# APAR: Modeling Irregular Target Functions in Tabular Regression via Arithmetic-Aware Pre-Training and Adaptive-Regularized Fine-Tuning

**Hong-Wei Wu, Wei-Yao Wang, Kuang-Da Wang, Wen-Chih Peng**

National Yang Ming Chiao Tung University, Hsinchu, Taiwan
johnnyhwu.cs11@nycu.edu.tw, sf1638.cs05@nctu.edu.tw, gdwang.cs10@nycu.edu.tw, wcpeng@cs.nycu.edu.tw

## Abstract

Tabular data are fundamental in common machine learning applications, ranging from finance to genomics and healthcare. This paper focuses on tabular regression tasks, a field where deep learning (DL) methods are not consistently superior to machine learning (ML) models due to the challenges posed by irregular target functions inherent in tabular data, causing sensitive label changes with minor variations from features. To address these issues, we propose a novel **A**rithmetic-Aware **P**re-training and **A**daptive-**R**egularized Fine-tuning framework (APAR), which enables the model to fit irregular target function in tabular data while reducing the negative impact of overfitting. In the pre-training phase, APAR introduces an arithmetic-aware pretext objective to capture intricate sample-wise relationships from the perspective of continuous labels. In the fine-tuning phase, a consistency-based adaptive regularization technique is proposed to self-learn appropriate data augmentation. Extensive experiments across 10 datasets demonstrated that APAR outperforms existing GBDT-, supervised NN-, and pretrain-finetune NN-based methods in RMSE (+9.43% $\sim$ 20.37%), and empirically validated the effects of pre-training tasks, including the study of arithmetic operations.

**Code** — https://github.com/johnnyhwu/APAR

## 1 Introduction

The tabular regression task, prevalent in sectors such as healthcare (Rao et al. 2023; Jain et al. 2024) and finance (Du, Wang, and Peng 2023; Deng et al. 2024), has commonly been addressed using Gradient Boosting Decision Tree (GBDT) models (e.g., CatBoost (Prokhorenkova et al. 2018)). Despite recent advancements in neural networks (NNs), they often fail to consistently outperform GBDT models in this domain (Wang et al. 2024). This is attributed to the understanding of important features from distinct characteristics of tabular data, such as feature heterogeneity and the presence of uninformative features, which make it challenging to identify important features. On the other hand, the irregular target functions prevent NNs from learning high-frequency components of heterogeneous tabular datasets, and negatively impact NN performance due to overfitting (Beyazit et al. 2023).

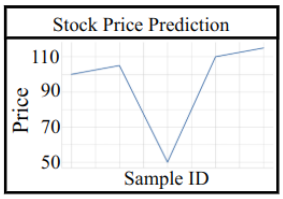


Prior research (Gorishniy et al. 2021; Yan et al. 2023; Chen et al. 2023a) has primarily focused on addressing feature heterogeneity and uninformative features; however, the issue of irregular target functions remains relatively unexplored, especially in the context of tabular regression tasks having continuous labels instead of explicit boundaries between labels. Irregular target functions play a critical role since minor deviations in input features lead to major changes in target values (Beyazit et al. 2023). For instance, as illustrated in Figure 1, in a medical scenario (e.g., health risk prediction), a slight variation in a patient's weight crossing a specific threshold can significantly alter the corresponding health status. In addition, the stock price is degraded significantly due to only the sentiment change (e.g., by wars). This phenomenon is less common in other data modalities; for example, a minor change in a single pixel in an image is unlikely to change its appearance. As NNs need to accurately model irregular target functions in tabular regression tasks while suffering from overfitting, it is crucial to emphasize the significance of advancing tabular regression methods capable of modeling sensitive changes between tabular features and labels.

Therefore, we focus on learning to fit irregular target functions for *tabular regression* tasks. Prior studies have mitigated this problem by addressing it from two perspectives: preventing overfitting on samples (Kossen et al. 2021; Ucar, Hajiramezanali, and Edwards 2021; Wang and Sun 2022) and features (Yoon et al. 2020; Arik and Pfister 2021; Somepalli et al. 2021). However, they are inferior in terms of utilizing label information due to the significant sparsity for regression labels (e.g., supervised contrastive learning for tabular classification tasks (Cui et al. 2024)) as well as in corrupting important features that are related to predictions (e.g., random feature masks (Chen et al. 2023b)).

To address the aforementioned challenges, we propose a novel **A**rithmetic-aware **P**re-training and **A**daptive-**R**egularized Fine-tuning framework (APAR) for tabular regression tasks, consisting of the pretrain-finetune strategy for modeling irregular target functions. Specifically, a Transformer-based (Vaswani et al. 2017) backbone is adopted with a tabular feature tokenizer to encode tabular heterogeneity. In the pre-training phase, an arithmetic-aware task is introduced to learn sample-wise relationships by predicting the combined answer of arithmetic operations on

| Volume | Sentiment | Interest | Indicator | Price |
|---|---|---|---|---|
| 1500 | Positive | 1.5 | 75 | 100 |
| 1600 | Positive | 1.6 | 76 | 105 |
| 1550 | Negative | 1.5 | 75 | 50 |
| 1650 | Positive | 1.7 | 77 | 110 |
| 1700 | Positive | 1.8 | 78 | 115 |




| Age | BP | LDL | Weight | Risk |
|---|---|---|---|---|
| 45 | 130 | 200 | 70 | 50 |
| 50 | 135 | 220 | 71 | 55 |
| 50 | 135 | 230 | 74 | 58 |
| 50 | 140 | 235 | 76 | 90 |
| 60 | 130 | 220 | 75 | 60 |

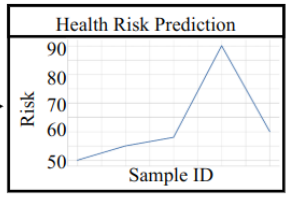



Figure 1: Illustrations of the impacts of irregular target functions commonly found in tabular regression tasks for finance (stock price prediction) and medical (health risk prediction) data. Small changes in features (marked in red) can lead to significant changes in the target variable.

continuous labels. In the fine-tuning phase, we propose an adaptive regularization technique to reinforce the model to self-learn proper data augmentation based on feature importance by training the model to understand similar representations between original and augmented data. We compared our APAR with GBDT and supervised as well as pretrain-finetune NNs on 10 datasets, which demonstrated a significant improvement of at least 9.43% in terms of the RMSE score compared with the state-of-the-art baseline.

In brief, our main contributions are described as follows:

- We present a principle recipe that enables NNs to effectively perform tabular regression tasks by addressing irregular target functions with the advantage of continuous labels and mitigating the negative impacts of overfitting on both samples and features.
- We introduce an arithmetic-aware pre-training method to learn the relationships between samples by solving arithmetic operations from continuous labels. In addition, our adaptive-regularized fine-tuning technique allows the model to perform self-guided data augmentation, offering effective regularization and generalization in downstream tasks.
- Extensive experiments across 10 datasets were systematically conducted to demonstrate an improvement from 9.43% to 20.37% compared with the GBDT-, supervised NN-, and pretrain-finetune NN-based methods.

## 2 Related Work

Recently, deep learning approaches for tabular data have demonstrated effective performance. For instance, Song et al. (2019) employed multi-head self-attention for intra-sample feature interactions. Huang et al. (2020) and Gorishniy et al. (2021) adapted the Transformer architecture to tabular data, targeting both categorical and numerical features. Nonetheless, GBDT-based approaches (Chen and Guestrin 2016; Ke et al. 2017; Prokhorenkova et al. 2018) are still competitive in tabular benchmarks due to the highly complex and heterogeneous characteristics of tabular data, causing NNs to overfit on irregular target functions.

To prevent NN from overfitting on samples, prior research has proposed to considering sample-wise relationships while learning individual representations. NPT (Kossen et al. 2021) and SAINT (Somepalli et al. 2021) utilize self-attention to explicitly reason about relationships between samples. Similarly, Ucar, Hajiramezanali, and Edwards (2021) employed self-supervised contrastive loss to ensure that the model outputs similar representations for different feature subsets of the same sample, and dissimilar representations for subsets of different samples. Although these approaches effectively capture relationships between samples, they require relatively large batch sizes (e.g., 4096 in NPT) to include more samples computed under self-attention, leading to high computational and memory consumption. Moreover, they do not leverage supervised labels to learn contextualized representations based on explicit information. To incorporate supervised lavels, Supervised Contrastive Loss (Khosla et al. 2020) and TransTab (Wang and Sun 2022) enable models to learn improved sample representations by making samples with the same label similar, and samples with different labels dissimilar. Similarly, Cui et al. (2024) improved positive samples by augmenting anchors with features from samples of the same class to learn better representations based on explicit labels. Nonetheless, these approaches rely on discrete labels to determine positive or negative samples, which becomes extremely sparse for regression tasks where continuous labels are used.

To prevent NN from overfitting features, another line of research has utilized regularization techniques to avoid the model paying too much attention to a single feature. For instance, VIME (Yoon et al. 2020), SAINT (Somepalli et al. 2021), TabNet (Arik and Pfister 2021) and ReConTab (Chen et al. 2023b) use feature corruption, which encourages the model to output consistent predictions when features are randomly masked or mixed-up with other samples. However, randomly masking or mixing-up might inadvertently corrupt important features, causing the model to learn to pre-

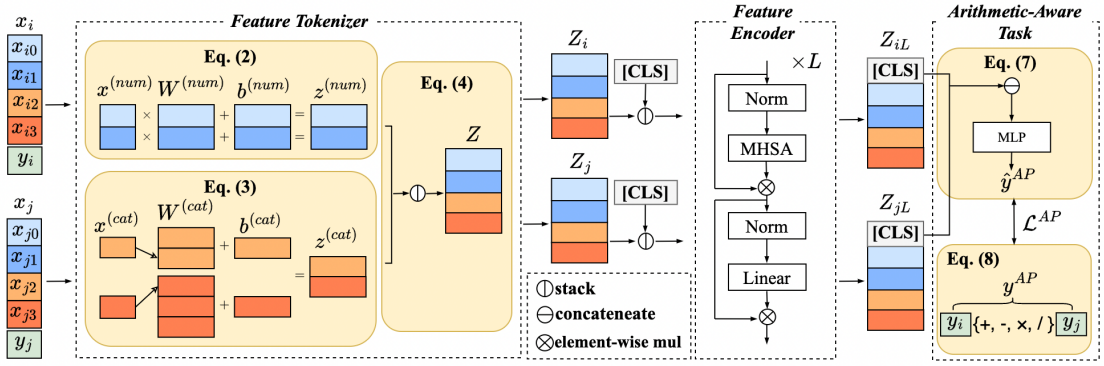



Figure 2: Illustration of the *Arithmetic-Aware Pre-Train* phase of APAR. Sample pairs are processed through the *Feature Tokenizer* and *Feature Encoder*, the outputs of which are concatenated for arithmetic prediction, enabling the model to understand inter-sample relationships in tabular regression.

dict based on uninformative features, and thus deteriorating learning representations. On the other hand, our proposed approach incorporates arithmetic from continuous labels as the pre-training task, and adaptively self-learns proper regularization during the fine-tuning stage.

## 3 Problem Formulation

In this paper, we focus on regression tasks within the tabular domain. A dataset is denoted as $D = \{(x_i, y_i)\}_{i=1}^{n}$, where $x_i = (x_{i1}^T, ..., x_{ik}^T) \in \mathbb{R}^k$ represents an object consisting of $k$ features, with $T \in \{num, cat\}$ indicating whether the feature is numerical or categorical. The corresponding label is denoted as $y_i \in \mathbb{R}$. Given an input sample $x$, our goal is to learn a representation $Z$ of the sample that effectively encapsulates both feature-wise and sample-wise information that is able to maintain robustness against heterogeneous and uninformative features to precisely predict the corresponding target $y$.

## 4 The Proposed Approach

The **APAR** framework employs a pretrain-finetune framework, outlined in Figures 2 (pre-training phase) and 3 (fine-tuning phase). Our APAR framework consists of two modules: the *Feature Tokenizer* and *Feature Encoder*. In the pre-training stage, a pair of two samples are encoded by the feature tokenizer and feature encoder to obtain their corresponding representations, and the contextualized [CLS] token is used for predicting arithmetic outcomes based on their numerical labels. In the fine-tuning stage, a test sample is augmented by applying it with a gate vector. Both the original and augmented samples are then encoded by the pre-trained feature tokenizer and feature encoder, generating two contextualized [CLS] tokens. The model is trained to adapt the gate vector based on self-learned feature importance, ensuring consistent predictions across the original and augmented samples.

### 4.1 Model Architecture

**Feature Tokenizer** To transform input features into representations, the feature tokenizer is introduced to convert categorical and numerical features of a sample into a sequence of embeddings. Similar to (Grinsztajn, Oyallon, and Varoquaux 2022), the feature tokenizer can prevent the rotational invariance of NNs by learning distinct embeddings for each feature.

Given $j$-th feature $x_{ij}$ of the $i$-th sample, the tokenizer generates a $d$-dimensional embedding $z_{ij} \in \mathbb{R}^d$. Formally, the embedding is computed as:

$$z_{ij} = b + f(x_{ij}) \in \mathbb{R}^d, \tag{1}$$

where $b$ is a bias term and $f$ represents a transformation function. For numerical features, $f$ involves an element-wise product with a weighting vector $W^{(num)} \in \mathbb{R}^d$:

$$z_{ij}^{(num)} = b_j^{(num)} + x_{ij}^{(num)} \cdot W_j^{(num)} \in \mathbb{R}^d. \tag{2}$$

For categorical features, $f$ applies a lookup in the embedding matrix $W^{(cat)} \in \mathbb{R}^{c \times d}$, where $c$ is the number of categories, and $e_{ij}$ is a one-hot vector for the corresponding categorical feature:

$$z_{ij}^{(cat)} = b_j^{(cat)} + e_{ij} \cdot W_j^{(cat)} \in \mathbb{R}^d. \tag{3}$$

Finally, the output from the feature tokenizer $Z_i$ is concatenated by the embeddings of all features of a sample $x_i$:

$$Z_i = \text{stack}[z_{i1}, ..., z_{ik}] \in \mathbb{R}^{k \times d}. \tag{4}$$

**Feature Encoder** Since our aim is to explore the strategies of the pre-training and fine-tuning stages similar to (Huang et al. 2020; Somepalli et al. 2021), the Transformer blocks encompassing multi-head self-attention and feed-forward networks are adopted as the feature encoder to encode intricate interrelations among heterogeneous and unformative features and to align the comparison.

Specifically, the embedding of a [CLS] token is first appended to the output $Z_i$ of the feature tokenizer, which is then fed into $L$ Transformer layers, $F_1, ..., F_L$:

$$\begin{aligned} Z_{i0} &= \text{stack}[\text{[CLS]}, Z_i], \\ Z_{il} &= F_l(Z_{i(l-1)}); l = 1, 2, ..., L. \end{aligned} \tag{5}$$

The output of the encoder can then be used to learn contextualized knowledge from the pre-training stage and downstream tasks from the fine-tuning stage.

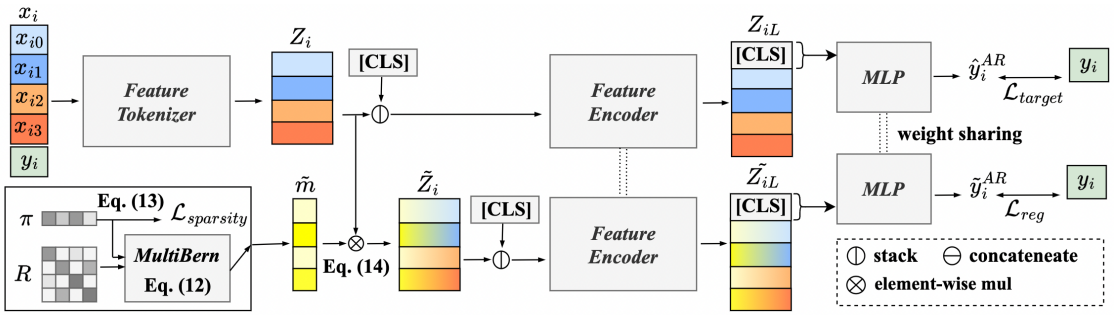



Figure 3: Illustration of the *Adaptive Regularization Fine-Tuning* phase of APAR. In this phase, an input sample is processed through the *Feature Tokenizer* to generate feature embeddings, which are augmented using a dynamically adaptive gate vector. The model is trained to predict consistent labels from varying inputs, which enhances the model's robustness to uninformative features and performance on the target task.

## 4.2 Arithmetic-Aware Pre-Training

The goal of the pre-training phase is to integrate sample-wise information into the representation of each sample; however, existing methods such as supervised contrastive learning (Khosla et al. 2020; Wang and Sun 2022; Cui et al. 2024) are ineffective in regression scenarios due to their reliance on discrete class labels, as opposed to regression's continuous labels. Also, simply relying on attention mechanisms (e.g., (Kossen et al. 2021; Somepalli et al. 2021)) underutilizes the relationship between label information across samples. To that end, we introduce a novel arithmetic-aware pretext task by conditioning continuous labels.

Analogous to solving for an unknown in a set of simultaneous equations in mathematics, our pre-training goal is to introduce constraints that are able to narrow the possible outcomes (i.e., search space) of the unknown. Therefore, the arithmetic-aware pre-training task is proposed to enable the model to discern relationships between samples by utilizing continuous labels in tabular regression. Intuitively, pairing sample A with different samples B, C, and D generates unique aggregated outcomes, such as A+B, A+C, and A+D. These pairs impose constraints and guide the model in learning a fine-grained representation of A that considers the context from not only itself but also other paired samples. In our work, we opt for a simple yet effective pre-training task by incorporating an arithmetic operator with two samples at a time as the pretext objective.

As shown in Figure 2, the arithmetic-aware pre-training process starts by selecting two random samples, $x_i$ and $x_j$, from the dataset, each with corresponding labels $y_i$ and $y_j$. These samples undergo processing through the *Feature Tokenizer* and *Feature Encoder* to produce their respective representations, $Z_{iL}$ and $Z_{jL}$:

$$\begin{aligned} Z_{i|j} &= \textit{FeatureTokenizer}(x_{i|j}), \\ Z_{iL|jL} &= \textit{FeatureEncoder}(\text{stack}[[\text{CLS}], Z_{i|j}]), \end{aligned} \tag{6}$$

where $i|j$ indicates the term is either the $i$- or $j$-th sample. Subsequently, the representations of the [CLS] token $Z_{iL}^{[\text{CLS}]}$ and $Z_{jL}^{[\text{CLS}]}$ are extracted from $Z_{iL}$ and $Z_{jL}$, respectively. They are then concatenated and fed into a Multilayer Perceptron (MLP) to predict the outcome $\hat{y}^{\text{AP}}$ of the arithmetic operation:

$$\hat{y}^{\text{AP}} = \text{MLP}(\text{concat}[Z_{iL}^{[CLS]}, Z_{jL}^{[CLS]}]). \tag{7}$$

The pre-training task involves applying arithmetic operations on the sample labels $y_i$ and $y_j$, including addition, subtraction, multiplication, and division[1]. The resulting ground truth $y^{\text{AP}}$ for the arithmetic task is represented as:

$$y^{\text{AP}} = \begin{cases} y_i + y_j, & \text{for addition;} \\ y_i - y_j, & \text{for subtraction;} \\ y_i \times y_j, & \text{for multiplication;} \\ y_i / y_j, & \text{for division.} \end{cases} \tag{8}$$

Finally, the model is then trained to minimize $\mathcal{L}^{\text{AP}}$:

$$\mathcal{L}^{\text{AP}} = \frac{1}{n} \sum^{n} (y^{\text{AP}} - \hat{y}^{\text{AP}})^2. \tag{9}$$

This pre-training task embeds an awareness of arithmetic relationships between samples into the model, thereby enabling it to adeptly handle the irregularities of target functions. The detailed procedure of arithmetic-aware pre-training is summarized in Algorithm 1 in Appendix A.1.

## 4.3 Adaptive-Regularized Fine-Tuning

In the fine-tuning phase of APAR, we introduce an adaptive-regularized fine-tuning method that adaptively augments samples by considering feature importance and their correlated structure. As shown in Figure 3, an input sample is processed by the pre-trained feature tokenizer to produce feature embeddings, which are subsequently augmented using a dynamically adaptive gate vector. The model is fine-tuned to predict a consistent label from these variant inputs, which enables the model to perform data augmentation, guided by the model-learned importance of each feature, to improve performance on the downstream task.

[1]We empirically chose arithmetic operations in our experiments (See Sec. 5.5 for detailed analyses).

**Adaptive Learning** When learning the importance of each feature, we consider the correlation structure of the features rather than assuming independence, as the feature selection process is influenced by these correlations (Katrutsa and Strijov 2017). Specifically, a correlated gate vector for augmenting the input sample is generated from a multivariate Bernoulli distribution where the mean is determined by learnable parameters reflecting feature importance. The distribution is jointly updated when fine-tuning the model to adaptively utilize the self-learned gate vector to augment the input sample.

Specifically, given the correlation matrix $R \in [-1, 1]^{k \times k}$ representing the correlation structure, the Gaussian copula is defined as:

$$C_R(U_1, ..., U_k) = \Phi_R(\Phi^{-1}(U_1), ..., \Phi^{-1}(U_k)), \quad (10)$$

where $\Phi_R$ denotes the joint cumulative distribution function (CDF) of a multivariate Gaussian distribution with a mean zero vector and correlation matrix $R$, $\Phi^{-1}$ is the inverse CDF of the standard univariate Gaussian distribution, and $U_j \sim Uniform(0, 1)$ for $j \in [k]$.

Afterwards, we sample a gate vector $m$ to augment the input sample from a multivariate Bernoulli distribution that maintains the correlation structure of the input features as:

$$m \sim MultiBern(\pi; R). \quad (11)$$

Formally, $m_j$ is set to 1 if $U_j \leq \pi_j$ and 0 if $U_j > \pi_j$, for $j \in [k]$. Here, $\pi$ represents a set of learnable parameters indicating feature importance. For differentiability, we apply the reparametrization trick (Wang and Yin 2020), resulting in a relaxed gate vector $\tilde{m}$ from the following equation:

$$\tilde{m}_j = \sigma(\frac{1}{\tau}(\log \pi_j - \log(1 - \pi_j) + \log U_j - \log(1 - U_j))), \quad (12)$$

where $\sigma(x) = (1 + \exp(-x))^{-1}$ is the sigmoid function, and a temperature parameter $\tau \in (0, \infty)$.

The relaxed gate vector $\tilde{m}$ is then used to be multiplied with the feature embeddings for data augmentation. The feature selection probability $\pi$ is learned during training and is adjusted according to the performance of the model. To induce the sparsity of the selected features, the sparsity loss, $\mathcal{L}_{\text{sparsity}}$ is calculated as follows:

$$\mathcal{L}_{\text{sparsity}} = \sum_{i}^{k} \pi_i. \quad (13)$$

**Fine-Tuning Loss Function** As depicted in Figure 3, for each input sample $x_i$ with the label $y_i$, it is processed by the feature tokenizer and feature encoder to obtain the representation of the sample $Z_{iL}$, as detailed in Equation (6). Simultaneously, the original feature embeddings $Z_i$ are element-wise multiplied with the relaxed gate vector $\tilde{m}$ to obtain augmented feature embeddings $\tilde{Z}_i$, as shown below:

$$\tilde{Z}_i = Z_i \odot \tilde{m}. \quad (14)$$

These augmented embeddings $\tilde{Z}_i$ are then stacked with the [CLS] token and input into the feature encoder to obtain the augmented representation $\tilde{Z}_{iL}$:

| | BD | AM | HS | GS | ER |
|---|---|---|---|---|---|
| #instances | 73203 | 60786 | 61784 | 36733 | 21643 |
| #num feats | 230 | 230 | 230 | 10 | 23 |
| #cat feats | 17 | 17 | 17 | 0 | 2 |
| | **PM** | **BS** | **YE** | **KP** | **FP** |
| #instances | 41757 | 17389 | 515345 | 241600 | 300153 |
| #num feats | 11 | 7 | 90 | 0 | 2 |
| #cat feats | 1 | 8 | 0 | 14 | 7 |

Table 1: Statistics of each dataset.

$$\tilde{Z}_{iL} = \textit{FeatureEncoder}(\text{stack}[[\text{CLS}], \tilde{Z}_i]). \quad (15)$$

The representation of the [CLS] token $Z_{iL}^{[\text{CLS}]}$ and $\tilde{Z}_{iL}^{[\text{CLS}]}$, extracted from $Z_{iL}$ and $\tilde{Z}_{iL}$, respectively, are input into an MLP to generate the corresponding predictions $\hat{y}_i^{\text{AR}}$ and $\tilde{y}_i^{\text{AR}}$, as described by the following equations:

$$\hat{y}_i^{\text{AR}} = \text{MLP}(Z_{iL}^{[\text{CLS}]}), \tilde{y}_i^{\text{AR}} = \text{MLP}(\tilde{Z}_{iL}^{[\text{CLS}]}). \quad (16)$$

The losses for the target task $\mathcal{L}_{\text{target}}$ and the regularization $\mathcal{L}_{\text{reg}}$ are computed as follows:

$$\mathcal{L}_{\text{target}} = \frac{1}{n}\sum_{i}^{n}(y_i - \hat{y}_i^{\text{AR}})^2, \mathcal{L}_{\text{reg}} = \frac{1}{n}\sum_{i}^{n}(y_i - \tilde{y}_i^{\text{AR}})^2. \quad (17)$$

During the fine-tuning phase, the total loss, $\mathcal{L}^{\text{AR}}$, is the weighted sum of the target task loss, regularization loss, and feature sparsity loss, defined as:

$$\mathcal{L}^{\text{AR}} = \alpha\mathcal{L}_{\text{target}} + \beta\mathcal{L}_{\text{reg}} + \gamma\mathcal{L}_{\text{sparsity}}, \quad (18)$$

where $\alpha$, $\beta$, and $\gamma$ are hyperparameters within the range $[0, 1]$. The procedure of adaptive-regularized fine-tuning is summarized in Algorithm 3 in Appendix A.3.

# 5 Experiments

In this section, we attempt to answer the following research questions on a wide range of real-world datasets:

- **RQ1:** Does our proposed framework, APAR, outperform the existing NN-based and GBDT-based approaches?
- **RQ2:** How does the performance of the proposed arithmetic-aware pre-training task compare to other pre-training approaches in tabular regression?
- **RQ3:** Does the adaptive regularization enhance the model's performance during the fine-tuning phase?
- **RQ4:** How do different arithmetic operations affect the performance across various scenarios?

## 5.1 Experimental Setup

**Datasets Overview.** In our experiments, we utilized 10 publicly real-world datasets across diverse tabular regression applications (i.e., property valuation, environmental monitoring, urban applications, and performance analysis), spanning a range of scales from large-scale (over 100K samples) to medium-scale (50K to 100K samples) and small-

| Group | | BD | AM | HS | GS | ER | PM | BS | YE | KP | FP | Rank |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| GBDT-based | XGB | 0.2476 | 0.2472 | 0.3509 | 0.1489 | 0.0510 | 0.6075 | 0.0244 | 0.2166 | 0.2559 | 0.4066 | 6.0 |
| | LGBM | 0.2506 | 0.2429 | 0.3354 | 0.1575 | 0.0693 | 0.7445 | 0.0458 | 0.2156 | 0.3718 | 0.4213 | 6.5 |
| | CB | 0.2406 | 0.2441 | 0.3423 | 0.1526 | 0.0557 | 0.7398 | 0.0469 | 0.2175 | 0.3087 | 0.4460 | 6.3 |
| Supervised NN-based | MLP | 0.2728 | 0.2617 | 0.4314 | 0.1743 | 0.0499 | 0.6973 | 0.0244 | 0.2179 | **0.0500** | 0.3659 | 6.6 |
| | AutoInt | 0.2498 | 0.2456 | 0.3510 | 0.1755 | 0.1549 | 0.7790 | 0.0974 | 0.2161 | 0.0830 | 0.3843 | 7.7 |
| | FT-T | 0.2452 | 0.2352 | 0.3457 | 0.1710 | 0.0538 | 0.8160 | 0.0591 | 0.2163 | 0.0547 | 0.3421 | 5.5 |
| | TabNet* | 0.2435 | 0.2502 | 0.3467 | 0.1249 | 0.0838 | 0.5537 | 0.0591 | **0.2114** | 0.1849 | 0.3657 | 5.1 |
| NN-based with a Pretrain-Finetune | VIME | 0.2412 | 0.2606 | 0.3746 | 0.1266 | 0.0422 | 0.6557 | 0.1360 | 0.2184 | 0.1766 | 0.3685 | 6.4 |
| | TabNet | 0.2404 | 0.2427 | 0.3333 | 0.1352 | 0.0846 | 0.5880 | 0.0479 | 0.2126 | 0.0566 | 0.3562 | 3.8 |
| | APAR | **0.2397** | **0.2293** | **0.3305** | **0.1205** | **0.0338** | **0.5239** | **0.0139** | 0.2148 | **0.0500** | **0.3303** | **1.3** |

Table 2: Quantitative results of all groups of baselines and our proposed APAR. For each dataset, the best result in each column is in boldface, while the second best result is underlined. * denotes without pre-training.

scale (less than 50K samples). The datasets include Taiwan Housing (BD, AM, HS) (M.O.I. Dept 2023) consisting of three building types (building, apartment, and house), Gas Emission (GS) (mis 2019a), Election Results (ER) (mis 2019b), Beijing PM2.5 (PM) (Chen 2017), Bike Sharing (BS) (Fanaee-T 2013), Year (YE) (Bertin-Mahieux 2011), Kernel Performance (KP) (Paredes and Ballester-Ripoll 2018), and Flight Price (FP) (Bathwal 2021). Each dataset presents unique characteristics, including variations in features and label ranges. A summary of the dataset characteristics is presented in Table 1, and we follow their corresponding protocols to split training, validation, and test sets.

Categorical features of each dataset were processed through label encoding (Hancock and Khoshgoftaar 2020), except in CatBoost (Prokhorenkova et al. 2018) where built-in categorical feature support was used, while continuous features and labels were transformed using logarithmic scaling (Changyong et al. 2014). For NN-based approaches, we utilize uniformly dimensioned embeddings for all categorical features.

**Baseline Methods.** We compared APAR against several baselines categorized into three groups: 1) GBDT-based: **XGBoost (XGB)** (Chen and Guestrin 2016), **LightGBM (LGBM)** (Ke et al. 2017) and **CatBoost (CB)** (Prokhorenkova et al. 2018). 2) Supervised NN-based: **MLP**, **AutoInt** (Song et al. 2019), and **FT-Transformer (FT-T)** (Gorishniy et al. 2021). 3) NN-based with a pretrain-finetune strategy: **VIME** (Yoon et al. 2020) and **TabNet** (Arik and Pfister 2021).

**Implementation Details.** Our proposed APAR framework was developed using PyTorch version 1.13.1. The training was performed on an NVIDIA GeForce RTX 3090 GPU. Regarding optimizers, we followed the original TabNet implementation by using the Adam optimizer (Kingma and Ba 2014). For all other models, we employed the AdamW optimizer (Loshchilov and Hutter 2018) with $\beta_1 = 0.9$, $\beta_2 = 0.999$, and a weight decay of 0.01. A consistent StepLR scheduler was used for all deep learning models, and the batch size was set at 256 for each dataset and algorithm. Training continued until there was no improvement on the validation set for 10 consecutive epochs.

**Evaluation Metrics.** Following (Gorishniy et al. 2021), we used the root mean squared error (RMSE) for evaluating regression models. The ranks for each dataset were determined by sorting the scores obtained. The $Rank$ reflects the average rank across all datasets. All the results are the average of 5 different random seeds.

## 5.2 Quantitative Results (RQ1)

Table 2 presents the quantitative performance of APAR and the baselines. Quantitatively, APAR was consistently superior to all approaches in overall ranking across 10 diverse datasets, achieving an average RMSE improvement of 9.18% compared to the second-best ranking method. We summarize the observations as follows:

**Selection of the Feature Encoder.** We can observe that TabNet(*) and FT-Transformer demonstrate better performance compared with the other baselines in all three categories since they adopt Transformer architectures as their backbones to model intricate characteristics across tabular samples as well as features. Nonetheless, the comparison of APAR, which employs the Transformer architecture in the feature encoder, and these baselines reveals the importance of considering the advantage of learning contextualized representations in a two-stage manner.

**Advantages of the Pretrain-Finetune Approach.** It is evident that comparing TabNet* with TabNet illustrates notable improvements with pre-training, demonstrating the value of the pretrain-finetune framework. However, VIME substantially hinders all performance due to not only the relatively simplified MLP architecture but also the lack of considering feature heterogeneity and rotational invariance, which again raises the need for leveraging the Transformer architecture with a feature tokenizer for tabular regression tasks. The effectiveness of our APAR highlights the capability of arithmetic-related pertaining tasks and adaptively learning contexts of features during the finetuning stage.

## 5.3 Effects of the Pre-Training Task (RQ2)

To testify the design of the pre-training task in APAR, we evaluate arithmetic-aware pre-training with four variants: 1) remove (w/o AP), replacing it with 2) feature reconstruction

| | | BD | AM | HS | GS | ER | PM | BS | YE | KP | FP |
|---|---|---|---|---|---|---|---|---|---|---|---|
| RQ2 | w/o AP | 0.2520 | 0.2377 | 0.3474 | 0.1549 | 0.0462 | 0.6173 | 0.0224 | 0.2175 | 0.0574 | 0.3603 |
| | AP $\rightarrow$ FR | 0.2468 | 0.2512 | 0.3530 | 0.1259 | 0.0297 | 0.5758 | 0.0173 | 0.2148 | 0.0548 | 0.3409 |
| | AP $\rightarrow$ MR | 0.2464 | 0.2464 | 0.3582 | 0.1956 | 0.1582 | 0.6403 | 0.0141 | 0.2198 | 0.0728 | 0.3718 |
| | AP $\rightarrow$ FR + MR | 0.2508 | 0.2319 | 0.3530 | 0.1360 | **0.0266** | 0.5729 | 0.0140 | **0.2112** | 0.0548 | 0.3406 |
| RQ3 | w/o AR | 0.2536 | 0.2383 | 0.3501 | 0.1240 | **0.0266** | 0.5955 | 0.0632 | 0.2161 | 0.0520 | 0.3344 |
| | APAR (Ours) | **0.2397** | **0.2293** | **0.3305** | **0.1205** | **0.0266** | **0.5239** | **0.0139** | 0.2148 | **0.0500** | **0.3303** |

Table 3: Ablative experiments of different pre-training tasks (RQ2) and adaptive regularization (RQ3).

| | BD | AM | HS | GS | ER | PM | BS | YE | KP | FP | Rank |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Addition | 0.2495 | **0.2293** | **0.3305** | 0.1453 | 0.0338 | **0.5239** | 0.0182 | 0.2158 | **0.0500** | **0.3303** | 2.2 |
| Subtraction | 0.2408 | 0.2413 | 0.3422 | 0.1250 | 0.0287 | 0.5783 | 0.0147 | 0.2199 | 0.0505 | 0.3458 | 2.5 |
| Multiplication | **0.2397** | 0.2317 | 0.3458 | **0.1205** | **0.0266** | 0.5863 | **0.0139** | **0.2148** | 0.0506 | 0.3321 | **1.9** |
| Division | 0.2469 | - | 0.3375 | 0.1420 | - | - | - | 0.2149 | 0.0596 | - | 3.4 |

Table 4: Performance of using different arithmetic operations in Arithmetic-Aware Pre-Training. "-" indicates that the model did not converge during the pre-training phase.

(AP $\rightarrow$ FR), 3) mask reconstruction (AP $\rightarrow$ MR), and 4) AP $\rightarrow$ FR+MR. Feature reconstruction is pre-trained to reconstruct with corrupt samples, which are randomly inserted constant values to some features. Mask reconstruction is pre-trained to predict the correct binary mask applied to the input sample, which aims to identify which parts of the input have been replaced. As shown in Table 3, it is obvious that removing the pre-training task degrades the performance for all scenarios. The deleterious effect of replacing our method with MR is due to the inadvertent masking of key features, which shifts the model's reliance to less relevant sample details and overlooks inter-sample relationships. Although combining FR with MR improves performance, a significant gap remains compared to our arithmetic-aware pre-training task, indicating that utilizing the continuous labels in the regression scenario to design arithmetic tasks effectively encourages the model to consider inter-sample relationships.

### 5.4 Effects of Adaptive Regularization (RQ3)

To investigate the impact of incorporating adaptive-regularized fine-tuning in APAR, the performance of the removal of this design (w/o AR) was compared, as shown in the RQ3 row in Table 3. Specifically, we fixed the target task loss weight $\alpha$ at 1 and optimized the regularization loss weight $\beta$ and sparsity loss weight $\gamma$ using the validation dataset. Removing the adaptive-regularized technique causes the model to be prone to overfitting on uninformative features, which degrades the performance across all datasets. In contrast, APAR mitigates this limitation by adaptive regularization, leading to a substantial improvement.

### 5.5 Variants of Arithmetic Operations (RQ4)

We studied the performance of addition, subtraction, multiplication, and division across all datasets, as detailed in Table 4. It can be seen that both addition and multiplication operations are more effective than subtraction and division operations, indicating positively changing the representation of two numerical labels introduces less offset of information to learn relations compared with negatively changing. In addition, using either addition or multiplication may depend on the scale of the labels of the dataset. For example, if the labels are small (e.g., $< 1$), it is expected that all multiplication pairs will become near 0, leading to ambiguity for model learning. Moreover, division-based pre-training was the least consistent, often failing to converge during pre-training, as indicated by the "-" symbol. This is because the divided changes are too significant to learn the relations. These results highlight the adaptability of the arithmetic-aware pre-training method that is able to benefit different regression scenarios from various arithmetic operators.

## 6 Conclusion and Future Works

This paper proposes APAR, a novel arithmetic-aware pre-training and adaptive-regularized fine-tuning framework for tabular regression tasks. Distinct from existing works that ineffectively corrupt important features and transfer to regression labels due to the sparsity of the continuous space, our proposed pre-training task is able to take sample-wise interactions into account, allowing the capability of modeling from the aspects of continuous labels. Meanwhile, our adaptive-regularized fine-tuning design dynamically adjusts appropriate data augmentation by conditioning it on the self-learned feature importance. Experiments on 10 real-world datasets show that our APAR significantly outperforms state-of-the-art approaches by between 9.43% to 20.37%. We believe that APAR serves as a generic framework for tabular regression applications due to the flexible design for pretrain-finetune frameworks, and multiple interesting directions could be further explored within the framework, such as automatically selecting appropriate arithmetic operations for effective pre-training, or extending APAR to classification tasks with Boolean operations (e.g., AND), etc.

## Acknowledgments

We would like to thank Fu-Chang Sun, Yi-Hsun Lin, Hsien-Chin Chou from E.SUN Bank for sharing data and discussing the findings.

## References

2019a. Gas Turbine CO and NOx Emission Data Set. UCI Machine Learning Repository. DOI: https://doi.org/10.24432/C5WC95.

2019b. Real-time Election Results: Portugal 2019. UCI Machine Learning Repository. DOI: https://doi.org/10.24432/C5NS5M.

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

Bathwal, S. 2021. Flight Price Prediction.

Bertin-Mahieux, T. 2011. YearPredictionMSD. UCI Machine Learning Repository. DOI: https://doi.org/10.24432/C50K61.

Beyazit, E.; Kozaczuk, J.; Li, B.; Wallace, V.; and Fadlallah, B. 2023. An Inductive Bias for Tabular Deep Learning. In *NeurIPS*.

Changyong, F.; Hongyue, W.; Naiji, L.; Tian, C.; Hua, H.; Ying, L.; et al. 2014. Log-transformation and its implications for data analysis. *Shanghai archives of psychiatry*, 26(2): 105.

Chen, J.; Yan, J.; Chen, D. Z.; and Wu, J. 2023a. Excelformer: A neural network surpassing gbdts on tabular data. *arXiv preprint arXiv:2301.02819*.

Chen, S. 2017. Beijing PM2.5. UCI Machine Learning Repository. DOI: https://doi.org/10.24432/C5JS49.

Chen, S.; Wu, J.; Hovakimyan, N.; and Yao, H. 2023b. ReConTab: Regularized Contrastive Representation Learning for Tabular Data. *CoRR*, abs/2310.18541.

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.

Cui, W.; Hosseinzadeh, R.; Ma, J.; Wu, T.; Sui, Y.; and Golestan, K. 2024. Tabular Data Contrastive Learning via Class-Conditioned and Feature-Correlation Based Augmentation. *CoRR*, abs/2404.17489.

Deng, S.; Su, J.; Zhu, Y.; Yu, Y.; and Xiao, C. 2024. Forecasting carbon price trends based on an interpretable light gradient boosting machine and Bayesian optimization. *Expert Systems with Applications*, 242: 122502.

Du, W.; Wang, W.; and Peng, W. 2023. DoRA: Domain-Based Self-Supervised Learning Framework for Low-Resource Real Estate Appraisal. In *CIKM*, 4552–4558. ACM.

Fanaee-T, H. 2013. Bike Sharing Dataset. UCI Machine Learning Repository. DOI: https://doi.org/10.24432/C5W894.

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

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

Hancock, J. T.; and Khoshgoftaar, T. M. 2020. Survey on categorical data for neural networks. *Journal of Big Data*, 7(1): 1–41.

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

Jain, R.; Singh, M.; Rao, A. R.; and Garg, R. 2024. Predicting hospital length of stay using machine learning on a large open health dataset. *BMC Health Services Research*, 24(1): 860.

Katrutsa, A.; and Strijov, V. 2017. Comprehensive study of feature selection methods to solve multicollinearity problem according to evaluation criteria. *Expert Systems with Applications*, 76: 1–11.

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. *Advances in neural information processing systems*, 30.

Khosla, P.; Teterwak, P.; Wang, C.; Sarna, A.; Tian, Y.; Isola, P.; Maschinot, A.; Liu, C.; and Krishnan, D. 2020. Supervised contrastive learning. *Advances in neural information processing systems*, 33: 18661–18673.

Kingma, D. P.; and Ba, J. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Kossen, J.; Band, N.; Lyle, C.; Gomez, A. N.; Rainforth, T.; and Gal, Y. 2021. Self-attention between datapoints: Going beyond individual input-output pairs in deep learning. *Advances in Neural Information Processing Systems*, 34: 28742–28756.

Loshchilov, I.; and Hutter, F. 2018. Fixing weight decay regularization in adam.

M.O.I. Dept, L. A. 2023. Taiwan Real Estate Transaction Platform.

Paredes, E.; and Ballester-Ripoll, R. 2018. SGEMM GPU kernel performance. UCI Machine Learning Repository. DOI: https://doi.org/10.24432/C5MK70.

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

Rao, A. R.; Jain, R.; Singh, M.; and Garg, R. 2023. Machine Learning Models For Patient Medical Cost Prediction and Trend Analysis Using Open Healthcare Data. In *2023 IEEE 3rd International Conference on Electronic Communications, Internet of Things and Big Data (ICEIB)*, 292–296. IEEE.

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

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. In *Proceedings of the 28th ACM international conference on information and knowledge management*, 1161–1170.

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

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, Ł.; and Polosukhin, I. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.

Wang, W.; Du, W.; Xu, D.; Wang, W.; and Peng, W. 2024. A Survey on Self-Supervised Learning for Non-Sequential Tabular Data. *CoRR*, abs/2402.01204.

Wang, X.; and Yin, J. 2020. Relaxed multivariate bernoulli distribution and its applications to deep generative models. In *Conference on Uncertainty in Artificial Intelligence*, 500–509. PMLR.

Wang, Z.; and Sun, J. 2022. Transtab: Learning transferable tabular transformers across tables. *Advances in Neural Information Processing Systems*, 35: 2902–2915.

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

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