*: Binning clusters the nearby values in each feature and eliminates the other information except the bin index. Deep networks can identify nearby samples in a distribution as similar, independent of their magnitude. (Section 6.1)
- *Ordering in BinRecon loss*: BinRecon loss utilizes the grouped rank information only while eliminating the raw value information. This ensures that the encoder network learns the ordering information, regardless of the magnitude of the values. (Section 6.1)

Overall, we implement SSL as follows. First, tabular inputs undergo a transformation that retains their semantic information. Then, the encoder network f_e takes the transformed input $\tilde{x}$ and produces the representation z, and the decoder network f_d models the representation z to the target $\hat{y}_{\text{SSL}}$ depending on the choice of pretext task. In this study, we consider four types of pretext tasks and the corresponding losses are ValueRecon, MaskXent, BinRecon, and BinXent. Once SSL is finished, the learned representations z are evaluated based on linear probing.

5. Experiments

In this section, we evaluate the effectiveness of binning as a pretext task across 25 public tabular datasets encompassing a range of data sizes and task types. Dataset details are provided in Supplementary A. For all datasets, we apply standardization for numerical features and labels for evaluating the regression tasks.

For the encoder network f_e, we experiment three types of deep networks: (1) MLPs, representing the simplest form of deep architecture; (2) FT-Transformer (Gorishniy et al., 2021), a simple adaptation of the Transformer architecture for the tabular domain; and (3) T2G-Former (Yan et al., 2023), the state-of-the-art deep architecture for tabular data problems. Note that a larger or more complex network does not guarantee better performance in tabular datasets (Gorishniy et al., 2021; Rubachev et al., 2022; Grinsztajn et al., 2022; Gorishniy et al., 2022; McElfresh et al., 2023). To determine the depth and width of f_e in the case of MLP, we identify the optimal configuration based on validation performance in the supervised setup, i.e., only the encoder with a linear head is trained with the supervised loss, ensuring the unsupervised nature of our framework. In the case of FT-Transformer and T2G-Former, we use the default setup from the original paper. For the decoder f_d, we always employ the MLP architecture as the same as the MLP-type encoder network. Consequently, all cases for each dataset have been trained on the same architecture and optimization setups. A detailed description is provided in Supplementary B. For a given network and dataset, we also investigate the masking probability $p_m \in \{0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9\}$ and the number of bins $T \in \{2, 5, 10, 20, 50, 100\}$. Then, we found the optimal configuration based on validation performance on each downstream task. After SSL, we evaluate the representations based on linear probing 10 times with different random seeds, and an average is reported. We evaluate the representation quality based on accuracy for classification tasks and RMSE for regression tasks. The full results with standard deviation are also available in Supplementary C. All experiments are conducted on a single NVIDIA GeForce RTX 3090. The codes are available in `https://github.com/kyungeun-lee/tabularbinning`.

[1]Detailed description is available in Supplementary E.1.

Table 1: Linear evaluation results for various SSL methods when the encoder network is fixed as MLP. For each method, we also determine the performance rankings for each dataset, and the average ranks are also provided in the last column. Best cases for each dataset are marked in **bold**.

(a) Binary classification (Metric: Accuracy)

Masking	Replacing value	SSL Objective(s)	CH	HI	AD	BM	PH	OS	CS	PO	Average Rank
FALSE	-	ValueRecon	0.810	0.651	0.837	0.899	0.728	0.883	0.709	0.851	7.571
TRUE	Const.	MaskXent	0.807	0.672	0.836	0.899	0.715	0.893	0.708	0.845	7.286
TRUE	Const.	ValueRecon	0.810	0.653	0.839	0.900	0.734	0.884	0.718	0.849	6.429
TRUE	Const.	MaskXent+ValueRecon	0.817	0.669	0.835	0.900	0.724	0.877	0.706	0.837	7.714
TRUE	Random	MaskXent	0.814	0.681	0.843	**0.901**	0.710	0.883	0.706	0.853	5.429
TRUE	Random	ValueRecon	0.811	0.661	0.838	0.898	0.736	0.885	0.714	0.842	7.143
TRUE	Random	MaskXent+ValueRecon	0.804	0.647	0.826	0.899	0.715	0.879	0.713	0.861	8.571
FALSE	-	BinXent	0.817	0.683	0.845	**0.901**	0.732	0.886	**0.738**	0.851	3.571
FALSE	-	BinRecon	**0.823**	**0.687**	0.840	0.900	**0.737**	0.889	0.724	**0.865**	**2.286**
TRUE	Const.	BinRecon	0.820	0.672	0.843	0.899	0.730	**0.896**	0.718	0.858	3.714
TRUE	Random	BinRecon	0.819	0.682	**0.846**	0.898	0.735	0.894	0.718	0.858	3.571

(b) Multiclass classification (Metric: Accuracy)

Masking	Replacing value	SSL Objective(s)	CO	OT	GE	VO	WQ	AL	HE	MNIST	p-MNIST	Average Rank
FALSE	-	ValueRecon	0.769	0.776	0.527	0.619	0.568	0.931	0.353	0.965	0.928	6.333
TRUE	Const.	MaskXent	0.784	0.777	0.518	0.545	0.547	0.909	0.341	0.793	0.554	9.333
TRUE	Const.	ValueRecon	0.783	0.791	0.557	0.622	0.586	0.931	0.354	0.966	0.925	4.111
TRUE	Const.	MaskXent+ValueRecon	0.750	0.774	0.519	0.610	0.571	0.931	0.360	0.941	0.907	7.444
TRUE	Random	MaskXent	0.763	0.791	0.555	0.549	0.544	0.925	0.336	0.945	0.817	8.000
TRUE	Random	ValueRecon	0.761	0.782	0.538	0.625	0.573	0.930	0.357	0.956	0.934	5.556
TRUE	Random	MaskXent+ValueRecon	0.769	0.779	0.521	0.564	0.519	0.925	0.353	0.945	0.906	8.333
FALSE	-	BinXent	0.742	0.781	0.517	0.600	0.565	0.903	0.354	0.956	0.908	8.333
FALSE	-	BinRecon	0.784	0.783	0.544	0.625	**0.592**	0.935	0.357	0.964	0.950	3.556
TRUE	Const.	BinRecon	0.812	0.792	0.559	0.647	0.581	0.943	0.359	0.974	0.964	2.222
TRUE	Random	BinRecon	**0.814**	**0.794**	**0.580**	**0.655**	0.574	**0.949**	**0.365**	**0.981**	**0.971**	**1.333**

(c) Regression (Metric: RMSE)

Masking	Replacing value	SSL Objective(s)	CA	HO	FI	MI	KI	CPU	DIA	EL	Average Rank
FALSE	-	ValueRecon	0.749	4.241	13900.720	0.784	0.163	3.876	1016.641	0.399	8.625
TRUE	Const.	MaskXent	0.709	4.548	13473.750	0.788	0.185	4.475	1259.744	0.396	8.875
TRUE	Const.	ValueRecon	0.693	4.086	13518.683	0.778	0.160	3.728	952.444	0.394	5.000
TRUE	Const.	MaskXent+ValueRecon	0.700	4.157	13915.875	0.775	0.174	5.644	2797.034	0.398	8.750
TRUE	Random	MaskXent	0.677	4.297	13826.641	0.782	0.176	3.951	1358.135	0.388	7.875
TRUE	Random	ValueRecon	0.713	4.127	13668.988	0.777	0.162	3.760	986.306	0.396	6.500
TRUE	Random	MaskXent+ValueRecon	0.701	4.136	14107.645	0.780	0.166	4.506	1917.875	0.397	8.750
FALSE	-	BinXent	0.690	4.116	**13038.762**	0.776	0.170	3.717	1207.923	0.383	4.875
FALSE	-	BinRecon	0.622	3.766	13453.309	**0.767**	**0.158**	3.208	897.645	0.370	2.250
TRUE	Const.	BinRecon	0.634	3.765	13208.133	0.773	**0.158**	**3.156**	957.801	0.371	2.375
TRUE	Random	BinRecon	**0.619**	**3.703**	13075.474	0.773	0.160	3.183	**870.283**	**0.368**	**1.625**

5.1. Comparison with the unsupervised methods: Linear evaluation results

We first compare a series of SSL methods utilizing the same MLP encoders for each dataset. To identify the compatibility of the binning task with other transformation functions, we include the cases optimizing BinRecon loss with masking. Finally, we experiment with four cases to validate our methodology; optimizing BinXent, treating bins as nominal classes; optimizing BinRecon, treating bins as ordinal values without any augmentation; optimizing BinRecon with masking as constant values; and optimizing BinRecon with masking as random values. In Table 1, four rows at the bottom correspond to our methods.

Binary classification: First, we compare the performance of eight datasets whose downstream task is binary classification in Table 1a. Interestingly, we found a consistent improvement when we changed the target for reconstruction loss from the raw values (ValueRecon) to bin indices (BinRecon) while other training details were fixed. These results indicate that learning irregular functions (from continuous to discrete) is more beneficial than learning smooth functions (from continuous to continuous)

Table 2: Comparison with the tree-based and deep learning methods including state-of-the-art models. For baselines, we directly reference the performance values from the papers to minimize ambiguity in selecting the hyperparameters. When the performance is not available, we leave them blank(-). For each dataset, the best cases among deep learning methods are marked in **bold**, and the second best results are <u>underlined</u>. SSL+Fine-tuning methods refer to the fine-tuning results of the baseline SSL methods investigated in Section 5.1. For SSL+Fine-tuning methods corresponding to the four rows at the bottom, we provide the best results among the combinations of various input transformations (None, Masking as constant, Masking as random) and encoder networks (MLP, FT-Transformer, T2G-Former). Training details and full results are provided in Supplementary C.

Training network and method	Binary classification				Multiclass classification						Regression		
	HI ↑	PH ↑	OS ↑	PO ↑	CO ↑	GE ↑	VO ↑	AL ↑	HE ↑	MNIST ↑	CA ↓	HO ↓	FI ↓
Tree-based machine learning algorithms													
XGBoost	0.726	0.721	0.840	0.711	0.969	0.683	0.699	0.924	0.348	0.977	0.434	3.152	10372.778
CatBoost	0.727	0.728	0.833	0.897	0.967	0.692	0.711	0.948	0.386	0.979	0.430	3.093	10636.322
Deep learning methods													
MLP	0.714	0.724	<u>0.896</u>	<u>0.901</u>	0.968	0.659	0.692	0.960	0.378	0.983	0.513	3.146	<u>10086.080</u>
ResNet	0.688	0.728	0.885	0.795	0.729	0.484	0.550	0.220	0.229	0.826	0.706	4.004	10226.508
TabNet (Arik & Pfister, 2021; Gorishniy et al., 2021)	0.719	-	-	-	0.957	0.587	0.568	0.954	0.378	0.968	0.510	-	-
NODE (Popov et al., 2019; Gorishniy et al., 2021)	0.726	-	-	-	0.958	-	-	0.918	0.359	-	<u>0.464</u>	-	-
DCN V2 (Wang et al., 2021; Gorishniy et al., 2021)	0.723	-	-	-	0.965	-	-	0.955	0.385	-	0.484	-	-
SCARF (Bahri et al., 2021)	0.585	0.710	0.878	0.838	0.654	0.325	0.289	0.731	0.050	0.801	1.084	5.595	13632.255
SAINT (Somepalli et al., 2021)	0.713	0.728	0.886	0.877	0.943	0.691	0.713	0.932	0.378	0.981	0.581	6.186	19366.582
FT-Transformer (Gorishniy et al., 2021)	0.729	0.724	0.882	0.890	0.970	0.664	0.705	0.960	**0.391**	0.966	0.487	3.319	10206.127
PLR (MLP-Ensemble) (Gorishniy et al., 2022)	<u>0.734</u>	-	-	-	0.970	0.674	-	-	-	-	0.467	3.050	-
PLR (FT-T-Ensemble) (Gorishniy et al., 2022)	<u>0.734</u>	-	-	-	**0.972**	0.646	-	-	-	-	<u>0.464</u>	3.162	-
T2G-Former (Yan et al., 2023)	<u>0.734</u>	0.746	0.884	0.881	0.968	0.656	0.717	<u>0.964</u>	**0.391**	<u>0.985</u>	**0.455**	3.138	10750.850
SSL(MaskXent)+Fine-tuning	0.725	<u>0.751</u>	0.892	0.897	0.970	<u>0.698</u>	0.717	0.963	0.383	<u>0.985</u>	0.479	<u>3.086</u>	10204.559
SSL(ValueRecon)+Fine-tuning	0.719	0.731	0.894	0.899	0.969	0.690	0.712	0.963	0.381	0.984	0.478	3.119	10333.400
SSL(MaskXent+ValueRecon)+Fine-tuning	0.727	0.737	0.894	0.896	0.968	0.658	0.709	0.959	0.382	0.984	0.475	3.257	10708.780
Ours – SSL(BinRecon)+Fine-tuning	**0.737**	**0.764**	**0.897**	**0.904**	<u>0.971</u>	**0.720**	**0.728**	**0.966**	<u>0.388</u>	**0.986**	<u>0.464</u>	**2.989**	**9757.950**

in tabular representation learning.

Multiclass classification: Next, we investigate nine datasets whose downstream task is multiclass classification in Table 1b. Unlike the binary classification tasks, we observe that optimizing BinRecon loss with masking consistently leads to additional improvements compared to the cases without masking, and optimizing BinXent does not work well. These results indicate that the order information is important for multiclass classification and BinRecon can effectively manipulate them. Further discussion will be provided in Section 6.

Regression: Finally, we test eight datasets whose downstream task is the regression in Table 1c. Since the evaluation metric is RMSE, lower values correspond to better-performing cases. Compared to other downstream tasks, regression tasks exhibit the most significant improvements with the binning pretext task. For instance, when comparing our method with the best baselines, we observed improvements of 10.27% for HO dataset, 8.63% for DIA dataset, and 8.57% for CA dataset.

5.2. Comparison with the supervised methods: Fine-tuning results

We observed that the binning consistently improves the unsupervised learning performance across the various tabular datasets and the downstream tasks. In this section, we compare our method against the supervised methods that utilize label information throughout the training. Our supervised baselines include tree-based algorithms, such as XGBoost (Chen & Guestrin, 2016) and CatBoost (Prokhorenkova et al., 2018), recent deep learning methods and fine-tuning results from SSL methods discussed in Section 5.1. Since supervised baselines often require extensive hyperparameter tuning, we directly reference the reported performances in the papers. When the performance has not been reported in the paper, we train with the default setup as depicted in the paper or leave them blank. For our methods, we first train encoder networks using the default setup with BinRecon loss in an unsupervised manner. Then, we conduct fine-tuning on the pre-trained encoders. The training details are provided in Supplementary C.

The results are summarized in Table 2 and Table 7, 10 in the supplementary material. Surprisingly, our method consistently outperforms both the tree-based and deep learning methods, even though it relies solely on changing the objective function to discretized bins during pre-training. On average, we outperform XGBoost by 5.55% (max. 27.14%), CatBoost by 2.18% (max. 8.26%), the state-of-the-art deep learning method (T2G-Former) by 2.30% (max. 9.76%), and the fine-tuning results of other SSL methods by 1.55% (max. 4.38%).

The superior performance of our method is primarily attributed to its unsupervised pretraining phase, a strategy particularly effective in deep learning and absent in tree-based algorithms. The key to its success lies in manipulating an appropriate inductive bias during pretraining. For our method, the binning objective effectively leverages the irregularities and mitigates the heterogeneity between the features as described in Section 4. Thanks to the successful implementation of this pretraining strategy, our approach achieves superior performance across a wide array of datasets.

6. Discussion

6.1. Ablation study on the individual factor for binning

In this section, we scrutinize the individual contributions of the components of binning, detailed in Section 4. Specifically, we examine the roles of discerning the order of samples within each feature, standardizing all features into equal sets, and grouping similar values. BinRecon encapsulates all three elements while ValueRecon disregards them completely. To dissect the influence of each factor, we systematically eliminate them one by one from the BinRecon loss as follows.

- Ordering: We shuffle the bin indices with different random seeds for each feature.
- Standardizing into equal sets: We replace the bin indices with the averages for each bin. Then, each feature includes different elements in different ranges.
- Grouping: We set $T^j = |\mathcal{D}^j_{\text{train}}|$ for every feature. In this case, each unique value corresponds to an individual bin, and only the order information remains.

As shown in Table 11 in supplementary material, we found that eliminating the standardizing factor shows the largest performance degradation, averaging a 6.85% decrease in 15 datasets among 25. This decline is much steeper than the effect of eliminating all three factors. From these observations, we infer that the standardizing factor which makes all features lie on the uniform distribution with identical elements is most critical for the successful implementation of binning. We provide a more detailed discussion related to the impact of the standardizing factor in Supplementary E.1.

6.2. Dependency between the number of bins and downstream task performance

In this section, we investigate the relationship between the number of bins and downstream task performance for BinXent and BinRecon without input transformation. To analyze results across datasets with varying ranges and different evaluation metrics, we assess normalized performance based on the best and worst performances among the models that differ by the number of bins, keeping the loss function consistent. As shown in Figure 6 in supplementary material, there is no clear relationship between the number of bins and normalized performance (Pearson correlation $\rho^2 = 0.01$, Kendall rank correlation $\tau = 0.16$ for BinXent, $\rho^2 = 0.04$, $\tau = 0.27$ for BinRecon), except that the number of bins should be not too small, but larger is not always better. This result is not surprising, as utilizing too few bins can eliminate necessary information while utilizing too many bins can diminish the benefits of binning.

6.3. Bin information is not usable unless it is provided as a pretext task

So far, we found that bin information is critical for achieving superior representations across various tabular data problems. However, even if we do not employ bin information as an explicit pretext task, it remains accessible from the raw values. In this section, we evaluate how accurately the learned representations can predict bin indices when we optimize ValueRecon or MaskXent during SSL. To gauge this, we measure the relative error increase against the results of BinRecon case. As shown in Table 12 in the supplementary material, the prediction error is steeply increased at an average of 66.3% when bin information is not provided. This underscores that while bin information can be derived from the data, its utility is markedly compromised unless it is adopted as a pretext task.

6.4. Visualization analysis

To demonstrate the superior capability of the binning task in effectively capturing irregular functions mapping continuous inputs to discretized bin indices, compared to other methods, we present a visualization analysis of representation vectors after SSL in Figure 4. Due to the high-dimensional nature of the representation vectors, we implement PCA for better interpretability. In the visualization, the bin indices are represented as different colors. A distinct pattern emerges from this analysis: the representation vectors are specifically grouped according to their bin indices in the case of BinRecon. This pronounced clustering is not evident when other pretext tasks are employed. These findings highlight the effectiveness of binning as a pretext task. It demonstrates the unique capacity of this approach to enable the encoder to accurately capture the irregular function, distinguishing it from other methods.

6.5. Optimizing multiple loss functions during SSL

In Section 4, we introduce the potential for integrating multiple loss functions in SSL by employing various decoders

Table 3: Comparison of fine-tuning performance for tabular SSL when applying binning as data augmentation (Randomized Quantization, RQ) versus using binning to define output labels (Ours).

Training method	HI ↑	PH ↑	OS ↑	PO ↑	CO ↑	GE ↑	VO ↑	AL ↑	HE ↑	MNIST ↑	CA ↓	HO ↓	FI ↓
RQ (Wu et al., 2023)	0.717	0.736	0.896	0.886	0.969	0.690	0.719	0.959	0.379	0.984	0.475	3.159	10398.616
Ours	**0.737**	**0.764**	**0.897**	**0.904**	**0.971**	**0.720**	**0.728**	**0.966**	**0.388**	**0.986**	**0.464**	**2.989**	**9757.950**

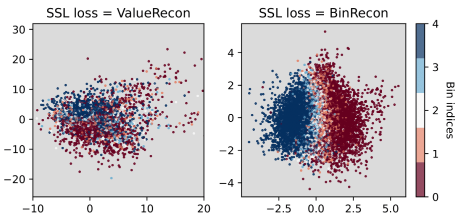


Figure 4: Visualization analysis using HO dataset. For better interpretability, we implement PCA for the learned representation vectors based on the different objective functions, plotting the first two principal components. Colors denote the bin indices of each sample.

that share a common input representation, z. This strategy underlines the flexibility of our approach, though it was not the primary focus of our current study.

Our preliminary investigations into the GE dataset revealed performance gains from this multi-decoder strategy. Specifically, under conditions of 2 bins and a 0.1 random masking probability, training with a single MaskXent or ValueRecon loss yielded linear probing performance of 0.509 and 0.553, individually. On the other hand, training with a single BinRecon loss yielded a linear probing performance of 0.560. Further, introducing an additional decoder to simultaneously optimize both BinRecon and MaskXent losses (with equal weights) or both BinRecon and ValueRecon losses (with equal weights) improved linear probing performance to 0.577. These observations imply that incorporating binning loss with other SSL objectives such as MaskXent could improve tabular representation learning.

6.6. Binning as an input transformation

Wu et al. (2023) introduced Randomized Quantization (RQ) as a data-agnostic augmentation strategy for contrastive representation learning, which applies binning directly to the *input samples*. In contrast, our method primarily utilizes classical binning on *output labels* within an auto-encoding-based self-supervised learning framework.

To determine the more effective application of binning for tabular representation learning, we implemented the RQ augmentation using the official codes, following the same experimental setups as our baseline methods, as detailed in both the manuscript and supplementary materials. For the hyperparameters of the RQ augmentation, we selected the same range as our method: $\{2, 5, 10, 20, 50, 100\}$.

As summarized in Table 3, our approach consistently outperformed the RQ method. According to Wu et al. (2023), employing binning as an augmentation strategy leads to inevitable information loss within the input samples. While in domains with inherent redundancy (*e.g.*, images), some information loss can be mitigated through other channels or local patterns, the tabular domain typically lacks such compensatory mechanisms. For example, in a medical dataset predicting diabetes, reducing detail in a critical feature like blood sugar levels cannot be compensated by other variables due to the independence of features within tabular data. Consequently, we anticipate that employing binning as an input transformation in the tabular domain may not be effective, as it would lead to the systematic removal of vital information.

7. Conclusion

In this work, we suggest a novel pretext task based on binning which can manipulate the unique properties of tabular datasets. The binning task can effectively address the challenges in tabular SSL, including mitigating the feature heterogeneity and learning the irregularities. Importantly, our method focuses exclusively on modifying the objective function and is independent of specific architectures or augmentation methods. Based on the extensive experiments, we found that the binning task not only improves the unsupervised representation learning but also is a powerful pretraining strategy to achieve consistently superior performance against the tree-based and other deep learning methods. In this study, we have uncovered the potential of leveraging the inherent properties of tabular data as pretext tasks for SSL. However, many unique characteristics remain unexplored, such as hierarchical relationships between features. We hope our work inspires further investigations into tabular-data-specific SSL in the future.

Impact Statement

This paper contributes to advancing the field of Machine Learning, particularly focusing on tabular data, a domain prevalent in numerous real-world applications. Our work holds the potential to significantly enhance data analysis and predictive modeling across various sectors, including healthcare, finance, and social sciences, where tabular data is extensively used. While we believe our method can lead to positive societal impacts, such as improved decision-making and more efficient data processing, we also acknowledge the importance of responsible use. It is crucial to ensure that the deployment of these advanced machine learning techniques is carried out with ethical considerations and a commitment to mitigating biases. We hope this research inspires further innovations in machine learning while prompting continuous discussion on its ethical and societal implications.

References

Arik, S. Ö. and Pfister, T. Tabnet: Attentive interpretable tabular learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, pp. 6679–6687, 2021.

Badirli, S., Liu, X., Xing, Z., Bhowmik, A., Doan, K., and Keerthi, S. S. Gradient boosting neural networks: Grownet. *arXiv preprint arXiv:2002.07971*, 2020.

Bahri, D., Jiang, H., Tay, Y., and Metzler, D. Scarf: Self-supervised contrastive learning using random feature corruption. *arXiv preprint arXiv:2106.15147*, 2021.

Baldi, P., Sadowski, P., and Whiteson, D. Searching for exotic particles in high-energy physics with deep learning. *Nature communications*, 5(1):4308, 2014.

Beyazit, E., Kozaczuk, J., Li, B., Wallace, V., and Fadlallah, B. H. An inductive bias for tabular deep learning. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

Blackard, J. A. and Dean, D. J. Comparative accuracies of artificial neural networks and discriminant analysis in predicting forest cover types from cartographic variables. *Computers and electronics in agriculture*, 24(3): 131–151, 1999.

Borisov, V., Leemann, T., Seßler, K., Haug, J., Pawelczyk, M., and Kasneci, G. Deep neural networks and tabular data: A survey. *IEEE Transactions on Neural Networks and Learning Systems*, 2022.

Chen, J., Liao, K., Wan, Y., Chen, D. Z., and Wu, J. Danets: Deep abstract networks for tabular data classification and regression. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pp. 3930–3938, 2022.

Chen, K.-Y., Chiang, P.-H., Chou, H.-R., Chen, T.-W., and Chang, T.-H. Trompt: Towards a better deep neural network for tabular data. *arXiv preprint arXiv:2305.18446*, 2023a.

Chen, P., Sarkar, S., Lausen, L., Srinivasan, B., Zha, S., Huang, R., and Karypis, G. Hytrel: Hypergraph-enhanced tabular data representation learning. *arXiv preprint arXiv:2307.08623*, 2023b.

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

Cherepanova, V., Levin, R., Somepalli, G., Geiping, J., Bruss, C. B., Wilson, A. G., Goldstein, T., and Goldblum, M. A performance-driven benchmark for feature selection in tabular deep learning. *Advances in Neural Information Processing Systems*, 36, 2024.

Cortez, P., Cerdeira, A., Almeida, F., Matos, T., and Reis, J. Modeling wine preferences by data mining from physicochemical properties. *Decision support systems*, 47(4): 547–553, 2009.

Dougherty, J., Kohavi, R., and Sahami, M. Supervised and unsupervised discretization of continuous features. In *Machine learning proceedings 1995*, pp. 194–202. Elsevier, 1995.

Geusebroek, J.-M., Burghouts, G. J., and Smeulders, A. W. The amsterdam library of object images. *International Journal of Computer Vision*, 61:103–112, 2005.

Gorishniy, Y., Rubachev, I., Khrulkov, V., and Babenko, A. Revisiting deep learning models for tabular data. *Advances in Neural Information Processing Systems*, 34: 18932–18943, 2021.

Gorishniy, Y., Rubachev, I., and Babenko, A. On embeddings for numerical features in tabular deep learning. *Advances in Neural Information Processing Systems*, 35: 24991–25004, 2022.

Goyal, P., Dollár, P., Girshick, R., Noordhuis, P., Wesolowski, L., Kyrola, A., Tulloch, A., Jia, Y., and He, K. Accurate, large minibatch sgd: Training imagenet in 1 hour. *arXiv preprint arXiv:1706.02677*, 2017.

Grinsztajn, L., Oyallon, E., and Varoquaux, G. Why do tree-based models still outperform deep learning on typical tabular data? *Advances in Neural Information Processing Systems*, 35:507–520, 2022.

Guyon, I., Sun-Hosoya, L., Boullé, M., Escalante, H. J., Escalera, S., Liu, Z., Jajetic, D., Ray, B., Saeed, M., Sebag, M., et al. Analysis of the automl challenge series. *Automated Machine Learning*, 177, 2019.

Han, J., Pei, J., and Tong, H. *Data mining: concepts and techniques*. Morgan kaufmann, 2022.

Hollmann, N., Müller, S., Eggensperger, K., and Hutter, F. Tabpfn: A transformer that solves small tabular classification problems in a second. *arXiv preprint arXiv:2207.01848*, 2022.

Huang, X., Khetan, A., Cvitkovic, M., and Karnin, Z. Tabtransformer: Tabular data modeling using contextual embeddings. *arXiv preprint arXiv:2012.06678*, 2020.

Kohavi, R. et al. Scaling up the accuracy of naive-bayes classifiers: A decision-tree hybrid. In *Kdd*, volume 96, pp. 202–207, 1996.

Kotelnikov, A., Baranchuk, D., Rubachev, I., and Babenko, A. Tabddpm: Modelling tabular data with diffusion models. In *International Conference on Machine Learning*, pp. 17564–17579. PMLR, 2023.

Loshchilov, I. and Hutter, F. Sgdr: Stochastic gradient descent with warm restarts. *arXiv preprint arXiv:1608.03983*, 2016.

Loshchilov, I. and Hutter, F. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

Majmundar, K., Goyal, S., Netrapalli, P., and Jain, P. Met: Masked encoding for tabular data. *arXiv preprint arXiv:2206.08564*, 2022.

McElfresh, D., Khandagale, S., Valverde, J., Ramakrishnan, G., Goldblum, M., White, C., et al. When do neural nets outperform boosted trees on tabular data? *arXiv preprint arXiv:2305.02997*, 2023.

Moro, S., Laureano, R., and Cortez, P. Using data mining for bank direct marketing: An application of the crisp-dm methodology. 2011.

Müller, S., Hollmann, N., Arango, S. P., Grabocka, J., and Hutter, F. Transformers can do bayesian inference. *arXiv preprint arXiv:2112.10510*, 2021.

Pace, R. K. and Barry, R. Sparse spatial autoregressions. *Statistics & Probability Letters*, 33(3):291–297, 1997.

Popov, S., Morozov, S., and Babenko, A. Neural oblivious decision ensembles for deep learning on tabular data. *arXiv preprint arXiv:1909.06312*, 2019.

Prokhorenkova, L., Gusev, G., Vorobev, A., Dorogush, A. V., and Gulin, A. Catboost: unbiased boosting with categorical features. *Advances in neural information processing systems*, 31, 2018.

Qin, T. and Liu, T.-Y. Introducing letor 4.0 datasets. *arXiv preprint arXiv:1306.2597*, 2013.

Rubachev, I., Alekberov, A., Gorishniy, Y., and Babenko, A. Revisiting pretraining objectives for tabular deep learning. *arXiv preprint arXiv:2207.03208*, 2022.

Sakar, C. O., Polat, S. O., Katircioglu, M., and Kastro, Y. Real-time prediction of online shoppers' purchasing intention using multilayer perceptron and lstm recurrent neural networks. *Neural Computing and Applications*, 31:6893–6908, 2019.

Somepalli, G., Goldblum, M., Schwarzschild, A., Bruss, C. B., and Goldstein, T. Saint: Improved neural networks for tabular data via row attention and contrastive pre-training. *arXiv preprint arXiv:2106.01342*, 2021.

Stewart, L., Bach, F., Berthet, Q., and Vert, J.-P. Regression as classification: Influence of task formulation on neural network features. In *International Conference on Artificial Intelligence and Statistics*, pp. 11563–11582. PMLR, 2023.

Ucar, T., Hajiramezanali, E., and Edwards, L. Subtab: Subsetting features of tabular data for self-supervised representation learning. *Advances in Neural Information Processing Systems*, 34:18853–18865, 2021.

Vanschoren, J., Van Rijn, J. N., Bischl, B., and Torgo, L. Openml: networked science in machine learning. *ACM SIGKDD Explorations Newsletter*, 15(2):49–60, 2014.

Vincent, P., Larochelle, H., Bengio, Y., and Manzagol, P.-A. Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th international conference on Machine learning*, pp. 1096–1103, 2008.

Wang, R., Shivanna, R., Cheng, D., Jain, S., Lin, D., Hong, L., and Chi, E. Dcn v2: Improved deep & cross network and practical lessons for web-scale learning to rank systems. In *Proceedings of the web conference 2021*, pp. 1785–1797, 2021.

Wu, H., Lei, C., Sun, X., Wang, P.-S., Chen, Q., Cheng, K.-T., Lin, S., and Wu, Z. Randomized quantization: A generic augmentation for data agnostic self-supervised learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 16305–16316, 2023.

Yan, J., Chen, J., Wu, Y., Chen, D. Z., and Wu, J. T2g-former: Organizing tabular features into relation graphs promotes heterogeneous feature interaction. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pp. 10720–10728, 2023.

Yoon, J., Zhang, Y., Jordon, J., and van der Schaar, M. Vime: Extending the success of self-and semi-supervised learning to tabular domain. *Advances in Neural Information Processing Systems*, 33:11033–11043, 2020.

Zhu, B., Shi, X., Erickson, N., Li, M., Karypis, G., and Shoaran, M. Xtab: Cross-table pretraining for tabular transformers. *arXiv preprint arXiv:2305.06090*, 2023.

A. Dataset detail

In this study, we use 25 public datasets mostly from the OpenML (Vanschoren et al., 2014) library, including the frequently used datasets in previous studies (Yoon et al., 2020; Ucar et al., 2021; Gorishniy et al., 2021; 2022). Each dataset has exactly one train-validation-test split, so all algorithms use the same splits as the previous studies (Gorishniy et al., 2021; 2022; Rubachev et al., 2022). We summarize the main properties of datasets in Table 4. For each dataset, we use a predefined batch size depending on the number of training samples: 64 when the number of training samples is less than 1000, 128 when the number of training samples is larger than 1000 and less than 5000, 256 when the number of training samples is larger than 5000 and less than 10000, 512 when the number of training samples is larger than 10000 and less than 50000, and 1024 when the number of training samples is larger than 50000.

We regard the feature as categorical when the number of unique values in the training dataset is less than 20 (5 for AL, MNIST, p-MNIST, MI). The categorical variables are fed into the feature tokenizer for FT-Transformer while MLP has no additional operation for them. For MNIST and p-MNIST datasets, we ignore the features that have only one possible value throughout the training dataset.

In this study, we introduce a new p-MNIST dataset as a simple modification of well-known MNIST dataset. In constructing the p-MNIST dataset, we permute the pixel values across all samples based on a single, predefined order. Specifically, we first generate a permutation of the pixel indices ([0, 783]) using a fixed random seed. This pre-determined order is then consistently applied to the pixel values in all images within the whole dataset. The primary intention behind this methodology is to disrupt the inherent locality present in MNIST images (i.e. nearby columns are more related), thereby rendering the data more tabular-like, where spatial locality is less apparent or quantifiable (i.e. nearby columns are not necessarily more related).

Table 4: Dataset summary.

Abbr.	Name	# Train	# Validation	# Test	# Num	# Cat	Task type	Batch size
CH	Churn Modeling [2]	6400	1600	2000	4	6	Binclass	256
HI	Higgs Small (Baldi et al., 2014)	62751	15688	19610	24	4	Binclass	1024
AD	Adult (Kohavi et al., 1996)	26048	6513	16281	2	12	Binclass	512
BM	Bank Marketing (Moro et al., 2011)	28934	7234	9043	7	9	Binclass	512
PH	Philippine (Guyon et al., 2019)	3732	933	1167	308	0	Binclass	128
OS	Online Shoppers (Sakar et al., 2019)	7891	1973	2466	8	9	Binclass	256
CS	German Credit dataset [3]	640	160	200	20	0	Binclass	64
PO	Phoneme	3458	865	1081	5	0	Binclass	128
CO	Covertype (Blackard & Dean, 1999)	371847	92962	116203	44	7	Multiclass	1024
OT	Otto Group Products [4]	39601	9901	12376	80	13	Multiclass	512
GE	Gesture Phase	6318	1580	1975	32	0	Multiclass	256
VO	Volkert [5] (Guyon et al., 2019)	37318	9330	11662	147	33	Multiclass	512
WQ	Wine Quality (Cortez et al., 2009)	4157	1040	1300	11	0	Multiclass	128
AL	ALOI (Geusebroek et al., 2005)	69120	17280	21600	124	4	Multiclass	1024
HE	Helena (Guyon et al., 2019)	62752	15688	19610	27	0	Multiclass	512
MNIST	Handwritten Digit Images	50000	10000	10000	627	90	Multiclass	512
p-MNIST	Permuted MNIST	50000	10000	10000	627	90	Multiclass	512
CA	California Housing (Pace & Barry, 1997)	13209	3303	4128	8	0	Regression	512
HO	House 16H [6]	14581	3646	4557	16	0	Regression	512
FI	FIFA	12273	3069	3836	28	0	Regression	512
MI	MSLR-WEB10K(Fold 1) (Qin & Liu, 2013)	723412	235259	241521	131	5	Regression	1024
KI	Forward kinetics of an 8 link robot arm[6]	5242	1311	1639	8	0	Regression	256
CPU	Computer Activity Databases[6]	5242	1311	1639	8	0	Regression	256
DIA	Diamonds	34521	8631	10788	9	0	Regression	512
EL	Electricity [7]	24623	6156	7695	7	0	Regression	512

[2]https://www.kaggle.com/datasets/shrutimechlearn/churn-modelling

[3]https://archive.ics.uci.edu/dataset/144/statlog+german+credit+data

[4]https://www.kaggle.com/c/otto-group-product-classification-challenge/data

[5]https://automl.chalearn.org/data

[6]http://www.ncc.up.pt/~ltorgo/Regression/DataSets.html

[7]https://github.com/LeoGrin/tabular-benchmark

B. Implementation details

We use the optimization strategy for SSL as follows. We do not tune any hyperparameter and the same configuration is applied to all cases.

- Optimizer: AdamW (Loshchilov & Hutter, 2017)
- Learning rate: 1e-4
- Weight decay: 1e-5
- Epochs: 1000
- Learning rate scheduler: Cosine annealing scheduler (Loshchilov & Hutter, 2016; Goyal et al., 2017)

For the hyperparameters related to SSL, we tried $p_m \in \{0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9\}$ and $T \in \{2, 5, 10, 20, 50, 100\}$. When we combine the transformation function and binning methods, to reduce the hyperparameter space, we tried $p_m \in \{0.1, 0.2, 0.3\}$ and $T \in \{2, 10\}$ for MLPs, and $p_m \in \{0.1, 0.2\}$ and $T \in \{2, 10\}$ for FT-Transformers. After SSL, we evaluate the pre-trained representations with the linear head. The linear head is trained with different random seeds 10 times, and the average performance is reported.

For other state-of-the-art models, we directly reference the reported performance in the papers to reduce the ambiguity from the random seeds or the tuning details.

B.1. MLP

For MLPs, we set the architecture when the validation performance is best under the supervised setup with the encoder network f_e after the grid search on the depth (1, 2, 3, 4, 5) and the width (128, 256, 512, 1024). The representation size is determined as identical to the width of MLPs. The following decoder network f_d is defined as symmetric with f_e.

For supervised learning, we use the same configuration of SSL summarized above, except that the learning rate is 0.001 and the number of epochs is 100. We summarize the best setups for all datasets as follows.

Table 5: MLP architectures.

Depth	Datasets	Width	Datasets
1	CH, HI, AD, BM, OS, FI, CS	128	CH, HI, AD, BM, OS, FI, MI, CA
2	MI, CPU, HE, OT, AL	256	CS, HE, KI, PH, HO
3	CA, KI, MNIST, EL	512	CPU, WQ, p-MNIST, DIA
4	WQ, p-MNIST, PH, HO, CO, GE, VO, PO	1024	CO, GE, VO, PO, MNIST, EL, OT, AL
5	DIA		

B.2. FT-Transformer

We do not conduct any hyperparameter tuning for FT-Transformer, and we use the default setup defined in (Gorishniy et al., 2021) with the number of blocks as 3. For three large-scale datasets, such as MI, MNIST, and p-MNIST, we set the number of blocks as 1 because of the computational budget. For the representation size, we adopt the value found in MLP cases. For f_d, we use the MLP network whose architecture is the same as Table 5.

B.3. T2G-Former

We do not conduct any hyperparameter tuning for FT-Transformer, and we use the default setup defined in (Yan et al., 2023) with the number of layers as 3, the dimension of tokens as 192, the number of heads as 8, and the activation function as ReGLU. For the representation size, we adopt the value found in MLP cases. For f_d, we use the MLP network whose architecture is the same as Table 5.

B.4. Linear evaluation and Fine-tuning

For linear evaluation, we use the same optimization configuration for SSL except for the learning rate of 0.01 for 100 epochs. For fine-tuning, we use the same setups of the supervised cases for 50 or 100 epochs.

C. Full results

Here, we present the comprehensive results from our manuscript, accompanied by standard deviations derived from 10 repetitions of the experiment.

Table 6: Full results of Table 1. We repeat the evaluation 10 times and the average and the standard deviations are provided.

(a) Binary classification

Masking	Masking value	Objective(s)	CH	HI	AD	BM	PH	OS	CS	PO
FALSE	-	ValueRecon	0.810±0.001	0.651±0.000	0.837±0.000	0.899±0.000	0.728±0.001	0.883±0.000	0.709±0.003	0.851±0.000
FALSE	RQ	ValueRecon	0.816±0.000	0.654±0.000	0.842±0.000	0.898±0.000	0.727±0.001	0.882±0.001	0.725±0.002	0.842±0.000
TRUE	Const.	MaskXent	0.807±0.001	0.672±0.000	0.836±0.000	0.899±0.000	0.715±0.000	0.893±0.000	0.708±0.004	0.845±0.000
TRUE	Const.	ValueRecon	0.810±0.000	0.653±0.000	0.839±0.000	0.900±0.000	0.734±0.001	0.884±0.000	0.718±0.002	0.849±0.001
TRUE	Const.	MaskXent+ValueRecon	0.817±0.001	0.669±0.000	0.835±0.000	0.900±0.000	0.724±0.001	0.877±0.000	0.706±0.000	0.837±0.002
TRUE	Random	MaskXent	0.814±0.000	0.681±0.000	0.843±0.000	**0.901**±0.000	0.710±0.000	0.883±0.000	0.706±0.000	0.853±0.000
TRUE	Random	ValueRecon	0.811±0.000	0.661±0.000	0.838±0.000	0.898±0.000	0.736±0.001	0.885±0.000	0.714±0.003	0.842±0.000
TRUE	Random	MaskXent+ValueRecon	0.804±0.001	0.647±0.000	0.826±0.000	0.899±0.000	0.715±0.003	0.879±0.001	0.713±0.003	0.861±0.001
FALSE	-	BinXent	0.817±0.001	0.683±0.000	0.845±0.000	**0.901**±0.000	0.732±0.001	0.886±0.000	**0.738**±0.000	0.851±0.001
FALSE	-	BinRecon	**0.823**±0.000	**0.687**±0.000	0.840±0.000	0.900±0.000	**0.737**±0.000	0.889±0.000	0.724±0.005	**0.865**±0.000
TRUE	Const.	BinRecon	0.820±0.000	0.672±0.000	0.843±0.000	0.899±0.000	0.730±0.000	**0.896**±0.000	0.718±0.004	0.858±0.000
TRUE	Random	BinRecon	0.819±0.001	0.682±0.000	**0.846**±0.000	0.898±0.000	0.735±0.000	0.894±0.000	0.718±0.004	0.858±0.000

(b) Multiclass classification

Masking	Masking value	Objective(s)	CO	OT	GE	VO	WQ	AL	HE	MNIST	p-MNIST
FALSE	-	ValueRecon	0.769±0.000	0.776±0.000	0.527±0.001	0.619±0.000	0.568±0.001	0.931±0.000	0.353±0.000	0.965±0.000	0.928±0.000
FALSE	RQ	ValueRecon	0.775±0.000	0.771±0.000	0.531±0.001	0.620±0.000	0.559±0.000	0.930±0.000	0.350±0.000	0.948±0.000	0.706±0.000
TRUE	Const.	MaskXent	0.784±0.000	0.777±0.000	0.518±0.001	0.545±0.000	0.547±0.000	0.909±0.001	0.341±0.000	0.793±0.000	0.554±0.000
TRUE	Const.	ValueRecon	0.783±0.000	0.791±0.000	0.557±0.001	0.622±0.000	0.586±0.001	0.931±0.000	0.354±0.000	0.966±0.000	0.925±0.000
TRUE	Const.	MaskXent+ValueRecon	0.750±0.000	0.774±0.001	0.519±0.005	0.610±0.000	0.571±0.001	0.931±0.000	0.360±0.000	0.941±0.000	0.907±0.000
TRUE	Random	MaskXent	0.763±0.000	0.791±0.000	0.555±0.000	0.549±0.000	0.544±0.001	0.925±0.000	0.336±0.000	0.945±0.000	0.817±0.000
TRUE	Random	ValueRecon	0.761±0.000	0.782±0.000	0.538±0.001	0.625±0.000	0.573±0.001	0.930±0.000	0.357±0.000	0.956±0.000	0.934±0.000
TRUE	Random	MaskXent+ValueRecon	0.769±0.000	0.779±0.001	0.521±0.004	0.564±0.001	0.519±0.004	0.925±0.001	0.353±0.001	0.945±0.000	0.906±0.001
FALSE	-	BinXent	0.742±0.000	0.781±0.000	0.517±0.001	0.600±0.001	0.565±0.001	0.903±0.001	0.354±0.000	0.956±0.000	0.908±0.000
FALSE	-	BinRecon	0.784±0.000	0.783±0.000	0.544±0.001	0.625±0.000	**0.592**±0.001	0.935±0.000	0.357±0.000	0.964±0.000	0.950±0.000
TRUE	Const.	BinRecon	0.812±0.000	0.792±0.000	0.559±0.001	0.647±0.000	0.581±0.001	0.943±0.000	0.359±0.000	0.974±0.000	0.964±0.000
TRUE	Random	BinRecon	**0.814**±0.000	**0.794**±0.000	**0.580**±0.000	**0.655**±0.000	0.574±0.001	**0.949**±0.000	**0.365**±0.000	**0.981**±0.000	**0.971**±0.000

(c) Regression

Masking	Masking value	Objective(s)	CA	HO	FI	MI	KI	CPU	DIA	EL
FALSE	-	ValueRecon	0.749±0.000	4.241±0.001	13900.720± 0.816	0.784±0.000	0.163±0.000	3.876±0.002	1016.641±0.191	0.399±0.001
FALSE	RQ	ValueRecon	0.714±0.000	4.165±0.000	13684.367±0.778	0.784±0.000	0.162±0.000	3.751±0.002	1056.453±0.229	0.398±0.000
TRUE	Const.	MaskXent	0.709±0.000	4.548±0.000	13473.750± 1.371	0.788±0.000	0.185±0.000	4.475±0.033	1259.744±1.066	0.396±0.000
TRUE	Const.	ValueRecon	0.693±0.000	4.086±0.000	13518.683± 0.936	0.778±0.000	0.160±0.000	3.728±0.003	952.444±0.130	0.394±0.000
TRUE	Const.	MaskXent+ValueRecon	0.700±0.001	4.157±0.045	13915.875±18.078	0.775±0.000	0.174±0.001	5.644±0.078	2797.034±191.324	0.398±0.001
TRUE	Random	MaskXent	0.677±0.000	4.297±0.000	13826.641± 0.624	0.782±0.000	0.176±0.000	3.951±0.001	1358.135±0.191	0.388±0.000
TRUE	Random	ValueRecon	0.713±0.000	4.127±0.000	13668.988± 1.262	0.777±0.000	0.162±0.000	3.760±0.002	986.306±0.359	0.396±0.000
TRUE	Random	MaskXent+ValueRecon	0.701±0.003	4.136±0.028	14107.645±29.125	0.780±0.001	0.166±0.001	4.506±0.034	1917.875±123.359	0.397±0.008
FALSE	-	BinXent	0.690±0.000	4.116±0.001	**13038.762**± 1.618	0.776±0.000	0.170±0.000	3.717±0.006	1207.923±2.552	0.383±0.000
FALSE	-	BinRecon	0.622±0.000	3.766±0.000	13453.309± 0.832	**0.767**±0.000	**0.158**±0.000	3.208±0.001	897.645±0.182	0.370±0.000
TRUE	Const.	BinRecon	0.634±0.000	3.765±0.000	13208.133± 1.960	0.773±0.000	**0.158**±0.000	**3.156**±0.002	957.801±0.070	0.371±0.000
TRUE	Random	BinRecon	**0.619**±0.000	**3.703**±0.000	13075.474± 0.800	0.773±0.000	0.160±0.000	3.183±0.001	**870.283**±0.093	**0.368**±0.000

Table 7: Fine-tuning results with standard deviation based on 10 times repeated experiments. In this case, the models are pretrained in an unsupervised fashion (i.e. optimizing BinRecon loss) and fine-tuned in a supervised fashion. For each dataset and encoder, we experiment with various combinations of input transformation methods and the number of bins as explained in Supplementary B. Then, we repeated the best case determined based on the validation performance.

Encoder	HI ↑	PH ↑	OS ↑	PO ↑	CO ↑	GE ↑	VO ↑	AL ↑	HE ↑	MNIST ↑	CA ↓	HO ↓	FI ↓
MLP	0.717±0.001	0.738±0.009	**0.897±0.000**	0.893±0.004	0.969±0.000	0.673±0.004	**0.728±0.001**	0.963±0.001	**0.388±0.001**	**0.986±0.000**	0.502±0.002	**2.989±0.015**	9963.609± 23.173
FT-Transformer	0.703±0.004	0.742±0.011	0.882±0.004	**0.904±0.003**	**0.971±0.000**	0.698±0.006	0.720±0.003	0.961±0.001	0.374±0.002	0.978±0.001	0.475±0.003	3.173±0.024	**9757.950±210.751**
T2G-Former	**0.737±0.001**	**0.764±0.008**	0.892±0.003	0.895±0.005	0.967±0.001	**0.720±0.002**	0.725±0.001	**0.966±0.001**	0.378±0.002	0.985±0.000	**0.464±0.001**	3.144±0.041	10155.818±132.559

Table 8: Fine-tuning results using a fixed set of hyperparameters across all datasets (encoder network: T2G-Former, input transformation: random masking with a 0.2 ratio, and the number of bins=10). In this case, the models are pretrained in an unsupervised fashion (i.e. optimizing BinRecon loss) and fine-tuned in a supervised fashion. Even with these fixed parameters, our method maintains a competitive edge over baseline approaches, thereby affirming the intrinsic strength and adaptability of our approach. Specifically, under this setting, our method still showcases notable performance improvements across a range of datasets, reaffirming its effectiveness beyond the scope of extensive hyperparameter optimization.

Encoder	HI ↑	PH ↑	OS ↑	PO ↑	CO ↑	GE ↑	VO ↑	AL ↑	HE ↑	MNIST ↑	CA ↓	HO ↓	FI ↓	Average Rank
XGBoost	0.726	0.721	0.840	0.711	0.969	0.683	0.699	0.924	0.348	0.977	0.434	3.152	10372.778 3.462	
SSL+Finetuning(T2G-Former, Random masking(0.2), MaskXent)	0.714	0.733	0.872	0.895	0.925	0.708	0.705	0.960	0.365	0.983	0.551	3.174	10201.881	2.692
SSL+Finetuning(T2G-Former, Random masking(0.2), ValueRecon)	0.719	0.727	0.870	0.892	0.874	0.673	0.713	0.762	0.343	0.982	0.474	3.310	10434.967	4.000
SSL+Finetuning(T2G-Former, Random masking(0.2), MaskXent+ValueRecon)	0.721	0.757	0.873	0.891	0.839	0.657	0.699	0.958	0.351	0.983	0.478	3.101	10063.307	3.000
Ours(T2G-Former, Random masking(0.2), Bin=10, BinRecon)	0.734	0.773	0.880	0.895	0.965	0.699	0.728	0.963	0.380	0.983	0.469	3.193	10006.578	1.462

We also summarize the detailed training setups for the best cases in Table 7 as follows.

Table 9: Training setups for the best cases in Table 7.

Datasets	HI	PH	OS	PO	CO	GE	VO	AL	HE	MNIST	CA	HO	FI
Encoders	T2G-Former	T2G-Former	MLP	FT-Transformer	FT-Transformer	T2G-Former	MLP	T2G-Former	MLP	MLP	T2G-Former	MLP	FT-Transformer
Input transformation	Masking(Random)	Masking(Random)	None	Masking(Random)	Masking(Const.)	Masking(Random)	Masking(Const.)	Masking(Random)	Masking(Random)	Masking(Random)	Masking(Const.)	Masking(Random)	Masking(Const.)
Masking probability (p_m)	0.1	0.2	-	0.1	0.2	0.2	0.3	0.2	0.2	0.3	0.2	0.2	0.2
Number of bins	2	10	2	10	10	10	10	10	2	10	10	2	2
Fine-tuning epochs	50	50	50	50	100	100	100	100	100	100	50	100	100

Here are the results for the other list of datasets, not included in Table 2. Again, we found that the binning method consistently outperforms other methods.

Table 10: Fine-tuning results for the other list of datasets, not included in Table 2.

Training network and method	Binary classification				Multiclass classification			Regression				
	CH ↑	AD ↑	BM ↑	CS ↑	OT ↑	WQ ↑	p-MNIST ↑	MI ↓	KI ↓	CPU ↓	DIA ↓	EL ↓
Tree-based machine learning algorithms												
XGBoost	0.859	0.875	0.903	0.710	0.827	0.632	0.978	0.742	0.128	15.437	564.547	0.293
CatBoost	0.861	0.873	0.907	0.750	0.825	0.659	0.980	0.741	0.090	2.668	531.584	0.291
Deep learning methods												
MLP	0.838	0.851	0.902	0.666	0.810	0.629	0.980	0.753	0.072	2.764	563.123	0.354
ResNet	0.827	0.842	0.903	0.750	0.745	0.570	0.806	0.769	0.160	3.517	919.240	0.409
FT-Transformer (Gorishniy et al., 2021)	0.831	0.836	0.904	0.676	0.796	0.618	0.957	**0.746**	0.073	2.746	538.575	0.350
T2G-Former (Yan et al., 2023)	**0.863**	**0.860**	0.903	0.681	**0.819**	0.599	0.980	0.754	0.069	2.708	544.061	0.350
SSL(RQ)+Fine-tuning	0.843	0.852	0.900	0.695	0.812	0.627	0.978	0.757	0.073	2.709	530.547	0.346
SSL(MaskXent)+Fine-tuning	0.841	0.849	0.904	0.713	0.816	0.639	0.978	0.753	0.071	2.786	538.677	**0.338**
SSL(ValueRecon)+Fine-tuning	0.837	0.851	0.904	0.681	0.816	0.631	0.978	0.753	0.072	2.688	541.866	0.347
SSL(MaskXent+ValueRecon)+Fine-tuning	0.836	0.848	0.902	0.725	0.815	0.628	0.978	0.752	0.071	2.712	576.607	0.344
Ours – SSL(BinRecon)+Linear eval/Fine-tuning	0.843	0.857	**0.910**	**0.774**	0.817	**0.648**	**0.982**	0.750	**0.068**	**2.686**	**531.458**	0.339

D. Additional results for discussion

D.1. Comparing the binning method between the quantiles and the equal-width

We found that the grouping is critical for implementing the binning task successfully. Instead of quantile-based binning in our method, we also can manipulate equal-width binning. Here, we experiment with which method can be more beneficial for binning between the quantile and fixed size. We test the same candidates for the number of bins for equal-width binning, and we compare the test performance when the validation performance is the best with the quantile-based ones.

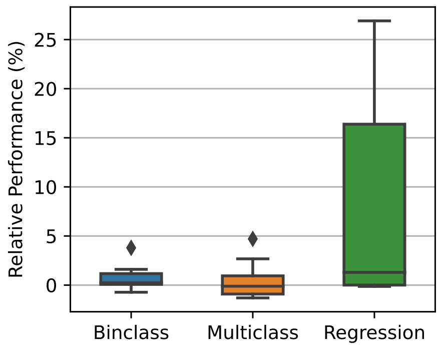


Figure 5: Relative performance when we change the binning method to the equal-width from the quantiles. When the values are positive, the quantile-based binning is better than the equal-width binning. When the values are negative, vice versa. In particular, for regression tasks, the quantile-based binning is much better than the equal-width binning.

The results are described in Figure 5. Among 25 datasets, equal-width binning showed better performance for three datasets (PH, HE, MNIST) to the extent of 0.6% at the maximum, and two binning methods showed comparable performance for two datasets (OT, AL). For the other 20 datasets, quantile-based binning showed better performance. In particular, for regression tasks, we found that the performance degrades 27% as the maximum when we change the binning method from quantile to fixed size. Finally, we conclude that quantile-based binning consistently results in good representations across various datasets.

D.2. Dependency between the number of bins and downstream task performance

We investigate the relationship between the number of bins and downstream task performance for BinXent and BinRecon without input transformation. Because the performance range is quite different between the datasets, we normalize the performance with the best and worst cases for each dataset. This approach allows us to normalize performance metrics across datasets with varying ranges and different evaluation metrics. Specifically, we assess the best and worst performance among six models that differ by the number of bins (2, 5, 10, 20, 50, 100), keeping the loss function (BinXent, BinRecon) consistent. Consequently, the normalized scale sets the best-performing case to 1 and the worst-performing case to 0.

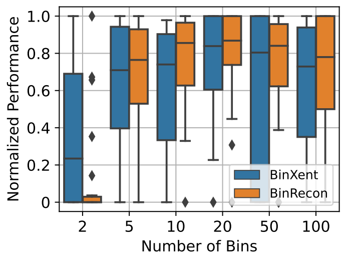


Figure 6: Empirical analysis on the dependency between the number of bins and the downstream task performance.

D.3. Results for Section 6

Table 11: Ablation test results on individual components of binning.

Ordering	Standardizing	Grouping	Improved	Deteriorated
Yes	Yes	Yes	- (*Baseline*)	- (*Baseline*)
No	Yes	Yes	1 (+4.70%)	12 (−4.05%)
Yes	No	Yes	1 (+5.21%)	15 (−6.85%)
Yes	No	No	3 (+1.95%)	12 (−5.83%)
No	No	No	-	18 (−6.02%)

Table 12: Binning regression task performance on various SSL methods. We provide the relative error with the baseline of the BinRecon case. For all cases, the error is increased by at least 38%. As a result, the binning indices are achievable from the raw inputs but not usable in the resulting representations when we do not explicitly provide as the pretext targets.

Masking	Masking value	Objective(s)	Relative error increase (%)
False	-	BinRecon	*(Baseline)* 0
False	-	ValueRecon	49.579
True	Const.	MaskXent	82.922
True	Const.	ValueRecon	38.444
True	Const.	MaskXent+ValueRecon	68.344
True	Random	MaskXent	111.708
True	Random	ValueRecon	38.135
True	Random	MaskXent+ValueRecon	60.016
		Average	66.285

D.4. Additional results for Section 6.6

Table 13: Comparison of fine-tuning performance for tabular SSL when applying binning as data augmentation (Randomized Quantization, RQ) versus using binning to define output labels (Ours).

Training method	HI ↑	PH ↑	OS ↑	PO ↑	CO ↑	GE ↑	VO ↑	AL ↑	HE ↑	MNIST ↑	CA ↓	HO ↓	FI ↓
RQ (Wu et al., 2023)	00.717±0.002	0.736±0.005	0.896±0.002	0.886±0.004	0.969±0.000	0.690±0.007	0.719±0.003	0.959±0.000	0.379±0.001	0.984±0.001	0.475±0.002	3.159±0.030	10398.616± 28.659
Ours	0.737±0.001	0.764±0.008	0.897±0.000	0.904±0.003	0.971±0.000	0.720±0.002	0.728±0.001	0.966±0.001	0.388±0.001	0.986±0.000	0.464±0.001	2.989±0.015	9757.950±210.751

E. Additional descriptions

E.1. Impact on uninformative features

Müller et al. (2021) and Stewart et al. (2023) demonstrated that incorporating discretization into the objective function of deep networks not only improves training efficiency but also proves to be a theoretically sound method for modeling any distribution. This underlines the significant potential of binning to enhance neural network performance. We propose that one key advantage of binning is its ability to simplify the dataset into T distinct values per feature, creating equal sets among all features. This property helps prevent uninformative features—those with low mutual information with the task label but potentially high entropy due to variance or a large number of unique values—from dominantly influencing the training process.

In scenarios where tabular SSL is applied using straightforward reconstruction loss, neural networks might inadvertently focus more on features characterized by their high variability(frequency) or unique value counts. This phenomenon, more pronounced in tabular data as noted in recent studies (Beyazit et al., 2023; Cherepanova et al., 2024), suggests that training could be skewed towards these high-frequency yet less informative features. By substituting output labels with bin indices during SSL, our method explicitly constrain all the features to regress on the SSL outputs with same frequency or same variability. Thus, it effectively circumvents that any specific feature from dominating during SSL, ensuring that such uninformative features do not overshadow the learning of meaningful representations.