AdapTable: Test-Time Adaptation for Tabular Data via Shift-Aware Uncertainty Calibrator and Label Distribution Handler

Anonymous Author(s)
Affiliation
Address
email

Abstract

In real-world scenarios, tabular data often suffer from distribution shifts that threaten the performance of machine learning models. Despite its prevalence and importance, handling distribution shifts in the tabular domain remains underexplored due to the inherent challenges within the tabular data itself. In this sense, test-time adaptation (TTA) offers a promising solution by adapting models to target data without accessing source data, crucial for privacy-sensitive tabular domains. However, existing TTA methods either 1) overlook the nature of tabular distribution shifts, often involving label distribution shifts, or 2) impose architectural constraints on the model, leading to a lack of applicability. To this end, we propose AdapTable, a novel TTA framework for tabular data. AdapTable operates in two stages: 1) calibrating model predictions using a shift-aware uncertainty calibrator, and 2) adjusting these predictions to match the target label distribution with a label distribution handler. We validate the effectiveness of AdapTable through theoretical analysis and extensive experiments on various distribution shift scenarios. Our results demonstrate AdapTable's ability to handle various real-world distribution shifts, achieving up to a 16% improvement on the HELOC dataset.

1 Introduction

Tabular data is one of the most abundant forms across various industries, including healthcare [27], finance [48], manufacturing [23], and public administration [17]. However, tabular learning models often face challenges in real-world applications due to distribution shifts, which severely degrade their integrity and reliability. In this regard, test-time adaptation (TTA) [29, 33, 53, 37, 38, 6] offers a promising solution to address this issue by adapting models under unknown distribution shifts using only unlabeled test data without access to training data.

Despite its potential, direct application of TTA without the consideration of characteristics of tabular data, results in limited performance gain or model collapse. We identify two primary reasons for this. First, representation learning in the tabular domain is often hindered by the entanglement of covariate shifts and concept shifts [32]. Consequently, TTA methods leveraging unsupervised objectives, which rely on cluster assumption often fail or lead to model collapse. Second, these approaches often do not take label distribution shifts into account, a key factor in the performance decline within the tabular domain. This issue is further aggravated by the tendency for predictions in the target domain to be biased towards the source label.

To address these issues, we propose AdapTable, a novel TTA approach tailored for tabular data. AdapTable consists of two main components: 1) a shift-aware uncertainty calibrator and 2) a label distribution handler. Our shift-aware uncertainty calibrator utilizes graph neural networks to assign

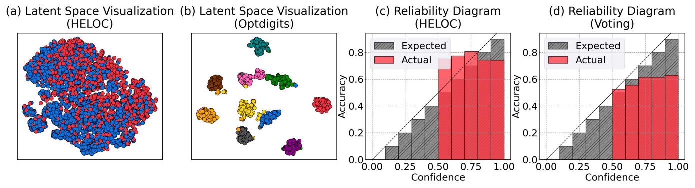

Figure 1: Latent space visualization with t-SNE comparing (a) tabular data [17] and (b) image data [5]. Reliability diagrams of (c) underconfident and (d) overconfident scenarios are shown. All experiments are conducted using an MLP architecture.

per-sample temperature for each model prediction. By treating each column as a node, it captures not only individual feature shifts but also complex patterns across features. Our label distribution handler then adjusts the calibrated model probabilities by estimating the label distribution of the current target batch. This process aligns predictions with the target label distribution, addressing biases towards the source distribution. AdapTable requires no parameter updates, making it model-agnostic and thus compatible with both deep learning models and gradient-boosted decision trees, offering high versatility for tabular data.

We evaluate AdapTable under various distribution shifts and demonstrate AdapTable consistently outperforms baselines, achieving up to 16% gains on the HELOC dataset. Furthermore, we provide theoretical insights into AdapTable's performance, supported by extensive ablation studies. We hereby summarize our contributions:

- We analyze the challenges of tabular distribution shifts to reveal why existing TTA methods fail, highlighting the entanglement of covariate, concept shifts, and label distribution shifts as key factors in performance degradation.
- Building on these analyses, we introduce AdapTable, a first model-agnostic TTA method specifically designed for tabular data. AdapTable addresses label distribution shifts by estimating and adjusting the label distribution of the current test batch, while also calibrating model predictions with a shift-aware uncertainty calibrator.
- Our extensive experiments demonstrate that AdapTable exhibits robust adaptation performance across various model architectures and under diverse natural distribution shifts and common corruptions, further supported by extensive ablation studies.

2 Analysis of Tabular Distribution Shifts

In this section, we examine why prior TTA methods struggle with distribution shifts in the tabular domain. First, we note that deep learning models' latent representations do not follow label-based cluster assumptions due to the entanglement of covariate and concept shifts, causing TTA methods relying on these assumptions [53, 29, 37] to falter in tabular data. Second, we identify label distribution shift as a key driver of performance degradation under distribution shifts, as discussed further in Section 2.1 and Section 2.2.

2.1 Indistinguishable Representations

We first reveal that deep tabular models fail to learn distinguishable embeddings. In Figures 1 (a) and (b), we visualize the embedding spaces of models trained on two datasets: HELOC [17], a pure tabular dataset, and Optdigits [5], a linearized image dataset. Notably, the deep learning models' representations adhere to the cluster assumption by labels only in the image data, not in the tabular data.

We attribute this unique behavior of deep tabular models to the high-frequency nature of tabular data. In the tabular domain, weak causality from inputs X to outputs Y due to latent confounders Z often leads to vastly different labels for similar inputs [20, 32]. For instance, cardiovascular disease risk predictions based on cholesterol, blood pressure, age, and smoking history are influenced by gender as a latent confounder, resulting in different risk levels for men and women despite identical inputs [35, 10]. This leads to high-frequency functions that are difficult for deep neural networks, which are biased toward low-frequency functions, to accurately model [4].

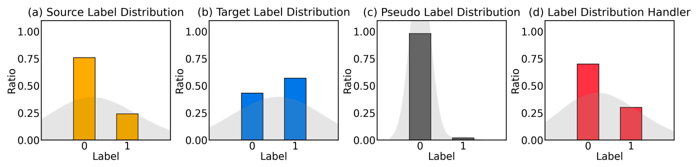

Figure 2: Label distribution of (a) source domain, (b) target domain, (c) estimated label distribution using pseudo labels, and (d) corrected label distribution of AdapTable are shown using MLP on HELOC dataset.

Consequently, prior TTA methods, which rely on cluster assumptions and primarily target input covariate shifts, show limited performance gains. Figure 7 demonstrates that these methods fail to improve beyond the vanilla performance of the source model due to the lack of a cluster assumption.

2.2 Importance of Label Distribution Shifts

Second, we find that label distribution shift is a primary cause of performance degradation, and accurate estimation of target label distribution can lead to significant performance gains. A recent benchmark study, TableShift [17] have emphasized that label distribution shift is a primary cause of performance degradation in tabular data. Specifically, They investigated the relationship between three key shift factors—input covariate shift (X-shift), concept shift ($Y|X$-shift), and label distribution shift (Y-shift)—and model performance, and discovered that label distribution shifts are strongly correlated with performance degradation. Our analysis in Section G further reveals that these shifts are highly prevalent in tabular data. This underscores the need for a test-time adaptation method that addresses label distribution shifts by estimating the target label distribution and adjusting predictions accordingly.

Moreover, we visualize model predictions in Figure 2 and observe that, similar to other domains [55, 25, 39], the marginal distribution of output labels is biased toward the source label distribution. Given that tabular models are often poorly calibrated (Figure 1), we conduct an experiment using a perfectly calibrated model, which yields high confidence for correct samples and low confidence for incorrect ones. As shown in Table 1, our label distribution adaptation method significantly improves under these conditions. This underscores the need for an uncertainty calibrator specific to tabular data.

Table 1: Key findings demonstrate that uncertainty calibration enhances the performance of the label distribution handler.

Method	HELOC	Voting
Source	47.6	79.3
AdapTable	<u>63.7</u>	<u>79.6</u>
AdapTable (Oracle)	**90.1**	**84.7**

3 AdapTable

This section introduces AdapTable, the first model-agnostic test-time adaptation strategy for tabular data. AdapTable uses per-sample temperature scaling to correct overconfident yet incorrect predictions; by treating each column as a graph node, it employs a shift-aware uncertainty calibrator with graph neural networks to capture both individual and complex feature shifts (Section 3.2). It also estimates the average label distribution of the current test batch and adjusts the model's output predictions accordingly (Section 3.3). We also provide a theoretical justification for how our label distribution estimation reduces the error bound in Section 3.4. The overall framework of AdapTable is depicted in Figure 3.

3.1 Test-Time Adaptation Setup for Tabular Data

We begin by defining the problem setup for test-time adaptation (TTA) for tabular data. Let $f_\theta : \mathbb{R}^D \to \mathbb{R}^C$ be a pre-trained classifier on the labeled source tabular domain $\mathcal{D}_s = \{(\mathbf{x}_i^s, y_i^s)\}_i \subset X_s \times Y_s$, where each pair consists of a tabular input $\mathbf{x}_i^s \in \mathcal{X} = \mathbb{R}^D$ and its corresponding output class label $y_i^s \in \mathcal{Y} = \{1, \cdots, C\}$. The classifier takes a row $\mathbf{x}_i \in \mathbb{R}^D$ from a table and returns output logit $f_\theta(\mathbf{x}_i) \in \mathbb{R}^C$. Here, D and C are the number of input features and output classes, respectively. The objective of TTA for tabular data is to adapt f_θ to the unlabeled target tabular domain $\mathcal{D}_t = \{\mathbf{x}_i^t\}_i = X_t$ during inference, without access to $\mathcal{D}_s$. Unlike most TTA methods that

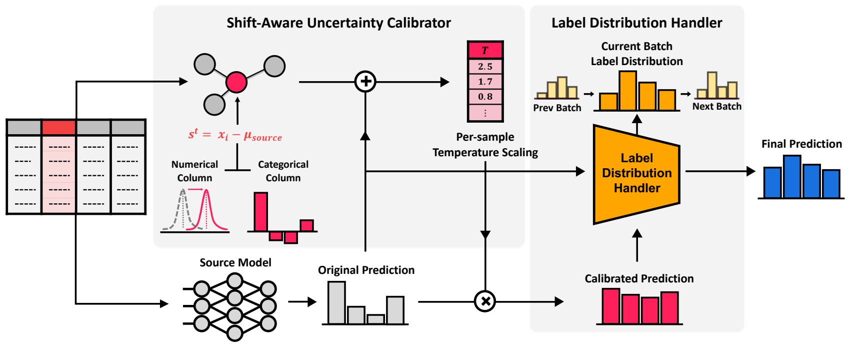

Figure 3: The overall pipeline of the AdapTable framework. AdapTable employs a per-sample temperature scaling to correct overconfident predictions by treating each column as a graph node, enabling a shift-aware uncertainty calibrator with graph neural networks to capture both individual and complex feature shifts (Section 3.2). It also estimates the label distribution of the current test batch and adjusts the model's predictions accordingly (Section 3.3).

fine-tune model parameters θ with unsupervised objectives, our approach directly adjusts the output prediction $f_\theta(\mathbf{x}_i^t)$.

3.2 Shift-Aware Uncertainty Calibrator

This section describes a shift-aware uncertainty calibrator $g_\phi : \mathbb{R}^C \times \mathbb{R}^{D\times N} \to \mathbb{R}^+$ designed to adjust the poorly calibrated original predictions $p_t(y|\mathbf{x}_i^t) = \text{softmax}(f_\theta(\mathbf{x}_i^t))$, where $\text{softmax}(z)_i = \exp(z_i)/\sum_{i'} \exp(z_{i'})$ normalizes the logits. Our shift-aware uncertainty calibrator lowers the confidence of overconfident yet incorrect predictions, thereby 1) facilitating better alignment of these predictions with the estimated target label distribution, and 2) mitigating their impact on the inaccurate estimation of the target label distribution.

Conventional post-hoc calibration methods [41, 50] typically take solely the original model prediction $f_\theta(\mathbf{x}_i^t)$ as input and return the corresponding temperature T_i without taking input variations into account. We argue that this can be suboptimal as it fails to account for the uncertainty arising from variations in the input itself. Instead, our g_ϕ not only considers $f_\theta(\mathbf{x}_i^t)$ but also incorporates $\mathbf{x}_i^t$ with the shift trend $\mathbf{s}^t$ of the current batch as additional inputs. Capturing the common shift patterns within the current batch enables a more accurate reflection of the uncertainty caused by the overall shift patterns within the current batch.

In detail, the shift trend $\mathbf{s}^t = (\mathbf{s}_u^t)_{u=1}^D \in \mathbb{R}^{D\times N}$ is defined for a specific column index u as follows:

$$\mathbf{s}_u^t = \Big(\mathbf{x}_{iu}^t - \frac{1}{|\mathcal{D}_s|}\sum_{i'=1}^{|\mathcal{D}_s|} \mathbf{x}_{i'u}^s\Big)_{i=1}^N \in \mathbb{R}^N. \tag{1}$$

Here, $\mathbf{s}_u^t$ represents the difference between the values of the u-th column within the current test batch and the average values of the corresponding column in the source data. Using $\mathbf{s}_u^t$ for each column u, we define a shift trend graph where each node u represents a column, and each edge captures the relationship between different columns; the node feature for each node u is defined as $\mathbf{s}_u^t$, and the adjacency matrix is represented by an $D \times D$ all-ones matrix.

A graph neural network (GNN) is then applied to the graph formed above, enabling the exchange of shift trends between different columns through message passing. This process generates a column-wise contextualized representation, which is then averaged to produce an overall feature representation that encompasses all columns. Finally, the averaged node representation is concatenated with the initial prediction $f_\theta(\mathbf{x}_i^t)$ to yield the final output temperature T_i. This GNN-based uncertainty calibration not only captures shifts in individual columns but also sensitively detects correlation shifts occurring simultaneously across different columns, which are common in the tabular domain. A more detailed explanation of the architecture and training of the shift-aware uncertainty calibrator can be found in Section B.

3.3 Label Distribution Handler

This section introduces a label distribution handler designed to accurately estimate the target label distribution for the current test batch and adjust the model's output predictions accordingly. This approach is empirically justified by our observation that the marginal distribution of model predictions $p_t(y)$ in the target domain tends to be biased towards the source label distribution $p_s(y)$, as discussed in Section 2.2 and illustrated in Figure 2.

A straightforward solution to correct this bias is to simply multiply $p_t(y)/p_s(y)$ to align the marginal label distribution [3]. Specifically, given $p_t(y|\mathbf{x}_i^t) = \text{softmax}(f_\theta(\mathbf{x}_i^t))$, the adjusted prediction would be:

$$\text{norm}(p_t(y|\mathbf{x}_i^t)p_t(y)/p_s(y)) \tag{2}$$

where $\text{norm}(z)_i = z_i/\sum_{i'} z_{i'}$ normalizes the unnormalized probability. However, we find two major issues: 1) $p_t(y|\mathbf{x}_i^t)$ is often poorly calibrated and 2) overconfident yet incorrect predictions significantly hinder the accurate estimation of the target label distribution $p_t(y)$ (Section 2.2).

To tackle these challenges, we propose a simple yet effective estimator $\bar{p}_i(y|\mathbf{x}_i^t)$ defined like below:

$$\bar{p}_i(y|\mathbf{x}_i^t) = \frac{\tilde{p}_t(y|\mathbf{x}_i^t) + \text{norm}\big(\tilde{p}_t(y|\mathbf{x}_i^t)p_t(y)/p_s(y)\big)}{2}. \tag{3}$$

The key differences between the original Equation 2 and our Equation 3 are: 1) we use the calibrated prediction $\tilde{p}_t(y|\mathbf{x}_i^t)$ instead of the original prediction $p_t(y|\mathbf{x}_i^t)$ to enhance uncertainty quantification, and 2) we combine the calibrated estimate $\tilde{p}_t(y|\mathbf{x}_i^t)$ with the distributionally aligned prediction $\text{norm}(\tilde{p}_t(y|\mathbf{x}_i^t)p_t(y)/p_s(y))$ for more robust estimation.

Given the already-known source label distribution $p_s(y)$, we now explain the step-by-step process for estimating $\tilde{p}_t(y|\mathbf{x}_i^t)$ and $p_t(y)$. $p_t(y|\mathbf{x}_i^t)$ is calibrated into $\tilde{p}_t(y|\mathbf{x}_i^t)$ through a two-stage uncertainty calibration process. Specifically, for a current test batch $\{\mathbf{x}_i^t\}_{i=1}^N$, we calculate shift trend $\mathbf{s}^t$ using Equation 1 and get per-sample temperature $T_i = g_\phi(f_\theta(\mathbf{x}_i^t), \mathbf{s}^t)$ using shift-aware uncertainty calibrator g_ϕ to capture overall distribution shifts, as well as correlation and individual column shifts within the current batch. Here, we define the uncertainty δ_i of $f_\theta(\mathbf{x}_i^t)$ as a reciprocal of the margin of the calibrated probability distribution $\text{softmax}(f_\theta(\mathbf{x}_i^t)/T_i)$. We then measure the quantiles for each instance $\mathbf{x}_i$ using δ_i within the current batch and recalibrate the original probability with $\tilde{T}_i$, resulting in $\tilde{p}_t(y|\mathbf{x}_i^t) = \text{softmax}(f_\theta(\mathbf{x}_i^t)/\tilde{T}_i)$. This process calibrates predictions by leveraging relative uncertainty within the batch. Our temperature $\tilde{T}_i$ is defined as:

$$\tilde{T}_i = \begin{cases} T & \text{if } \delta_i \geq Q\big(\{\delta_{i'}\}_{i'=1}^N, q_{\text{high}}\big) \\ 1/T & \text{if } \delta_i \leq Q\big(\{\delta_{i'}\}_{i'=1}^N, q_{\text{low}}\big) \\ 1 & \text{otherwise,} \end{cases} \tag{4}$$

where $Q(X, q)$ is a quantile function which gives the value corresponding to the lower q quantile in X, $T = 1.5\rho/(\rho - 1 + 10^{-6})$ is a temperature with source class imbalance ratio $\rho = \max_j p_s(y)_j / \min_j p_s(y)_j$, and q_{low} and q_{high} represent the low and high uncertainty quantiles, respectively. This two-stage uncertainty calibration comprehensively evaluates the current batch and estimates relative uncertainty using $\mathbf{s}^t$, g_ϕ, and $\tilde{T}_i$.

Meanwhile, the target label distribution $p_t(y)$ is estimated as follows:

$$p_t(y) = (1-\alpha) \cdot \frac{1}{N}\sum_{i=1}^{N} p_t^{\text{de}}(y|\mathbf{x}_i^t) + \alpha \cdot p_t^{\text{oe}}(y), \tag{5}$$

where $p_t^{\text{de}}(y|\mathbf{x}_i^t) = \text{norm}\big(p_t(y|\mathbf{x}_i^t)/p_s(y)\big)$ is a debiased target label estimator that departs from $p_s(y)$, and $p_t^{oe}(y)$ is an online target label estimator, initialized as uniform distribution and updated as:

$$p_t^{\text{oe}}(y) = (1-\alpha) \cdot \frac{1}{N}\sum_{i=1}^{N} \bar{p}_t(y|\mathbf{x}_i^t) + \alpha \cdot p_t^{\text{oe}}(y) \tag{6}$$

from the current batch to the next, with a smoothing factor α. This online target label estimator leverages label locality between nearby test batches, making it effective for accurately estimating the next batch's target label distribution. A more detailed explanation of AdapTable is provided in Section B.

Table 2: The average balanced accuracy (%) and macro F1 score (%) with their standard errors for both supervised models and TTA baselines are reported across six datasets including natural distribution shifts within the TableShift [17] benchmark. The results are averaged over three random repetitions.

	HELOC		Voting		Hospital Readmission		ICU Mortality		Childhood Lead		Diabetes	
Method	bAcc.	F1	bAcc.	F1	bAcc.	F1	bAcc.	F1	bAcc.	F1	bAcc.	F1
k-NN	62.0 ± 0.0	40.3 ± 0.0	76.9 ± 0.0	71.1 ± 0.0	57.7 ± 0.0	56.9 ± 0.0	81.5 ± 0.3	47.6 ± 0.0	57.6 ± 0.1	56.9 ± 0.0	67.9 ± 0.3	53.3 ± 0.1
LogReg	63.5 ± 0.0	44.2 ± 0.0	80.2 ± 0.0	76.2 ± 0.0	61.4 ± 0.0	58.9 ± 0.0	61.6 ± 0.0	62.2 ± 0.0	50.0 ± 0.0	47.9 ± 0.0	71.0 ± 0.0	55.4 ± 0.0
RandomForest	58.2 ± 7.6	32.2 ± 1.5	$\mathbf{81.7 \pm 0.1}$	68.4 ± 0.7	64.4 ± 0.5	42.1 ± 1.2	$\mathbf{85.2 \pm 0.4}$	52.0 ± 0.1	50.0 ± 0.0	47.9 ± 0.0	$\mathbf{76.5 \pm 0.1}$	46.9 ± 0.1
XGBoost	57.6 ± 7.2	39.9 ± 4.9	80.5 ± 0.2	75.8 ± 0.4	63.1 ± 0.1	61.3 ± 0.4	79.9 ± 0.1	64.3 ± 0.1	50.0 ± 0.0	47.9 ± 0.0	71.5 ± 0.1	56.2 ± 0.1
CatBoost	65.4 ± 0.0	51.7 ± 0.0	80.4 ± 0.0	76.8 ± 0.0	63.4 ± 0.0	61.8 ± 0.5	81.4 ± 0.0	59.8 ± 0.0	50.0 ± 0.0	47.9 ± 0.0	65.0 ± 0.0	59.3 ± 0.0
+ AdapTable	$\mathbf{65.5 \pm 0.0}$	$\mathbf{65.4 \pm 0.0}$	79.6 ± 0.0	$\mathbf{78.6 \pm 0.0}$	$\mathbf{65.4 \pm 0.0}$	$\mathbf{62.5 \pm 0.3}$	82.6 ± 0.0	$\mathbf{64.8 \pm 0.3}$	$\mathbf{62.8 \pm 0.4}$	$\mathbf{61.7 \pm 0.3}$	74.2 ± 0.0	$\mathbf{62.5 \pm 0.3}$
Source	53.2 ± 1.5	38.2 ± 3.5	76.5 ± 0.5	77.3 ± 0.4	61.1 ± 0.1	60.2 ± 0.3	56.3 ± 0.0	58.1 ± 0.0	50.0 ± 0.0	47.9 ± 0.0	55.2 ± 0.0	55.5 ± 0.0
PL	51.8 ± 1.0	34.9 ± 2.3	75.6 ± 0.5	76.6 ± 0.5	60.5 ± 0.1	58.9 ± 0.3	56.3 ± 0.0	58.0 ± 0.1	50.0 ± 0.0	47.9 ± 0.0	55.1 ± 0.1	55.3 ± 0.0
TTT++	53.2 ± 1.5	38.2 ± 3.6	76.8 ± 0.5	77.6 ± 0.2	61.1 ± 0.1	60.2 ± 0.3	56.6 ± 0.5	58.5 ± 0.1	50.0 ± 0.0	47.9 ± 0.0	55.4 ± 0.0	55.7 ± 0.0
TENT	51.2 ± 1.2	33.2 ± 2.6	74.0 ± 0.6	74.9 ± 0.6	60.2 ± 0.1	58.3 ± 0.3	55.1 ± 0.1	56.3 ± 0.1	50.0 ± 0.0	47.9 ± 0.0	55.0 ± 0.0	55.0 ± 0.0
EATA	53.2 ± 1.5	38.2 ± 3.6	76.5 ± 0.5	77.3 ± 0.4	61.1 ± 0.1	60.2 ± 0.4	56.3 ± 0.0	58.1 ± 0.0	50.0 ± 0.0	47.9 ± 0.0	55.2 ± 0.0	55.5 ± 0.0
SAR	50.0 ± 0.0	30.1 ± 0.0	62.0 ± 1.2	59.4 ± 1.6	57.1 ± 1.1	51.3 ± 2.2	51.1 ± 0.1	49.1 ± 0.2	50.0 ± 0.0	47.9 ± 0.0	53.4 ± 0.0	52.2 ± 0.0
LAME	50.0 ± 0.0	30.1 ± 0.0	54.6 ± 0.5	46.8 ± 1.0	54.9 ± 0.5	46.9 ± 1.0	50.0 ± 0.0	46.7 ± 0.0	50.0 ± 0.0	47.9 ± 0.0	54.8 ± 0.1	54.8 ± 0.2
AdapTable	$\mathbf{65.8 \pm 0.6}$	$\mathbf{64.5 \pm 0.3}$	$\mathbf{78.4 \pm 0.3}$	$\mathbf{78.6 \pm 0.0}$	$\mathbf{61.7 \pm 0.0}$	$\mathbf{61.7 \pm 0.0}$	$\mathbf{65.9 \pm 0.1}$	$\mathbf{65.4 \pm 0.1}$	$\mathbf{69.2 \pm 0.1}$	$\mathbf{60.9 \pm 0.3}$	$\mathbf{70.9 \pm 0.1}$	$\mathbf{68.3 \pm 0.1}$

3.4 Theoretical Insights

Theorem 3.1. *Let $\hat{Y}|X$ and $\hat{Y}_o|X$ be defined as follows:*

$$\hat{Y}|X = \{\arg\max_{j\in\mathcal{Y}} f_\theta(\mathbf{x})_j | \mathbf{x} \in X\}, \tag{7}$$

$$\hat{Y}_o|X = \{\arg\max_{j\in\mathcal{Y}} f_\theta(\mathbf{x})_j + \log p_t^{oe}(y)_j | \mathbf{x} \in X\}. \tag{8}$$

Given the error $\epsilon(\hat{Y}|X) = \mathbb{P}(\hat{Y} \neq Y|X)$, with true labels Y of inputs X, the error gap $|\epsilon(\hat{Y}|X_s) - \epsilon(\hat{Y}_o|X_t)|$ is upper bounded by

$$K_1 \left\| 1 - \frac{p_t^{oe}(y)}{p_t(y)} \right\|_1 BSE(\hat{Y}) + K_2 \Delta_{CE}(\hat{Y}), \tag{9}$$

where K_1 and K_2 are constants related to $p_t(y)$, and $p_s(y)$, respectively.

Theorem 3.1 extends Theorem 2.3 in ODS [56] to cases where the source label distribution is not uniform. It decomposes the error gap between the original model on the source domain and the adapted model for the target model with $p_t^{oe}(y)$ on the target domain into several components. These components include $\|1 - p_t^{oe}(y)/p_t(y)\|_1$, which is an error of the estimated target label distribution, $BSE(\hat{Y})$, which reflects the model's performance on the source domain, and $\Delta_{CE}(\hat{Y})$, which measures the generalization of feature representations adapted by the TTA algorithm. Overall, Theorem 3.1 underscores the importance of tracking label distributions and efficiently adapting models to handle label distribution shifts. The detailed explanation and proof of Theorem 3.1 can be found in Section C.

4 Experiments

This section validates AdapTable's effectiveness. We begin with an overview of our experimental setup in Section 4.1 and then address key research questions: Is AdapTable effective across various tabular distribution shifts, including natural shifts and common corruptions across different tabular models? (Section 4.2), Do AdapTable's components contribute to overall performance improvements, and do they function as intended? (Section 4.3) Does AdapTable demonstrate strengths in computational efficiency and hyperparameter sensitivity, which are crucial for test time adaptation? (Section H.2)

4.1 Experimental Setup

Datasets. We evaluate AdapTable on six diverse datasets—HELOC, Voting, Hospital Readmission, ICU Mortality, Childhood Lead, and Diabetes—within the tabular distribution shift benchmark [17], covering healthcare, finance, and politics with both numerical and categorical features. Additionally, we verify its robustness against six common corruptions—Gaussian, Uniform, Random Drop, Column Drop, Numerical, and Categorical—to ensure its efficacy beyond label distribution shifts. More details of these shifts are in Section D.

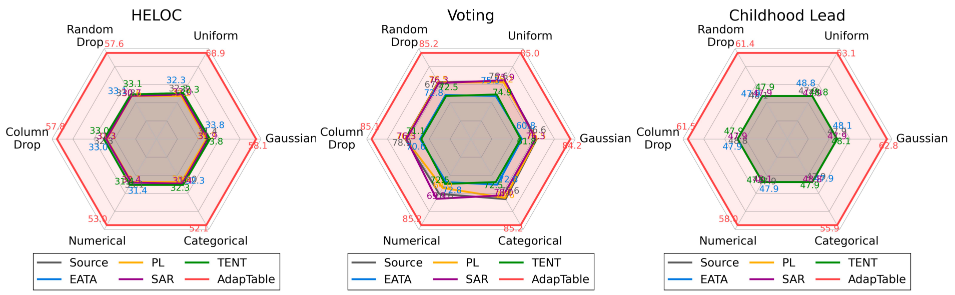

Figure 4: The average macro F1 score for AdapTable and TTA baselines is reported under six common corruptions using MLP across three datasets within the TableShift [17] benchmark.

Model architectures and baselines. To verify the proposed method under various tabular model architectures, we mainly use MLP, a widely used tabular learning architecture. Additionally, we validate AdapTable on CatBoost [12] and three other representative deep tabular learning models—AutoInt [47], ResNet [19], and FT-Transformer [19]. We compare AdapTable with six TTA baselines—PL [29], TTT++ [33], TENT [53], EATA [37], SAR [38], and LAME [6]. TabLog [42] is excluded due to its architectural constraint on logical neural networks [43]. We also provide performance references from classical machine learning models: k-nearest neighbors (k-NN), logistic regression (LogReg), random forest (RandomForest), XGBoost [8], and CatBoost [12].

Evaluation metrics and implementation details. As shown in Figure 2 and Section G, tabular data often exhibit extreme class imbalance. Since accuracy may not be effective in these cases, we use macro F1 score (F1) and balanced accuracy (bAcc.) as the primary evaluation metrics. For all experiments, we use a fixed batch size of 64, a common setting in TTA baselines [44, 53]. The smoothing factor α, low uncertainty quantile q_{low}, and high uncertainty quantile q_{high} are set to 0.1, 0.25, and 0.75, respectively. In all tables, we mark the **best** and <u>second-best</u> results.

4.2 Main Results

Result on natural distribution shifts. Table 2 presents results on natural distribution shifts. Existing TTA methods, successful in computer vision, struggle in the tabular domain, often failing to outperform the source model or offering limited performance gains. In contrast, AdapTable achieves state-of-the-art results across all datasets, with dramatic performance improvements of up to 26% on the HELOC dataset. Since AdapTable does not rely on model parameter tuning, it can be easily applied to classical machine learning models; when integrated with CatBoost, AdapTable consistently improves performance across all datasets, showcasing its versatility, whereas other baselines cannot be similarly integrated as they require model parameter updates.

Result on common corruptions. We further evaluate the efficacy of AdapTable across six types of common corruptions in real-world applications by applying them to the test sets of three datasets—HELOC, Voting, and Childhood Lead. As shown in Figure 4, prior TTA methods fail considerably, showing only marginal gains over the unadapted source model across all corruption types. It is worth noting that previous TTA methods have demonstrated significant improvements when dealing with common corruptions in vision data, highlighting the difference between corruptions in the tabular domain its counterpart in vision domain. Meanwhile, Adaptable shows substantial improvements across all types of corruptions, showing more than 10% gains of accuracy on all scenarios, demonstrating its robustness across different types of corruptions.

Table 3: Ablation study comparing the shift-aware uncertainty calibrator with classical methods—Platt scaling (PS) and isotonic regression (IR). The results are averaged over three random repetitions.

Method	HELOC	Voting	Hospital Readmission
Source	38.2 ± 3.5	<u>77.3 ± 0.4</u>	<u>60.2 ± 0.3</u>
PS	<u>61.6 ± 1.3</u>	73.3 ± 0.2	59.4 ± 0.3
IR	61.3 ± 1.7	74.3 ± 0.2	58.0 ± 0.4
AdapTable	**64.5 ± 0.6**	**78.6 ± 0.0**	**61.7 ± 0.0**

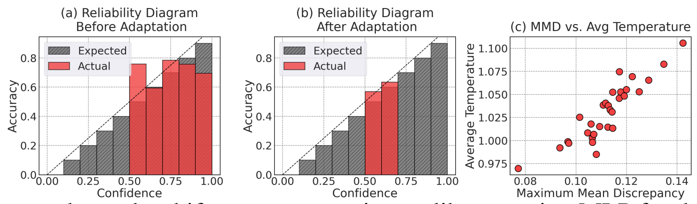

Figure 5: Ablation study on the shift-aware uncertainty calibrator using MLP for the HELOC dataset. (a) and (b) show reliability diagrams before and after calibration, while (c) depicts the average temperature relative to the maximum mean discrepancy (MMD) between the training set and the sampled test sets.

4.3 Ablation Study

Shift-aware uncertainty calibrator. We first validate the shift-aware uncertainty calibrator from Section 3.2. Figures 5(a) and (b) show reliability diagrams before and after calibration, demonstrating that our calibrator significantly reduces both overconfidence and underconfidence. Next, Figure 5(c) assesses shift-awareness by plotting the average temperature against the maximum mean discrepancy (MMD) with training data. As expected, greater shifts lead to higher temperatures, indicating increased uncertainty. The strong correlation between MMD and average temperature confirms the calibrator's effectiveness under distribution shifts. Finally, Table 3 compares our calibrator with classical methods like Platt scaling and isotonic regression. While classical methods show inconsistent performance across datasets, our shift-aware calibrator consistently outperforms them, effectively handling domain shifts during calibration.

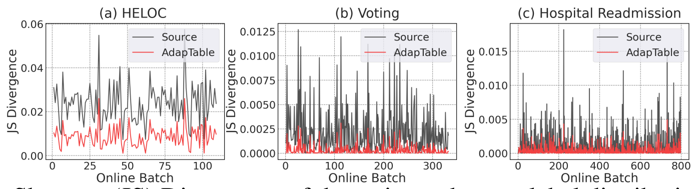

Figure 6: Jensen-Shannon (JS) Divergence of the estimated target label distribution before and after applying the label distribution handler using MLP on three datasets. The x-axis indicates the online batch index, and the y-axis shows the per-batch JS divergence from the ground truth labels.

Label distribution handler. We next validate the label distribution handler's efficacy by first comparing the Jensen–Shannon (JS) divergence between true and estimated label distributions across online batches in Figure 6. The results show that our handler significantly improves label distribution estimation accuracy, with low JS divergence across all datasets. We then assess its robustness under severe class imbalance (ratio of 10) and class-wise temporal correlation. As shown in Table 4, AdapTable achieves up to 27% and 19% performance improvements in the HELOC and Childhood Lead datasets, respectively. More experimental details are in Section D.

Table 4: The average macro F1 score (%) with standard errors for TTA baselines is reported using MLP across three datasets with 1) class imbalance and 2) temporal correlation from the TableShift benchmark. The results are averaged over three random repetitions.

	Class Imbalance			Temporal Correlation		
Method	HELOC	Voting	Childhood Lead	HELOC	Voting	Childhood Lead
Source	32.5 ± 3.5	52.3 ± 4.9	36.7 ± 6.5	31.6 ± 0.3	62.2 ± 0.1	35.1 ± 0.2
PL	32.0 ± 3.6	52.1 ± 4.9	36.7 ± 6.5	30.9 ± 0.2	54.9 ± 0.1	35.1 ± 0.2
TENT	32.5 ± 3.5	52.3 ± 4.9	36.7 ± 6.5	31.6 ± 0.3	55.7 ± 0.1	35.1 ± 0.2
EATA	32.5 ± 3.5	52.3 ± 4.9	36.7 ± 6.5	31.6 ± 0.3	55.7 ± 0.1	35.1 ± 0.2
SAR	31.8 ± 3.5	57.1 ± 5.3	36.7 ± 6.5	32.0 ± 0.2	54.4 ± 0.5	35.1 ± 0.2
LAME	29.9 ± 3.5	58.7 ± 4.0	36.7 ± 6.5	29.0 ± 0.1	38.0 ± 0.4	35.1 ± 0.2
AdapTable	**59.7 ± 0.8**	**62.0 ± 4.6**	**63.9 ± 1.0**	**56.1 ± 0.3**	**64.5 ± 0.0**	**64.8 ± 0.3**

5 Conclusion

In this paper, we have introduced AdapTable, a test-time adaptation framework tailored for tabular data. AdapTable overcomes the limitations of previous methods, which fail to address label distribution shifts, and lack versatility across architectures. Our approach, combined with a shift-aware uncertainty calibrator that enhances calibration via modeling column shifts, and a label distribution handler that adjusts the output distribution based on real-time estimates of the current batch's label distribution. Extensive experiments show that AdapTable achieves state-of-the-art performance across various datasets and architectures, effectively managing both natural distribution shifts and common corruptions.

References

[1] Sercan Ö Arik and Tomas Pfister. TabNet: Attentive interpretable tabular learning. In *AAAI Conference on Artificial Intelligence (AAAI)*, 2021.

[2] American Diabetes Association. Economic costs of diabetes in the us in 2017. *Diabetes care*, 2018.

[3] David Berthelot, Nicholas Carlini, Ekin D Cubuk, Alex Kurakin, Kihyuk Sohn, Han Zhang, and Colin Raffel. Remixmatch: Semi-supervised learning with distribution alignment and augmentation anchoring. In *International Conference on Learning Representations (ICLR)*, 2020.

[4] Ege Beyazit, Jonathan Kozaczuk, Bo Li, Vanessa Wallace, and Bilal Fadlallah. An inductive bias for tabular deep learning. *Conference in Neural Information Processing Systems (NeurIPS)*, 2024.

[5] Bernd Bischl, Giuseppe Casalicchio, Matthias Feurer, Frank Hutter, Michel Lang, Rafael G. Mantovani, Jan N. van Rijn, and Joaquin Vanschoren. Openml benchmarking suites. In *Conference on Neural Information Processing Systems (NeurIPS)*, 2021.

[6] Malik Boudiaf, Romain Mueller, Ismail Ben Ayed, and Luca Bertinetto. Parameter-free online test-time adaptation. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.

[7] Kyle Brown, Derek Doran, Ryan Kramer, and Brad Reynolds. HELOC applicant risk performance evaluation by topological hierarchical decomposition. In *NeurIPS Workshop on Challenges and Opportunities for AI in Financial Services*, 2018.

[8] Tianqi Chen and Carlos Guestrin. XGBoost: A scalable tree boosting system. In *ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD)*, 2016.

[9] John Clore, Krzysztof Cios, Jon DeShazo, and Beata Strack. Diabetes 130-US hospitals for years 1999-2008. UCI Machine Learning Repository, 2014.

[10] Ersilia M DeFilippis and Harriette GC Van Spall. Is it time for sex-specific guidelines for cardiovascular disease? *Journal of the American College of Cardiology (JACC)*, 2021.

[11] Tuan Dinh, Yuchen Zeng, Ruisu Zhang, Ziqian Lin, Michael Gira, Shashank Rajput, Jy-yong Sohn, Dimitris Papailiopoulos, and Kangwook Lee. Lift: Language-interfaced fine-tuning for non-language machine learning tasks. In *Conference on Neural Information Processing Systems (NeurIPS)*, 2022.

[12] Anna Veronika Dorogush, Vasily Ershov, and Andrey Gulin. CatBoost: gradient boosting with categorical features support. In *NeurIPS Workshop on ML Systems*, 2017.

[13] Xi Fang, Weijie Xu, Fiona Anting Tan, Jiani Zhang, Ziqing Hu, Yanjun Jane Qi, Scott Nickleach, Diego Socolinsky, Srinivasan Sengamedu, Christos Faloutsos, et al. Large language models (llms) on tabular data: Prediction, generation, and understanding-a survey. *Transactions on Machine Learning Research (TMLR)*, 2024.

[14] Matthias Fey and Jan Eric Lenssen. Fast graph representation learning with pytorch geometric. In *ICLR Workshop on Representation Learning on Graphs and Manifolds*, 2019.

[15] Centers for Disease Control, Prevention, et al. National health and nutrition examination survey (nhanes) data. *NCfHS, editor. NCHS*, 2003.

[16] Yossi Gandelsman, Yu Sun, Xinlei Chen, and Alexei Efros. Test-time training with masked autoencoders. In *Conference on Neural Information Processing Systems (NeurIPS)*, 2022.

[17] Josh Gardner, Zoran Popovic, and Ludwig Schmidt. Benchmarking distribution shift in tabular data with tableshift. In *Conference on Neural Information Processing Systems (NeurIPS)*, 2023.

[18] Taesik Gong, Jongheon Jeong, Taewon Kim, Yewon Kim, Jinwoo Shin, and Sung-Ju Lee. Note: Robust continual test-time adaptation against temporal correlation. In *Conference on Neural Information Processing Systems (NeurIPS)*, 2022.

[19] Yury Gorishniy, Ivan Rubachev, Valentin Khrulkov, and Artem Babenko. Revisiting deep learning models for tabular data. In *Conference on Neural Information Processing Systems (NeurIPS)*, 2021.

[20] Léo Grinsztajn, Edouard Oyallon, and Gaël Varoquaux. Why do tree-based models still outperform deep learning on tabular data? arxiv 2022. In *Conference on Neural Information Processing Systems (NeurIPS)*, 2022.

[21] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[22] Stefan Hegselmann, Alejandro Buendia, Hunter Lang, Monica Agrawal, Xiaoyi Jiang, and David Sontag. Tabllm: Few-shot classification of tabular data with large language models. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2023.

[23] Daniel Hein, Stefan Depeweg, Michel Tokic, Steffen Udluft, Alexander Hentschel, Thomas A. Runkler, and Volkmar Sterzing. A benchmark environment motivated by industrial control problems. In *IEEE Symposium Series on Computational Intelligence (SSCI)*, 2017.

[24] Noah Hollmann, Samuel Müller, Katharina Eggensperger, and Frank Hutter. TabPFN: A transformer that solves small tabular classification problems in a second. In *International Conference on Learning Representations (ICLR)*, 2023.

[25] Sehyun Hwang, Sohyun Lee, Sungyeon Kim, Jungseul Ok, and Suha Kwak. Combating label distribution shift for active domain adaptation. In *European Conference on Computer Vision (ECCV)*, 2022.

[26] Alistair Johnson, Lucas Bulgarelli, Tom Pollard, Steven Horng, Leo Anthony Celi, and Roger Mark. Mimic-iv, 2021.

[27] Alistair EW Johnson, Tom J Pollard, Lu Shen, Li-wei H Lehman, Mengling Feng, Mohammad Ghassemi, Benjamin Moody, Peter Szolovits, Leo Anthony Celi, and Roger G Mark. Mimic-iii, a freely accessible critical care database. *Scientific data*, 2016.

[28] Changhun Kim, Joonhyung Park, Hajin Shim, and Eunho Yang. SGEM: Test-time adaptation for automatic speech recognition via sequential-level generalized entropy minimization. In *Conference of the International Speech Communication Association (INTERSPEECH)*, 2023.

[29] Dong-Hyun Lee. Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks. In *ICML Workshop on Challenges in Representation Learning*, 2013.

[30] Jian Liang, Ran He, and Tieniu Tan. A comprehensive survey on test-time adaptation under distribution shifts. *International Journal of Computer Vision (IJCV)*, 2024.

[31] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, 2017.

[32] Jiashuo Liu, Tianyu Wang, Peng Cui, and Hongseok Namkoong. On the need for a language describing distribution shifts: Illustrations on tabular datasets. In *Conference on Neural Information Processing Systems (NeurIPS)*, 2023.

[33] Yuejiang Liu, Parth Kothari, Bastien Van Delft, Baptiste Bellot-Gurlet, Taylor Mordan, and Alexandre Alahi. Ttt++: When does self-supervised test-time training fail or thrive? In *Conference on Neural Information Processing Systems (NeurIPS)*, 2021.

[34] Aditya Krishna Menon, Sadeep Jayasumana, Ankit Singh Rawat, Himanshu Jain, Andreas Veit, and Sanjiv Kumar. Long-tail learning via logit adjustment. In *International Conference on Learning Representations (ICLR)*, 2021.

[35] Lori Mosca, Elizabeth Barrett-Connor, and Nanette Kass Wenger. Sex/gender differences in cardiovascular disease prevention: what a difference a decade makes. *Circulation*, 2011.

[36] Fionn Murtagh. Multilayer perceptrons for classification and regression. *Neurocomputing*, 1991.

[37] Shuaicheng Niu, Jiaxiang Wu, Yifan Zhang, Yaofo Chen, Shijian Zheng, Peilin Zhao, and Mingkui Tan. Efficient test-time model adaptation without forgetting. In *International Conference on Machine Learning (ICML)*, 2022.

[38] Shuaicheng Niu, Jiaxiang Wu, Yifan Zhang, Zhiquan Wen, Yaofo Chen, Peilin Zhao, and Mingkui Tan. Towards stable test-time adaptation in dynamic wild world. In *International Conference on Learning Representations (ICLR)*, 2023.

[39] Joonhyung Park, Hyunjin Seo, and Eunho Yang. Pc-adapter: Topology-aware adapter for efficient domain adaption on point clouds with rectified pseudo-label. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, 2023.

[40] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. In *Conference on neural information processing systems (NeurIPS)*, 2019.

[41] John Platt. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. In *Advances in Large Margin Classifiers*, 2000.

[42] Weijieying Ren, Xiaoting Li, Huiyuan Chen, Vineeth Rakesh, Zhuoyi Wang, Mahashweta Das, and Vasant G Honavar. Tablog: Test-time adaptation for tabular data using logic rules. In *International Conference on Machine Learning (ICML)*, 2024.

[43] Ryan Riegel, Alexander Gray, Francois Luus, Naweed Khan, Ndivhuwo Makondo, Ismail Yunus Akhalwaya, Haifeng Qian, Ronald Fagin, Francisco Barahona, Udit Sharma, et al. Logical neural networks. *arXiv preprint arXiv:2006.13155*, 2020.

[44] Steffen Schneider, Evgenia Rusak, Luisa Eck, Oliver Bringmann, Wieland Brendel, and Matthias Bethge. Improving robustness against common corruptions by covariate shift adaptation. In *Conference on Neural Information Processing Systems (NeurIPS)*, 2020.

[45] Wenqi Shi, Ran Xu, Yuchen Zhuang, Yue Yu, Hang Wu, Carl Yang, and May D Wang. Medadapter: Efficient test-time adaptation of large language models towards medical reasoning. *arXiv preprint arXiv:2405.03000*, 2024.

[46] Hajin Shim, Changhun Kim, and Eunho Yang. Cloudfixer: Test-time adaptation for 3d point clouds via diffusion-guided geometric transformation. In *European Conference on Computer Vision (ECCV)*, 2024.

[47] Weiping Song, Chence Shi, Zhiping Xiao, Zhijian Duan, Yewen Xu, Ming Zhang, and Jian Tang. Autoint: Automatic feature interaction learning via self-attentive neural networks. In *ACM International Conference on Information and Knowledge Management (CIKM)*, 2019.

[48] American National Election Studies. Fico. the explainable machine learning challenge., 2019. URL `https://community.fico.com/s/explainable-machine-learning-challenge`.

[49] American National Election Studies. Anes time series cumulative data file [dataset and documentation]. september 16, 2022 version, 2022. URL `www.electionstudies.org`.

[50] Mario Stylianou and Nancy Flournoy. Dose finding using the biased coin up-and-down design and isotonic regression. In *Biometrics*, 2002.

[51] Yu Sun, Xiaolong Wang, Zhuang Liu, John Miller, Alexei Efros, and Moritz Hardt. Test-time training with self-supervision for generalization under distribution shifts. In *International Conference on Machine Learning (ICML)*, 2020.

[52] Remi Tachet des Combes, Han Zhao, Yu-Xiang Wang, and Geoffrey J Gordon. Domain adaptation with conditional distribution matching and generalized label shift. In *Conference on Neural Information Processing Systems (NeurIPS)*, 2020.

[53] Dequan Wang, Evan Shelhamer, Shaoteng Liu, Bruno Olshausen, and Trevor Darrell. Tent: Fully test-time adaptation by entropy minimization. In *International Conference on Learning Representations (ICLR)*, 2021.

[54] Xiao Wang, Hongrui Liu, Chuan Shi, and Cheng Yang. Be confident! towards trustworthy graph neural networks via confidence calibration. In *Conference on Neural Information Processing Systems (NeurIPS)*, 2021.

[55] Ruihan Wu, Chuan Guo, Yi Su, and Kilian Q Weinberger. Online adaptation to label distribution shift. In *Conference on Neural Information Processing Systems (NeurIPS)*, 2021.

[56] Zhi Zhou, Lan-Zhe Guo, Lin-Han Jia, Dingchu Zhang, and Yu-Feng Li. ODS: Test-time adaptation in the presence of open-world data shift. In *International Conference on Machine Learning (ICML)*, 2023.

Appendix

A Related Work

Machine learning for tabular data. The distinct nature of tabular data reduces the effectiveness of deep neural networks, making gradient-boosted decision trees [8, 12] more suitable. However, research continues to develop deep learning models tailored for tabular data [36, 47, 1, 19], including recent efforts involving large language models [13, 22, 24, 11] that leverage textual prior knowledge. Notably, our method is architecture-agnostic and can be applied to any model.

Distribution shifts in the tabular domain. Recently, distribution shift benchmarks for tabular data have been introduced [32, 17]. WhyShift [32] reveals that concept shifts ($Y|X$-shifts) are more prevalent and detrimental than covariate shifts (X-shifts). TableShift [17] offers a benchmark with 15 classification tasks, highlighting a strong correlation between shift gaps and label distribution shifts (Y-shifts), which supports the validity of our method.

Test-time adaptation. Over the past years, test-time adaptation (TTA) methods have been proposed across various domains, such as computer vision [53, 18, 38, 46], natural language processing [45, 30], and speech processing [28]. These methods adapt pre-trained models to unlabeled target domains without requiring access to source data, making them well-suited for sensitive tabular data. TabLog [42] is a recent TTA method specifically for tabular data, but it has architectural constraints and lacks a comprehensive analysis of distribution shifts. This underscores the need for model-agnostic TTA methods with a deeper understanding of tabular data, which we address in this paper.

B Detailed Algorithm of AdapTable

Post-training shift-aware uncertainty calibrator. Given a pre-trained tabular classifier $f_\theta : \mathbb{R}^D \to \mathbb{R}^C$ on the source domain $\mathcal{D}_s = \{(\mathbf{x}_i^s, y_i^s)\}_i$, we introduce a post-training phase for a shift-aware uncertainty calibrator $g_\phi : \mathbb{R}^C \times \mathbb{R}^{D\times N} \to \mathbb{R}^+$. This calibrator is trained after the initial training of f_θ using the same training dataset $\mathcal{D}_s$. For a given training batch $\{(\mathbf{x}_i^s, y_i^s)\}_{i=1}^N$, we compute the shift trend $\mathbf{s}^s = (\mathbf{s}_u^s)_{u=1}^D$ for a specific column index u as follows:

$$\mathbf{s}_u^s = \big(\mathbf{x}_{iu}^s - \frac{1}{|\mathcal{D}_s|}\sum_{i'=1}^{|\mathcal{D}_s|} \mathbf{x}_{i'u}^s\big)_{i=1}^N,$$

where we add a linear layer to $\mathbf{s}_u^s$ for categorical column u to transform it into a one-dimensional representation, ensuring alignment with the numerical columns. Using $\mathbf{s}^s$, we construct a shift trend graph, where each node u represents a column, and edges capture the relationships between columns. The node features are given by $\mathbf{s}_u^t$, and the graph is connected using an all-ones adjacency matrix. A graph neural network (GNN) is applied to this graph, facilitating the exchange of shift trends between columns through message passing, which generates a contextualized column-wise representation $\boldsymbol{h}_u^s$. These representations are averaged to form a global feature representation $\boldsymbol{h}^s = \frac{1}{D}\sum_{u=1}^D \boldsymbol{h}_u^s$, which is then concatenated with the initial model prediction $f_\theta(\mathbf{x}_i^s)$ to produce the final output temperature T_i. With the calibrated probability $p_i = \text{softmax}\big(f_\theta(\mathbf{x}_i^s)/T_i\big)$, with the per-sample temperature T_i calculated above, we define the most plausible and second plausible class indices j^* and j^{**} as follows:

$$j^* = \arg\max_{j\in\mathcal{Y}} p_{ij} \quad \text{and} \quad j^{**} = \arg\max_{j\in\mathcal{Y}, j\neq j^*} p_{ij}.$$

The focal loss $\mathcal{L}_{\text{FL}}$ [31] and the calibration loss $\mathcal{L}_{\text{CAL}}$ [54] are used to train the shift-aware uncertainty calibrator g_ϕ, defined as:

$$\mathcal{L}_{\text{FL}}(\mathbf{x}_i^s, y_i^s) = \sum_{j=1}^C \mathbb{1}_{\{y_i^s\}}(j)(1-p_{ij})^\gamma \log p_{ij}, \tag{10}$$

$$\mathcal{L}_{\text{CAL}}(\mathbf{x}_i^s, y_i^s) = \mathbb{1}_{\{y_i^s\}}(j^*)(1-p_{ij^*}+p_{ij^{**}}) + \mathbb{1}_{\mathcal{Y}\setminus\{y_i^s\}}(j^*)(p_{ij^*}-p_{ij^{**}}), \tag{11}$$

Algorithm 1 AdapTable

```
1: Input: Pre-trained classifier $f_\theta(\cdot)$, post-trained shift-aware uncertainty calibrator $g_\phi(\cdot,\cdot)$, indicator function $\mathbb{1}_{(\cdot)}(\cdot)$, quantile function $Q(\cdot,\cdot)$, softmax function softmax$(\cdot)$, normalization function norm$(\cdot)$, source data $\mathcal{D}_s = \{(\mathbf{x}_i^s, y_i^s)\}_i$, current test batch $\{\mathbf{x}_i^t\}_{i=1}^N$, source class imbalance ratio $\rho = \max_j p_s(y)_j / \min_j p_s(y)_j$
2: Parameters: Smoothing factor $\alpha$, low uncertainty quantile $q_{\text{low}}$, high uncertainty quantile $q_{\text{high}}$
3: $p_s(y),\ T \leftarrow \big(\frac{1}{|\mathcal{D}_s|}\sum_{i=1}^{|\mathcal{D}_s|}\mathbb{1}_{\{j\}}(y_i^s)\big)_{j=1}^{C},\ 1.5\rho/(\rho-1+10^{-6})$
4: for $u = 1$ to $D$ do
5:     $\mathbf{s}_u^t \leftarrow \big(\mathbf{x}_{iu}^t - \frac{1}{|\mathcal{D}_s|}\sum_{i'=1}^{|\mathcal{D}_s|}\mathbf{x}_{i'u}^s\big)_{i=1}^N$        ▷ Compute shift trend $\mathbf{s}^t$
6: end for
7: for $i = 1$ to $N$ do
8:     $p_t(y|\mathbf{x}_i^t) \leftarrow \text{softmax}\big(f_\theta(\mathbf{x}_i^t)\big)$
9:     $T_i \leftarrow g_\phi\big(f_\theta(\mathbf{x}_i^t), \mathbf{s}^t\big)$        ▷ Determine per-sample temperature $\mathbf{x}_i^t$
10:    $j^*,\ j^{**} \leftarrow \arg\max_{1\le j\le C} p_t(y|\mathbf{x}_i^t)_j,\ \arg\max_{1\le j\le C, j\ne j^*} p_t(y|\mathbf{x}_i^t)_j$
11:    $\delta_i \leftarrow \big(\text{softmax}\big(f_\theta(\mathbf{x}_i^t)/T_i\big)_{j^*} - \text{softmax}\big(f_\theta(\mathbf{x}_i^t)/T_i\big)_{j^{**}}\big)^{-1}$ ▷ Define uncertainty of $\mathbf{x}_i^t$ as a margin of $f_\theta(\mathbf{x}_i^t)/T_i$
12:    $p_t^{\text{de}}(y|\mathbf{x}_i^t) \leftarrow \text{norm}\big(p_t(y|\mathbf{x}_i^t)/p_s(y)\big)$        ▷ Compute debiased target label estimator
13: end for
14: $p_t(y) \leftarrow (1-\alpha)\cdot\frac{1}{N}\sum_{i=1}^N p_t^{\text{de}}(y|\mathbf{x}_i^t) + \alpha\cdot p_t^{\text{oe}}(y)$        ▷ Estimate target label distribution
15: for $i = 1$ to $N$ do
16:    if $\delta_i \ge Q\big(\{\delta_{i'}\}_{i'=1}^N, q_{\text{high}}\big)$ then
17:        $\tilde{T}_i \leftarrow T$
18:    else if $\delta_i \le Q\big(\{\delta_{i'}\}_{i'=1}^N, q_{\text{low}}\big)$ then
19:        $\tilde{T}_i \leftarrow 1/T$        ▷ Calculate temperature $\tilde{T}_i$ using uncertainty $\delta_i$
20:    else
21:        $\tilde{T}_i \leftarrow 1$
22:    end if
23:    $\tilde{p}_t(y|\mathbf{x}_i^t) \leftarrow \text{softmax}\big(f_\theta(\mathbf{x}_i^t)/\tilde{T}_i\big)$        ▷ Perform temperature scaling with $\tilde{T}_i$
24:    $\bar{p}_t(y|\mathbf{x}_i^t) \leftarrow \big(\tilde{p}_t(y|\mathbf{x}_i^t) + \text{norm}(\tilde{p}_t(y|\mathbf{x}_i^t)p_t(y)/p_s(y))\big)/2$        ▷ Perform self-ensembling
25: end for
26: $p_t^{\text{oe}}(y) \leftarrow (1-\alpha)\cdot\frac{1}{N}\sum_{i=1}^N \bar{p}_t(y|\mathbf{x}_i^t) + \alpha\cdot p_t^{\text{oe}}(y)$        ▷ Update online target label estimator
27: Output: Final predictions $\{\bar{p}_i(y)\}_{i=1}^N$
```

where $\mathbb{1}_{\boldsymbol{A}}(x)$ is an indicator function:

$$\mathbb{1}_{\boldsymbol{A}}(x) = \begin{cases} 1 & \text{if } x \in \boldsymbol{A} \\ 0 & \text{otherwise.} \end{cases}$$

$\mathcal{L}_{\text{FL}}$ addresses class imbalance by reducing the impact of easily classified examples, while $\mathcal{L}_{\text{CAL}}$ penalizes the gap between p_{ij^*} and $p_{ij^{**}}$ for correct predictions, encouraging them to converge for incorrect predictions. For all experiments, we set $\gamma = 2$ and $\lambda_{\text{CAL}} = 0.1$.

Label distribution handler. During the test phase after post-training g_ϕ, we introduce a label distribution handler using an estimator $\bar{p}_i(y|\mathbf{x}_i^t)$, defined as:

$$\bar{p}_i(y|\mathbf{x}_i^t) = \frac{\tilde{p}_t(y|\mathbf{x}_i^t) + \text{norm}\big(\tilde{p}_t(y|\mathbf{x}_i^t)p_t(y)/p_s(y)\big)}{2},$$

where $\tilde{p}_t(y|\mathbf{x}_i^t)$ represents the calibrated prediction. This approach enhances uncertainty quantification and combines the calibrated estimation with the distributionally aligned prediction for more robust estimation. To compute $\tilde{p}_t(y|\mathbf{x}_i^t)$, we perform a two-stage uncertainty calibration. Specifically, for a given test batch $\{\mathbf{x}_i^t\}_{i=1}^N$, we calculate the shift trend $\mathbf{s}^t = (\mathbf{s}_u^t)_{u=1}^D \in \mathbb{R}^{D\times N}$ as:

$$\mathbf{s}_u^t = \big(\mathbf{x}_{iu}^t - \frac{1}{|\mathcal{D}_s|}\sum_{i'=1}^{|\mathcal{D}_s|}\mathbf{x}_{i'u}^s\big)_{i=1}^N \in \mathbb{R}^N.$$

Then, a per-sample temperature $T_i = g_\phi(f_\theta(\mathbf{x}_i^t), \mathbf{s}^t)$, which was defined in Equation 1 is computed. The uncertainty δ_i of $f_\theta(\mathbf{x}_i^t)$ is defined as the reciprocal of the margin of the calibrated probability distribution softmax$(f_\theta(\mathbf{x}_i^t)/T_i)$:

$$\delta_i = \frac{1}{\text{softmax}\big(f_\theta(\mathbf{x}_i^t)/T_i\big)_{j^*} - \text{softmax}\big(f_\theta(\mathbf{x}_i^t)/T_i\big)_{j^{**}}},$$

where j^* and j^{**} are the most plausible and second plausible class indices:

$$j^* = \arg\max_{j \in \mathcal{Y}} f_\theta(\mathbf{x}_i^t)_j \quad \text{and} \quad j^{**} = \arg\max_{j \in \mathcal{Y}, j \neq j^*} f_\theta(\mathbf{x}_i^t)_j.$$

Based on δ_i, the recalibrated temperature $\tilde{T}_i$is applied:

$$\tilde{T}_i = \begin{cases} T & \text{if } \delta_i \geq Q\big(\{\delta_{i'}\}_{i'=1}^N, q_{\text{high}}\big) \\ 1/T & \text{if } \delta_i \leq Q\big(\{\delta_{i'}\}_{i'=1}^N, q_{\text{low}}\big) \\ 1 & \text{otherwise,} \end{cases}$$

where $T = 1.5\rho/(\rho - 1 + 10^{-6})$ with $\rho = \max_j p_s(y)_j / \min_j p_s(y)_j$, and q_{low} and q_{high} are the low and high uncertainty quantiles, respectively. The target label distribution $p_t(y)$ is then estimated using the following formula:

$$p_t(y) = (1 - \alpha) \cdot \frac{1}{N} \sum_{i=1}^N p_t^{\text{de}}(y|\mathbf{x}_i^t) + \alpha \cdot p_t^{\text{oe}}(y),$$

where $p_t^{\text{de}}(y|\mathbf{x}_i^t) = \text{norm}\big(p_t(y|\mathbf{x}_i^t)/p_s(y)\big)$ serves as a debiased target label estimator, deviating from the source label distribution $p_s(y)$. The online target label estimator $p_t^{\text{oe}}(y)$ is initialized with a uniform distribution and updated with each new batch as follows:

$$p_t^{\text{oe}}(y) = (1 - \alpha) \cdot \frac{1}{N} \sum_{i=1}^N \bar{p}_t(y|\mathbf{x}_i^t) + \alpha \cdot p_t^{\text{oe}}(y),$$

where α is a smoothing factor. This update process leverages information from the current batch to refine the target label distribution estimation over time. The overall procedure of the proposed AdapTable method is summarized in Algorithm 1.

C Proof of Theorem 3.1

Let's first define the balanced source error $BSE(\hat{Y})$ on the source dataset and the conditional error gap $\Delta_{CE}(\hat{Y})$ between $\mathbb{P}(\hat{Y} \neq Y|X_s)$ and $\mathbb{P}(\hat{Y} \neq Y|X_t)$ as follows:

$$BSE(\hat{Y}) = \max_{i \in \mathcal{Y}} \mathbb{P}(\hat{Y} \neq i | Y = i, X_s), \tag{12}$$

$$\Delta_{CE}(\hat{Y}) = \max_{i \neq i' \in \mathcal{Y}} \Big| \mathbb{P}(\hat{Y} = i | Y = i', X_s) - \mathbb{P}(\hat{Y} = i | Y = i', X_t) \Big|. \tag{13}$$

Definition C.1. *(Generalized Label Shift in Tachet des Combes et al. [52]). Both input covariate distribution $\mathbb{P}(X_s) \neq \mathbb{P}(X_t)$ and output label distribution $\mathbb{P}(Y|X_s) \neq \mathbb{P}(Y|X_t)$ change. Yet, there exists a hidden representation $H = g^*(X)$ such that the conditional distribution of H given Y remains the same across both domains, i.e., $\forall i \in \mathcal{Y}$,*

$$\mathbb{P}(H|Y = i, X_s) = \mathbb{P}(H|Y = i, X_t). \tag{14}$$

Proof. We start by applying the law of total probability and triangle inequality to derive the following inequality:

$$\begin{aligned}
&\left|\epsilon(\hat{Y}|X_s) - \epsilon(\hat{Y}_o|X_t)\right| \\
&= \left|\mathbb{P}(\hat{Y} \neq Y|X_s) - \mathbb{P}(\hat{Y}_o \neq Y|X_t)\right| \\
&= \Big|\sum_{i\neq i'} \mathbb{P}(\hat{Y} = i, Y = i'|X_s) - \sum_{i\neq i'} \mathbb{P}(\hat{Y}_o = i, Y = i'|X_t)\Big| \\
&= \Big|\sum_{i\neq i'} \mathbb{P}(Y = i'|X_s)\mathbb{P}(\hat{Y} = i|Y = i', X_s) - \sum_{i\neq i'} \mathbb{P}(Y = i'|X_t)\mathbb{P}(\hat{Y}_o = i|Y = i', X_t)\Big| \\
&\leq \sum_{i\neq i'} \Big|\mathbb{P}(Y = i'|X_s)\mathbb{P}(\hat{Y} = i|Y = i', X_s) - \mathbb{P}(Y = i'|X_t)\mathbb{P}(\hat{Y}_o = i|Y = i', X_t)\Big|.
\end{aligned} \tag{15}$$

According to Equation 8 in [34], $\hat{Y}_o$ satisfies the following condition under generalized label shift condition in Definition C.1:

$$\mathbb{P}(\hat{Y}_o = i|H, X_t) = \frac{p_t^{oe}(y)_i}{\mathbb{P}(Y = i|X_s)}\mathbb{P}(\hat{Y} = i|H, X_t). \tag{16}$$

By multiplying both sides of Equation 16 by $\mathbb{P}(H|Y, X_t)$, we obtain:

$$\begin{aligned}
\mathbb{P}(\hat{Y}_o = i|H, X_t)\mathbb{P}(H|Y, X_t) &= \frac{p_t^{oe}(y)_i}{\mathbb{P}(Y = i|X_s)}\mathbb{P}(\hat{Y} = i|H, X_t)\mathbb{P}(H|Y, X_t) \\
\mathbb{P}(\hat{Y}_o = i|Y, X_t) &= \frac{p_t^{oe}(y)_i}{\mathbb{P}(Y = i|X_s)}\mathbb{P}(\hat{Y} = i|Y, X_t).
\end{aligned} \tag{17}$$

Next, by substituting Equation 17 into Equation 15, and letting $Y = i'$, we have:

$$\begin{aligned}
&\left|\epsilon(\hat{Y}|X_s) - \epsilon(\hat{Y}_o|X_t)\right| \\
&\leq \sum_{i\neq i'} \left|\mathbb{P}(Y = i'|X_s)\mathbb{P}(\hat{Y} = i|Y = i', X_s) - \mathbb{P}(Y = i'|X_t)\frac{p_t^{oe}(y)_i}{\mathbb{P}(Y = i|X_s)}\mathbb{P}(\hat{Y} = i|Y = i', X_t)\right|.
\end{aligned} \tag{18}$$

Using Lemma A.2 from [52], we can further estimate the upper bound of Equation 18 as follows:

$$\begin{aligned}
&\left|\epsilon(\hat{Y}|X_s) - \epsilon(\hat{Y}_o|X_t)\right| \\
&\leq \sum_{i\neq i'} \mathbb{P}(Y = i'|X_t)\left|1 - \frac{p_t^{oe}(y)_i}{\mathbb{P}(Y = i|X_s)}\right|\Big(\alpha_{i'}\mathbb{P}(\hat{Y} = i|Y = i', X_s) + \beta_{i'}\mathbb{P}(\hat{Y} = i|Y = i', X_t)\Big) \\
&\quad + \mathbb{P}(Y = i'|X_s)\Delta_{CE}(\hat{Y}) + \mathbb{P}(Y = i'|X_t)\frac{p_t^{oe}(y)_i}{\mathbb{P}(Y = i|X_s)}\Delta_{CE}(\hat{Y}) \\
&\overset{(i)}{\leq} \sum_{i\neq i'} \mathbb{P}(Y = i'|X_t)\left|1 - \frac{p_t^{oe}(y)_i}{\mathbb{P}(Y = i|X_s)}\right|\Big(\alpha_{i'}\mathbb{P}(\hat{Y} = i|Y = i', X_s) + \beta_{i'}\mathbb{P}(\hat{Y} = i|Y = i', X_t)\Big) \\
&\quad + (C-1)\Delta_{CE}(\hat{Y}) + \left(\sum_{i\neq i'} \frac{\mathbb{P}(Y = i'|X_t)}{\mathbb{P}(Y = i|X_s)}\right)\left(\sum_{i\neq i'} p_t^{oe}(y)_i\right)\Delta_{CE}(\hat{Y}) \\
&\leq \sum_{i\neq i'} \mathbb{P}(Y = i'|X_t)\left|1 - \frac{p_t^{oe}(y)_i}{\mathbb{P}(Y = i|X_s)}\right|\Big(\alpha_{i'}\mathbb{P}(\hat{Y} = i|Y = i', X_s) + \beta_{i'}\mathbb{P}(\hat{Y} = i|Y = i', X_t)\Big) \\
&\quad + (C-1)\Delta_{CE}(\hat{Y}) + \frac{(C-1)^2}{\min_{i\in\mathcal{Y}} \mathbb{P}(Y = i|X_s)}\Delta_{CE}(\hat{Y}),
\end{aligned} \tag{19}$$

where $\alpha_{i'}, \beta_{i'} \geq 0$ and $\alpha_{i'} + \beta_{i'} = 1$, (i) holds by Hölder's inequality. By letting $\alpha_{i'} = 1$ and $\beta_{i'} = 0$ for all $i' \in \mathcal{Y}$, and defining K_1 and K_2 as:

$$K_1 = C(C-1)^2 \max_{i \in \mathcal{Y}} \mathbb{P}(Y = i | X_t),$$
$$K_2 = (C-1) + \frac{(C-1)^2}{\min_{i \in \mathcal{Y}} \mathbb{P}(Y = i | X_s)},$$

we finally get:

$$\begin{aligned}
&\left|\epsilon(\hat{Y}|X_s) - \epsilon(\hat{Y}_o|X_t)\right| \\
&\leq \sum_{i \neq i'} \mathbb{P}(Y = i'|X_t) \left|1 - \frac{p_t^{oe}(y)_i}{\mathbb{P}(Y = i|X_s)}\right| \mathbb{P}(\hat{Y} = i | Y = i', X_s) + K_2 \Delta_{CE}(\hat{Y}) \\
&\leq \max_{i' \in \mathcal{Y}} \mathbb{P}(Y = i'|X_t) \sum_{i \neq i'} \left|1 - \frac{p_t^{oe}(y)_i}{\mathbb{P}(Y = i|X_s)}\right| \mathbb{P}(\hat{Y} = i | Y = i', X_s) + K_2 \Delta_{CE}(\hat{Y}) \\
&\overset{(i)}{\leq} \max_{i' \in \mathcal{Y}} \mathbb{P}(Y = i'|X_t) \left(\sum_{i \neq i'} \left|1 - \frac{p_t^{oe}(y)_i}{\mathbb{P}(Y = i|X_s)}\right|\right) \left(\sum_{i \neq i'} \mathbb{P}(\hat{Y} = i | Y = i', X_s)\right) + K_2 \Delta_{CE}(\hat{Y}) \\
&\overset{(ii)}{\leq} \max_{i' \in \mathcal{Y}} \mathbb{P}(Y = i'|X_t)(C-1) \sum_{i=1}^{C} \left|1 - \frac{p_t^{oe}(y)_i}{\mathbb{P}(Y = i|X_s)}\right| C(C-1) BSE(\hat{Y}) + K_2 \Delta_{CE}(\hat{Y}) \\
&= \max_{i' \in \mathcal{Y}} \mathbb{P}(Y = i'|X_t) C(C-1)^2 \left\|1 - \frac{p_t^{oe}(y)}{p_t(y)}\right\|_1 BSE(\hat{Y}) + K_2 \Delta_{CE}(\hat{Y}) \\
&\overset{(iii)}{=} K_1 \left\|1 - \frac{p_t^{oe}(y)}{p_t(y)}\right\|_1 BSE(\hat{Y}) + K_2 \Delta_{CE}(\hat{Y}),
\end{aligned} \tag{20}$$

where (i) holds by Hölder's inequality, (ii) holds by the definition of $BSE(\hat{Y})$, and (iii) holds by the definition of K_1. □

We observe that in practice, using $\hat{Y}_o|X = \{\arg\max_{j \in \mathcal{Y}} f_\theta(\mathbf{x})_j + \log p_t^{oe}(y)_j | \mathbf{x} \in X\}$ can result in performance degradation due to an error accumulation in $p_t^{oe}(y)$. However, our approach, which integrates a two-stage uncertainty calibration with g_ϕ and a debiased target label estimator $p_t^{de}(y)$, demonstrates empirical efficacy across various experiments.

D Dataset Descriptions

D.1 Natural Distibution Shifts

In our experiments, we verify our method across six different datasets—HELOC, Voting, Hospital Readmission, ICU Mortality, Childhood Lead, and Diabetes—within the Tableshift Benchmark [17], all of which include natural distribution shifts between training and test data. For all datasets, the numerical features are normalized—subtraction of mean and division by standard deviation, while categorical features are one-hot encoded. We find that different encoding types do not play a significant role in terms of accuracy, as noted in Grinsztajn et al. [20]. Detailed statistics specifications of each dataset are listed in Table 5.

- **HELOC:** This task predicts Home Equity Line of Credit (HELOC) [7] repayment using FICO data [48], focusing on shifts in third-party risk estimates. The dataset includes 10,459 observations, and a distribution shift occurs by using the 'External Risk Estimate' as a domain split. Estimates above 63 are used for training, while those 63 or below are held out for testing, illustrating potential biases in credit assessments.
- **Voting:** Using ANES [49] data, this task predicts U.S. presidential election voting behavior with 8,280 observations. Distribution shift is introduced by splitting the data based on geographic region, with the southern U.S. serving as the out-of-domain region. This simulates

how voter behavior predictions might vary when polling data is collected in one region and used to predict outcomes in another.

- **Hospital Readmission:** Hospital Readmission [9] predicts 30-day readmission of diabetic patients using data from 130 U.S. hospitals over 10 years. The distribution shift occurs by splitting the data based on admission source, with emergency room admissions held out as the target domain. This tests how well models trained on other sources perform when applied to patients admitted through the emergency room.
- **ICU Mortality:** The task predicts ICU patient mortality using MIMIC-iii data [27], focusing on shifts related to insurance type. The dataset includes 23,944 observations, and a distribution shift is created by excluding Medicare and Medicaid patients from the training set, designating them as the target domain. This highlights how insurance type can affect mortality predictions.
- **Childhood Lead:** This task predicts elevated blood lead levels in children using NHANES data [15], with 27,499 observations. A distribution shift is introduced by splitting the data based on poverty using the poverty-income ratio (PIR) as a threshold. Those with a PIR of 1.3 or lower are held out for testing, simulating risk assessment in lower-income households.
- **Diabetes:** This task predicts diabetes using BRFSS data [2], focusing on racial shifts across 1.4 million observations. Distribution shift occurs by focusing on the differences in diabetes risk between racial and ethnic groups, particularly highlighting the higher risk faced by non-white groups compared to White non-Hispanic individuals.

Table 5: Summary of the datasets used in our experiments, including the total number of instances (Total Samples), the number of instances allocated to training, validation, and test sets (Training Samples, Validation Samples, Test Samples), the total number of features (Total Features), and a breakdown into numerical and categorical features (Numerical Features, Categorical Features). All tasks involve binary classification.

Statistic	HELOC	Voting	Hospital Readmission	ICU Mortality	Childhood Lead	Diabetes
Total Samples	9,412	60,376	89,542	21,549	24,749	1,299,758
Training Samples	2,220	34,796	34,288	7,116	11,807	969,229
Validation Samples	278	4,349	4,286	889	1,476	121,154
Test Samples	6,914	21,231	50,968	13,544	11,466	209,375
Total Features	22	54	46	7491	7	25
Numerical Features	20	8	12	7490	4	6
Categorical Features	2	46	34	1	3	19

D.2 Common Corruptions

Let $\mathbf{x}_i^t = (\mathbf{x}_{ij}^t)_{j=1}^D \in \mathbb{R}^D$ be the i-th row of a table with D columns in the test data. We define $\bar{\mathbf{x}}_j^s$ as a random variable that follows the empirical marginal distribution of the j-th column in the training set $\mathcal{D}_s$, given by:

$$\mathbb{P}(\bar{\mathbf{x}}_j^s = k) = \frac{1}{|\mathcal{D}_s|} \sum_{i=1}^{|\mathcal{D}_s|} \mathbb{1}_{\{k\}}(\mathbf{x}_{ij}^s),$$

where $k \in \mathbb{R}$. Additionally, let $\mu_j^s = \mathbb{E}[\bar{\mathbf{x}}_j^s]$ and $\sigma_j^s = \sqrt{\text{Var}(\bar{\mathbf{x}}_j^s)}$ be the mean and standard deviation of the random variable $\bar{\mathbf{x}}_j^s$, respectively. To effectively simulate natural distribution shifts that commonly occur beyond label distribution shifts, we introduce six types of corruptions—Gaussian noise (**Gaussian**), uniform noise (**Uniform**), random missing values (**Random Drop**), common column missing across all test data (**Column Drop**), important numerical column shift (**Numerical**), and important categorical column shift (**Categorical**)—as follows:

- **Gaussian:** For $\mathbf{x}_{ij}^t$, Gaussian noise $z \sim \mathcal{N}(0, 0.1^2)$ is independently injected as:

$$\mathbf{x}_{ij}^t \leftarrow \mathbf{x}_{ij}^t + z \cdot \sigma_j^s.$$

- **Uniform:** For the $\mathbf{x}_{ij}^t$, uniform noise $u \sim \mathcal{U}(-0.1, 0.1)$ is independently injected as:

$$\mathbf{x}_{ij}^t \leftarrow \mathbf{x}_{ij}^t + u \cdot \sigma_j^s.$$

- **Random Drop:** For each column $\mathbf{x}_{ij}^t$, a random mask $m_{ij} \sim \text{Bernoulli}(0.2)$ is applied, and the feature is replaced by a random sample $\bar{\mathbf{x}}_j^s$ drawn from the empirical marginal distribution of the j-th column of the training set:

  $$\mathbf{x}_{ij}^t \leftarrow (1 - m_{ij}) \cdot \mathbf{x}_{ij}^t + m_{ij} \cdot \bar{\mathbf{x}}_j^s.$$

- **Column Drop:** For each column $\mathbf{x}_{ij}^t$, a random mask $m_j \sim \text{Bernoulli}(0.2)$ is applied, and the feature is replaced by a random sample $\bar{\mathbf{x}}_j^s$ as follows:

  $$\mathbf{x}_{ij}^t \leftarrow (1 - m_j) \cdot \mathbf{x}_{ij}^t + m_j \cdot \bar{\mathbf{x}}_j^s.$$

  Unlike random drop corruption, where the mask m_{ij} is resampled for each j-th column of the i-th test instance $\mathbf{x}_{ij}^t$, a single random mask m_j is sampled for each j-th column and applied uniformly across all test data.

- **Numerical:** Important numerical column shift simulates natural domain shifts where the test distribution of the most important numerical column deviates significantly from the training distribution. We first identify the most important numerical column, j^*, using a pre-trained XGBoost [8]. A Gaussian distribution

  $$\mathcal{N}(z|\mu_{j^*}^s, \sigma_{j^*}^s) = \frac{1}{\sqrt{2\pi\sigma_{j^*}^s}} \exp\left(-\frac{(z - \mu_{j^*}^s)^2}{2(\sigma_{j^*}^s)^2}\right)$$

  is then fitted to the j^*-th column of the training data, using $\mu_{j^*}^s$ and $\sigma_{j^*}^s$. The likelihood of each test sample $\mathbf{x}_i^t$ is then computed as $\mathcal{N}(\mathbf{x}_{ij^*}^t|\mu_{j^*}^s, \sigma_{j^*}^s)$. Finally, test samples are drawn inversely proportional to their likelihood, with the sampling probability $\mathbb{P}(\mathbf{x}_i^t)$ of $\mathbf{x}_i^t$ is defined as:

  $$\mathbb{P}(\mathbf{x}_i^t) = \frac{\mathcal{N}(\mathbf{x}_{ij^*}^t|\mu_{j^*}^s, \sigma_{j^*}^s)^{-1}}{\sum_{i'=1}^{|\mathcal{D}_t|} \mathcal{N}(\mathbf{x}_{i'j^*}^t|\mu_{j^*}^s, \sigma_{j^*}^s)^{-1}}.$$

- **Categorical:** Important categorical column shift simulates natural domain shifts where the test distribution of the most important categorical column deviates significantly from the training distribution. Again, we first identify the most important categorical column, j^*, using a pre-trained XGBoost [8]. A categorical distribution, which generalizes the Bernoulli distribution,

  $$\mathcal{C}(z|p_1, \cdots, p_K) = p_1^{\mathbb{1}_{\{1\}}(z)} \cdots p_K^{\mathbb{1}_{\{K\}}(z)},$$

  is then fitted to the j^*-th column of the training data, where K is the number of distinct categorical features in the j^*-th column, and $p_k = \mathbb{P}(\bar{\mathbf{x}}_j^s = k)$ for $k = 1, \cdots, K$. The likelihood of each test sample $\mathbf{x}_i^t$ is then computed as $\mathcal{C}(\mathbf{x}_{ij^*}^t|p_1, \cdots, p_K)$. Finally, test samples are drawn inversely proportional to their likelihood, with the sampling probability $\mathbb{P}(\mathbf{x}_i^t)$ of $\mathbf{x}_i^t$ is defined as:

  $$\mathbb{P}(\mathbf{x}_i^t) = \frac{\mathcal{C}(\mathbf{x}_{ij^*}^t|p_1, \cdots, p_K)^{-1}}{\sum_{i'=1}^{|\mathcal{D}_t|} \mathcal{C}(\mathbf{x}_{i'j^*}^t|p_1, \cdots, p_K)^{-1}}.$$

D.3 Label Distribution Shifts

- **Class Imbalance:** This label distribution shift simulates a highly class-imbalanced test stream, where labels that are rare in the training set are more likely to appear frequently in the test set. Given a class imbalance ratio $\rho = 10$, we first rank the output labels $y_i^t \in \mathcal{Y}$ for each test sample $\mathbf{x}_i^t$ in ascending order of their frequency in the training set, assigning ranks from 1 to C, where C is the number of classes. Specifically, $\text{rank}(y_i^t) = 1$ indicates that y_i^t is the least frequent label in the training set, while $\text{rank}(y_i^t) = C$ indicates that y_i^t is the most frequent. We then define the unnormalized sampling probability for each test sample $\mathbf{x}_i^t$ as:

  $$\tilde{\mathbb{P}}(\mathbf{x}_i^t) = \frac{\text{rank}(y_i^t)}{C}(\rho - 1) + 1.$$

  The normalized sampling probability $\mathbb{P}(\mathbf{x}_i^t)$ for each test sample $\mathbf{x}_i^t$ is then defined as:

  $$\mathbb{P}(\mathbf{x}_i^t) = \frac{\tilde{\mathbb{P}}(\mathbf{x}_i^t)}{\sum_{i'=1}^{|\mathcal{D}_t|} \tilde{\mathbb{P}}(\mathbf{x}_{i'}^t)}.$$

- **Temporal Correlation:** To simulate temporal correlations in test data, we employ a custom sampling strategy using the Dirichlet distribution. This approach effectively captures temporal dependencies by dynamically adjusting the label distribution over time. We begin with a uniform probability distribution $\mathbb{P}_0 = (1/C)_{j=1}^{C}$, where C is the number of classes. For sampling the i-th test instance, a probability distribution $\boldsymbol{\pi}_i$ is drawn from the Dirichlet distribution:
$$\boldsymbol{\pi}_i \sim \text{Dirichlet}(\mathbb{P}_{i-1}),$$
and then smoothed using $\eta = 10^{-6}$ to avoid zero probabilities for any class j:
$$\boldsymbol{\pi}_i = \frac{\max(\eta, \boldsymbol{\pi}_i)}{\sum_{j=1}^{C} \max(\eta, \boldsymbol{\pi}_{ij})}.$$
A label y_i^t is subsequently sampled according to $\boldsymbol{\pi}_i$, and the corresponding test instance $\mathbf{x}_i^t$ is randomly selected from the test data with label y_i^t. After the i-th sampling, the distribution $\mathbb{P}_i$ is updated using the recent history of sampled labels within a sliding window of size $w = 5$:
$$\mathbb{P}_i \leftarrow \left(\frac{1}{w} \sum_{i'=i-w+1}^{i} \mathbb{1}_{\{j\}}(y_{i'}^t) \right)_{j=1}^{C}.$$

E Baseline Details

E.1 Deep Tabular Learning Architectures

- **MLP:** Multi-Layer Perceptron (MLP) [36] is a foundational deep learning architecture characterized by multiple layers of interconnected nodes, where each node applies a non-linear activation function to a weighted sum of its inputs. In the tabular domain, MLP is often employed as a default deep learning model, with each input feature corresponding to a node in the input layer.
- **AutoInt:** Automatic Feature Interaction Learning via Self-Attentive Neural Networks (AutoInt) [47] is a model that automatically learns complex feature interactions in tasks like click-through rate (CTR) prediction, where features are typically sparse and high-dimensional. It uses a multi-head self-attentive neural network to map features into a low-dimensional space and capture high-order combinations, eliminating the need for manual feature engineering. AutoInt efficiently handles large datasets, outperforms existing methods, and provides good explainability.
- **ResNet:** ResNet for tabular data [19], is a modified version of the original ResNet architecture [21], tailored to capture intricate patterns within structured datasets. Although earlier efforts yielded modest results, recent studies have re-explored ResNet's capabilities, inspired by its success in computer vision and NLP. This ResNet-like model for tabular data is characterized by a streamlined design that facilitates optimization through nearly direct paths from input to output, enabling the effective learning of deeper feature representations.
- **FT-Transformer:** Feature Tokenizer along with Transformer (FT-Transformer) [19], represents a straightforward modification of the Transformer architecture tailored for tabular data. In this model, the feature tokenizer component plays a crucial role by converting all features, whether categorical or numerical, into tokens. Subsequently, a series of Transformer layers are applied to these tokens within the Transformer component, along with the added [CLS] token. The ultimate representation of the [CLS] token in the final Transformer layer is then utilized for the prediction.

E.2 Supervised Baselines

- **k-NN:** k-Nearest Neighbors (k-NN) is a fundamental model in tabular learning that identifies the k closest data points based on a chosen metric. It makes predictions through majority voting for classification or weighted averaging for regression. The hyperparameter k influences the model's sensitivity.

- **LogReg:** Logistic Regression (LogReg) is a linear classification model that estimates the probability of class membership using a logistic function, which maps the linear combination of features to a range of $[0, 1]$. With proper regularization, LogReg can achieve performance comparable to state-of-the-art tabular models.
- **RandomForest:** Random Forest is an ensemble learning algorithm that builds multiple decision trees to improve accuracy and reduce overfitting. It is particularly effective at capturing non-linear patterns and is robust against outliers.
- **XGBoost:** Extreme Gradient Boosting (XGBoost) [8] is a boosting algorithm that sequentially builds weak learners, typically decision trees, to correct errors made by previous models. XGBoost is known for its high predictive performance and ability to handle complex relationships through regularization.
- **CatBoost:** CatBoost [12], like XGBoost, is a boosting algorithm that excels in handling categorical features without extensive preprocessing. It is highly effective in real-world datasets, offering strong performance, albeit at the cost of increased computational resources and the need for parameter tuning.

E.3 Test-Time Adaptation Baselines

- **PL:** Pseudo-Labeling (PL) [29] leverages a pseudo-labeling strategy to update model parameters during test time.
- **TTT++:** Improved Test-Time Training (TTT++) [33] enhances test-time adaptation by using feature alignment strategies and regularization, eliminating the need to access source data during adaptation.
- **TENT:** Test ENTropy minimization (TENT) [53] updates the scale and bias parameters in the batch normalization layer during test time by minimizing entropy within a given test batch.
- **EATA:** Efficient Anti-forgetting Test-time Adaptation (EATA) [37] mitigates the risk of unreliable gradients by filtering out high-entropy samples and applying a Fisher regularizer to constrain key model parameters during adaptation.
- **SAR:** Sharpness-Aware and Reliable optimization (SAR) [38] builds on TENT by filtering samples with large entropy, which can cause model collapse during test time, using a predefined threshold.
- **LAME:** Laplacian Adjusted Maximum-likelihood Estimation (LAME) [6] employs an output adaptation strategy during test-time, focusing on adjusting the model's output probabilities rather than tuning its parameters.

F Further Experimental Details

F.1 Further Implementation Details

All experiments are conducted on two servers. The first server is equipped with a 40-core Intel Xeon E5-2630 v4 CPU, 252GB RAM, 4 NVIDIA TITAN Xp GPUs, and runs Ubuntu 18.04.4. The second server has a 40-core Intel Xeon E5-2640 v4 CPU, 128GB RAM, 8 NVIDIA TITAN Xp GPUs, and runs Ubuntu 22.04.4. All architectures were implemented using Python 3.8.16 with PyTorch [40] and PyTorch Geometric [14]. The specific versions of all software libraries and frameworks used are provided in the `AdapTable/requirements.txt` file of the supplementary materials. We also include our source code in `AdapTable` folder of the supplementary materials. Please refer to this for all experimental details and to clarify any uncertainties.

F.2 Hyperparameters for Supervised Baselines

For k-NN, LogReg, RandomForest, XGBoost, and CatBoost, optimal parameters are determined for each dataset using a random search with 10 iterations on the validation set. The search space for each method is specified in Table 6.

Table 6: Hyperparameter search space of supervised baselines. # neighbors denotes the number of neighbors, # estim denotes the number of estimators, depth denotes the maximum depth, and lr denotes the learning rate, respectively.

Method	Search Space
k-NN	# neighbors: $\{2, \cdots, 12\}$
RandomForest	# estim: $\{50, 100, 150, 200\}$, depth: $\{2, 3, \cdots, 12\}$
XGBoost	# estim: $\{50, 100, 150, 200\}$, depth: $\{2, 3, \cdots, 12\}$, lr: $\{0.01, 0.01 + (1 - 0.01)/19, \cdots, 1\}$, gamma: $\{0, 0.05, \cdots, 0.5\}$
CatBoost	# iterations: $\{50, 100, \cdots, 2000\}$, lr: $\{0.01, 0.01 + (1 - 0.01)/19, \cdots, 1\}$, depth: $\{5, \cdots, 40\}$

F.3 Hyperparameters for TTA Baselines

In scenarios where the test set is unknown, tuning the hyperparameters of TTA methods on the test set would be considered cheating. Therefore, we tune all hyperparameters for each TTA method and backbone classifier architecture using the Numerical common corruption on the CMC tabular dataset, which we did not use as test data in OpenML-CC18 [5] benchmark. PL, TENT [53], and SAR [38] require three main hyperparameters—learning rate, number of adaptation steps per batch, and the option for episodic adaptation, where the model is reset after each batch. PL [29] and TENT use a learning rate of 0.0001 with 1 adaptation step and episodic updates. Additionally, SAR requires a threshold to filter high-entropy samples and is configured with a learning rate of 0.001, 1 adaptation step, and episodic updates. For TTT++ [33], EATA [37], and LAME [6], we follow the authors' hyperparameter settings, except for the learning rate and adaptation steps. TTT++ and EATA were configured with a learning rate of 0.00001, 10 adaptation steps, and episodic updates. LAME, which only adjusts output logits, does not require hyperparameters related to gradient updates. For all baselines, hyperparameter choices remained consistent across different architectures, including MLP, AutoInt, ResNet, and FT-Transformer. The hyperparameter search space for each method are detailed in Table 7.

Table 7: Hyperparameter search space of test-time adaptation baselines. Here, we only denote the common hyperparameters, where method specific hyperparameters are specified in Section F.3.

Hyperparameter	Search Space
lr	$\{10^{-3}, 10^{-4}, 10^{-5}, 10^{-6}\}$
# steps	$\{1, 5, 10, 15, 20\}$
episodic	{True, False}

F.4 Hyperparameters for AdapTable

AdapTable requires three test-time hyperparameters—the smoothing factor α, and the low and high uncertainty quantiles q_{low} and q_{high}. For fairness, we tune all AdapTable hyperparameters across different backbone architectures using the Numerical common corruption on the CMC dataset from the OpenML-CC18 benchmark [5], which is not used as test data. We observe that AdapTable's hyperparameter choices remain consistent across various architectures, including MLP, AutoInt, ResNet, and FT-Transformer. Notably, AdapTable demonstrates high insensitivity to variations in α, q_{low}, and q_{high}, which are uniformly set to 0.1, 0.25, and 0.75, respectively, across all datasets and architectures.

G Additional Analysis

G.1 Latent Space Visualizations

In Figure 9, we further visualize latent spaces of test instances using t-SNE across six different datasets and four representative deep tabular learning architectures to illustrate the observation discussed in Section 2.1. This visualization highlights the complex decision boundaries within the latent space of tabular data, which are significantly more intricate than those observed in other domains. By comparing the upper four rows—HELOC, Voting, Hospital Readmission, and Childhood Lead—with the lower two rows—linearized image data (MFEAT-PIXEL) and homogeneous DNA string sequences (DNA)—it becomes evident that the latent space decision boundaries in the tabular domain

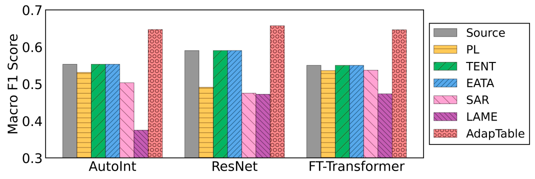

Figure 7: The average macro F1 score of AdapTable and TTA baselines across three datasets (HELOC, Voting, Childhood Lead) using various backbone architectures.

are particularly complex. According to WhyShift [32], this complexity is primarily due to latent confounders inherent in tabular data and concept shifts, where such confounders cause output labels to vary greatly for nearly identical inputs. As discussed in Section 2.1, this further underscores the limitations of existing TTA methods [51, 16, 33, 6, 56], which often depend on the cluster assumption.

G.2 Reliability Diagrams

Figure 10 presents additional reliability diagrams across five different datasets and four representative deep tabular learning architectures, illustrating that tabular data often displays a mix of overconfident and underconfident prediction patterns. This contrasts with the consistent overconfidence observed in the image domain [50] and underconfidence in the graph domain [54]. As shown in Figure 10, the Voting and Hospital Readmission datasets consistently exhibit overconfident behavior across all architectures, while the HELOC, Childhood Lead, and Diabetes datasets demonstrate underconfident tendencies. These observations underscore the need for a tabular-specific uncertainty calibration method.

G.3 Label Distribution Shifts and Prediction Bias Towards Source Label Distributions

We demonstrate that the data distribution shift we primarily target in the tabular domain—label distribution shift—occurs frequently in practice. Figure 11 presents the source label distribution (a), target label distribution (b), pseudo label distribution for test data using the source model (c), and the estimated target label distribution after applying our label distribution handler (d) across the five datasets. Comparing (a) and (b) in each row, it is evident that label distribution shift occurs across all datasets. In (c), we observe that the marginal label distribution predicted by the source model is commonly biased towards the source label distribution. Lastly, (d) illustrates that our label distribution handler effectively estimates the target label distribution, guiding the pseudo label distribution towards the target label distribution.

G.4 Entropy Distributions

We highlight a unique characteristic of tabular data: model prediction entropy consistently shows a strong bias toward underconfidence. To illustrate this, we present entropy distribution histograms for test instances across six datasets and four representative deep tabular learning architectures in Figure 12. A clear pattern emerges when comparing the upper four rows (HELOC, Voting, Hospital Readmission, Childhood Lead) with the lower two (Optdigits, DNA). The upper rows exhibit consistently high entropy, indicating a skew toward underconfidence, while the lower rows do not, except for Childhood Lead, where extreme class imbalance causes the model to collapse to the major class. This analysis highlights the distinct bias of tabular data toward underconfident predictions, a pattern less common in other domains. This aligns with findings that applying unsupervised objectives like entropy minimization to high-entropy samples can result in gradient explosions and model collapse [38].

H Additional Experiments

H.1 Result Across Diverse Model Architectures

In Figure 7, we report AdapTable's effectiveness across three mainstream tabular learning architectures—AutoInt [47], ResNet [19], and FT-Transformer [19]. We report the average macro F1 score across three datasets—HELOC, Voting, and Childhood Lead. None of the baselines outperform

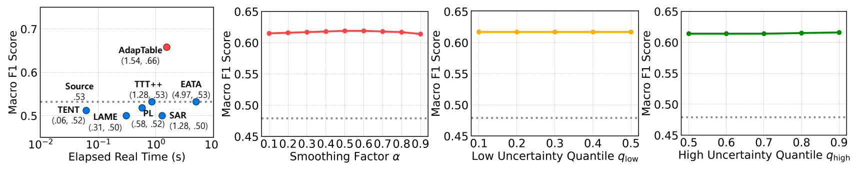

Figure 8: Computational efficiency (leftmost figure) and hyperparameter sensitivity analysis of AdapTable (three figures on the right) using MLP on the HELOC and Childhood Lead datasets, respectively.

the original source model, with LAME [6] even showing significant performance drops. In contrast, AdapTable consistently achieves significant improvements across all architectures, highlighting its robustness and versatility.

H.2 Further Analysis

Computational efficiency. The leftmost part of Figure 8 compares the computational efficiency of AdapTable with TTA baselines. On the HELOC dataset, AdapTable's total elapsed time is approximately 1.54 seconds, translating to about 0.0002 seconds per sample, which is highly desirable. Moreover, AdapTable achieves an optimal efficiency-efficacy trade-off.

Hyperparameter sensitivity. Figure 8 further analyzes the hyperparameter sensitivity of AdapTable on the Childhood Lead dataset. As shown in the figure, AdapTable remains highly insensitive to changes in the smoothing factor α, low uncertainty quantile q_{low}, and high uncertainty quantile q_{high}.

H.3 Detailed Results Across Common Corruptions and Datasets

Figure 4 presents the average F1 score across six types of common corruption and three datasets. Here, we provide more detailed results, including the standard errors. As shown in Table 8, AdapTable outperforms baseline TTA methods by a large margin across all datasets and corruption types. This further highlights the empirical efficacy of AdapTable, not only in handling label distribution shifts but also in addressing various common corruptions.

H.4 All Results Across Datasets and Model Architectures

In Figure 7, we demonstrate the effectiveness of AdapTable across various tabular model architectures by reporting the average performance across three datasets. Here, we provide the mean and standard error for each dataset and architecture. As shown in Table 9, AdapTable consistently achieves state-of-the-art performance with significant improvements across all model architectures and datasets. This further underscores the versatility and robustness of AdapTable.

H.5 Additional Computational Efficiency Analysis

One may wonder whether the post-training time required for AdapTable's shift-aware uncertainty calibrator is prohibitively long. To address this concern, we measure and report the elapsed real time for post-training our shift-aware uncertainty calibrator on the medium-scale Hospital Readmission dataset using the FT-Transformer architecture. The post-training process takes approximately 9.2 seconds. For small- and medium-scale datasets, the post-training process typically requires only a few seconds, and even in our largest experimental setting, the time remains minimal, taking at most a few minutes.

I Limitations and Broader Impacts

I.1 Limitations

Similar to other test-time training (TTT) methods [51, 33, 16], AdapTable requires an additional post-training stage to integrate a shift-aware uncertainty calibrator during the source model's training phase. While full test-time adaptation methods [53, 37, 38] avoid this, our analysis in Section 2.1 and experiments in Section 4 show that they fail in the tabular domain due to their focus on input covariate shifts, which are often entangled with concept shifts. According to WhyShift [32], concept shifts, driven by changes in latent confounders, require natural language descriptions of the shift conditions, necessitating a data-centric approach. Additionally, while AdapTable performs well across various corruptions beyond label distribution shifts (Figure 4), it is primarily focused on addressing label distribution shifts. Further exploration is needed to assess its effectiveness in handling input covariate shifts or concept shifts.

I.2 Broader Impacts

Tabular data is prevalent across industries such as healthcare [27, 26], finance [48, 49], manufacturing [23], and public administration [17]. Our research addresses the critical yet underexplored challenge of distribution shifts in tabular data, a problem that has not received sufficient attention. We believe that our approach can significantly enhance the performance of machine learning models in various industries by improving model adaptation to tabular data, thereby creating meaningful value in practical applications. Through our data-centric analysis in Section 2, we identify why existing TTA methods fail in the tabular domain and introduce a tabular-specific approach for handling label distribution shifts in Section 3. We hope this work will provide valuable insights for future research on test-time adaptation in tabular data. Additionally, by making our source code publicly available, we aim to support real-world applications across various fields, benefiting both academia and industry.

Table 8: The average macro F1 score (%) with their standard errors for TTA baselines is reported across six common corruptions—Gaussian, Uniform, Random Drop, Column Drop, Numerical, and Categorical—over three datasets—HELOC, Voting, and Childhood Lead. The results are averaged over three random repetitions.

Dataset	Method	Gaussian	Uniform	Random Drop	Column Drop	Numerical	Categorical
HELOC	Source	33.1 ± 0.0	33.0 ± 0.0	31.4 ± 0.1	32.3 ± 1.4	33.8 ± 0.2	32.3 ± 0.3
	PL	31.2 ± 0.0	31.2 ± 0.0	30.6 ± 0.0	31.1 ± 0.7	32.1 ± 0.2	30.4 ± 0.2
	TENT	33.1 ± 0.0	33.0 ± 0.0	31.4 ± 0.1	32.3 ± 1.4	33.8 ± 0.2	32.3 ± 0.3
	EATA	33.1 ± 0.0	33.0 ± 0.0	31.4 ± 0.1	32.3 ± 1.4	33.8 ± 0.2	32.3 ± 0.3
	SAR	31.9 ± 0.1	32.0 ± 0.1	30.7 ± 0.2	31.3 ± 0.8	32.4 ± 0.4	31.4 ± 0.3
	LAME	30.1 ± 0.0	30.1 ± 0.0	30.1 ± 0.0	30.1 ± 0.0	30.9 ± 0.1	29.4 ± 0.2
	AdapTable	**57.6 ± 0.1**	**57.8 ± 0.0**	**53.0 ± 0.1**	**52.1 ± 3.2**	**58.1 ± 0.1**	**58.9 ± 0.4**
Voting	Source	76.6 ± 0.0	76.5 ± 0.0	72.5 ± 0.2	72.8 ± 0.4	76.3 ± 0.1	85.2 ± 0.1
	PL	75.6 ± 0.3	75.2 ± 0.3	71.1 ± 0.5	70.6 ± 0.5	75.9 ± 0.1	85.1 ± 0.1
	TENT	76.6 ± 0.0	76.5 ± 0.0	72.5 ± 0.2	72.8 ± 0.4	76.3 ± 0.1	85.2 ± 0.1
	EATA	76.6 ± 0.0	76.5 ± 0.0	72.5 ± 0.2	72.8 ± 0.4	76.3 ± 0.1	85.2 ± 0.1
	SAR	67.2 ± 1.0	64.0 ± 0.2	61.8 ± 1.0	60.8 ± 0.8	69.9 ± 0.1	84.2 ± 0.1
	LAME	39.4 ± 0.2	39.4 ± 0.1	37.3 ± 0.0	37.8 ± 0.2	39.4 ± 0.2	81.4 ± 0.2
	AdapTable	**78.9 ± 0.0**	**78.6 ± 0.1**	**74.9 ± 0.1**	**75.5 ± 0.5**	**78.0 ± 0.1**	**85.0 ± 0.4**
Childhood Lead	Source	47.9 ± 0.0	47.9 ± 0.0	47.9 ± 0.0	47.9 ± 0.0	48.1 ± 0.0	48.8 ± 0.0
	PL	47.9 ± 0.0	47.9 ± 0.0	47.9 ± 0.0	47.9 ± 0.0	48.1 ± 0.0	48.8 ± 0.0
	TENT	47.9 ± 0.0	47.9 ± 0.0	47.9 ± 0.0	47.9 ± 0.0	48.1 ± 0.0	48.8 ± 0.0
	EATA	47.9 ± 0.0	47.9 ± 0.0	47.9 ± 0.0	47.9 ± 0.0	48.1 ± 0.0	48.8 ± 0.0
	SAR	47.9 ± 0.0	47.9 ± 0.0	47.9 ± 0.0	47.9 ± 0.0	48.1 ± 0.0	48.8 ± 0.0
	LAME	47.9 ± 0.0	47.9 ± 0.0	47.9 ± 0.0	47.9 ± 0.0	48.1 ± 0.0	48.8 ± 0.0
	AdapTable	**61.4 ± 0.1**	**61.5 ± 0.0**	**58.0 ± 0.1**	**55.9 ± 1.6**	**62.8 ± 0.2**	**53.1 ± 0.2**

Table 9: The average macro F1 score (%) with their standard errors for TTA baselines is reported across three datasets—HELOC, Voting, and Childhood Lead—using three model architectures—AutoInt, ResNet, and FT-Transformer. The results are averaged over three random repetitions.

Model	Method	HELOC	Voting	Childhood Lead
AutoInt	Source	34.9 ± 0.0	77.5 ± 0.0	47.9 ± 0.0
	PL	31.6 ± 0.0	76.5 ± 0.1	47.9 ± 0.0
	TENT	34.9 ± 0.0	77.5 ± 0.0	47.9 ± 0.0
	EATA	34.9 ± 0.0	77.5 ± 0.0	47.9 ± 0.0
	SAR	62.0 ± 0.4	31.2 ± 0.7	47.9 ± 0.0
	LAME	30.1 ± 0.0	37.3 ± 0.0	47.9 ± 0.0
	AdapTable	**56.3 ± 0.1**	**79.2 ± 0.0**	**61.8 ± 0.1**
ResNet	Source	52.0 ± 0.0	76.6 ± 0.0	47.9 ± 0.0
	PL	34.3 ± 0.1	73.3 ± 0.1	47.9 ± 0.0
	TENT	52.0 ± 0.0	76.6 ± 0.0	47.9 ± 0.0
	EATA	52.0 ± 0.0	76.7 ± 0.0	47.9 ± 0.0
	SAR	55.1 ± 0.5	52.2 ± 0.5	47.9 ± 0.0
	LAME	30.1 ± 0.0	75.1 ± 0.1	47.9 ± 0.0
	AdapTable	**61.9 ± 0.0**	**78.7 ± 0.0**	**61.3 ± 0.1**
FT-Transformer	Source	33.0 ± 0.0	77.3 ± 0.0	47.9 ± 0.0
	PL	30.6 ± 0.0	76.0 ± 0.1	47.9 ± 0.0
	TENT	33.0 ± 0.0	77.3 ± 0.0	47.9 ± 0.0
	EATA	33.0 ± 0.0	77.3 ± 0.0	47.9 ± 0.0
	SAR	35.3 ± 0.1	73.6 ± 0.3	47.9 ± 0.0
	LAME	30.7 ± 0.1	71.5 ± 0.1	47.9 ± 0.0
	AdapTable	**55.0 ± 0.0**	**79.2 ± 0.1**	**61.7 ± 0.1**

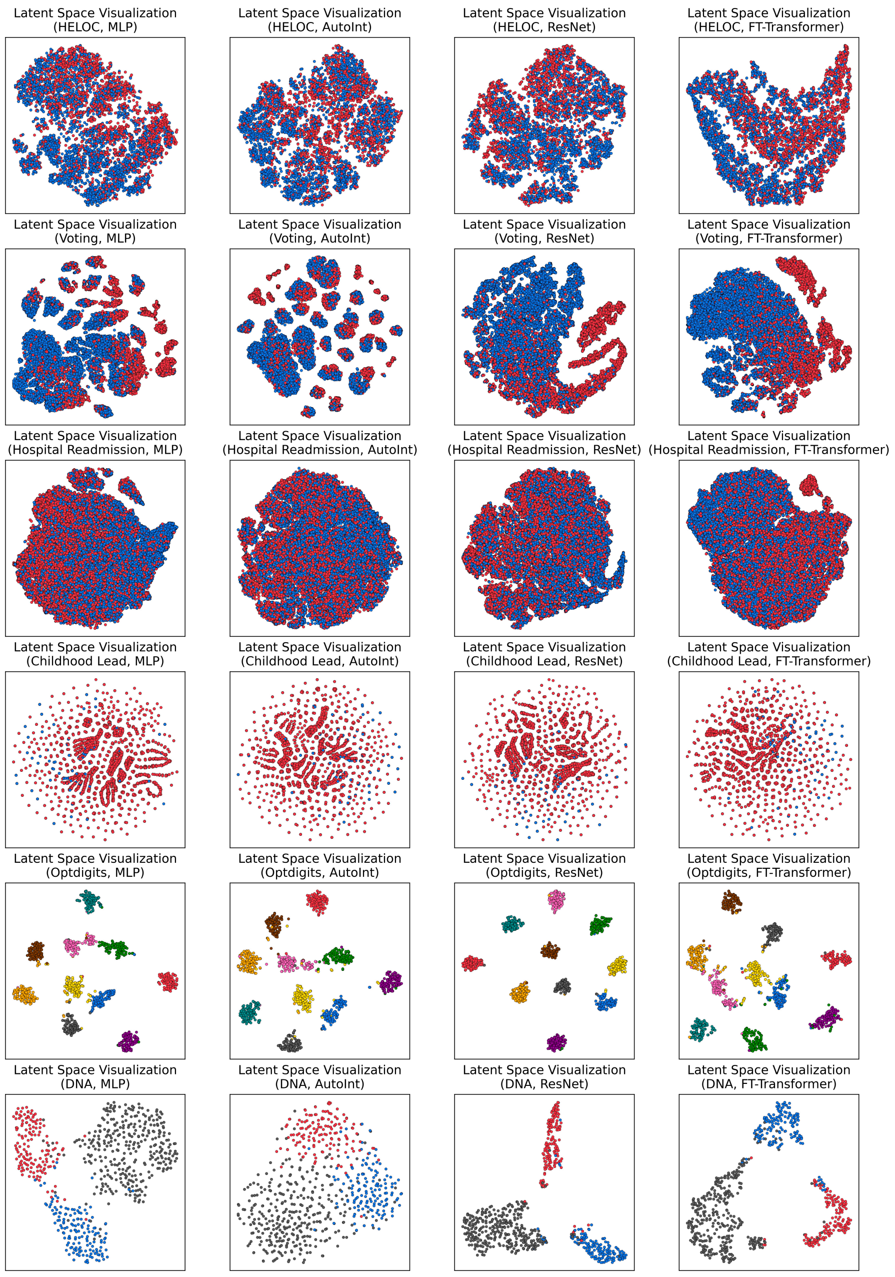

Figure 9: Latent space visualizations of test samples using t-SNE across six diverse datasets, including tabular datasets (HELOC, Voting, Hospital Readmission, and Childhood Lead) and non-tabular datasets (Optdigits, DNA), applied to various deep tabular learning architectures.

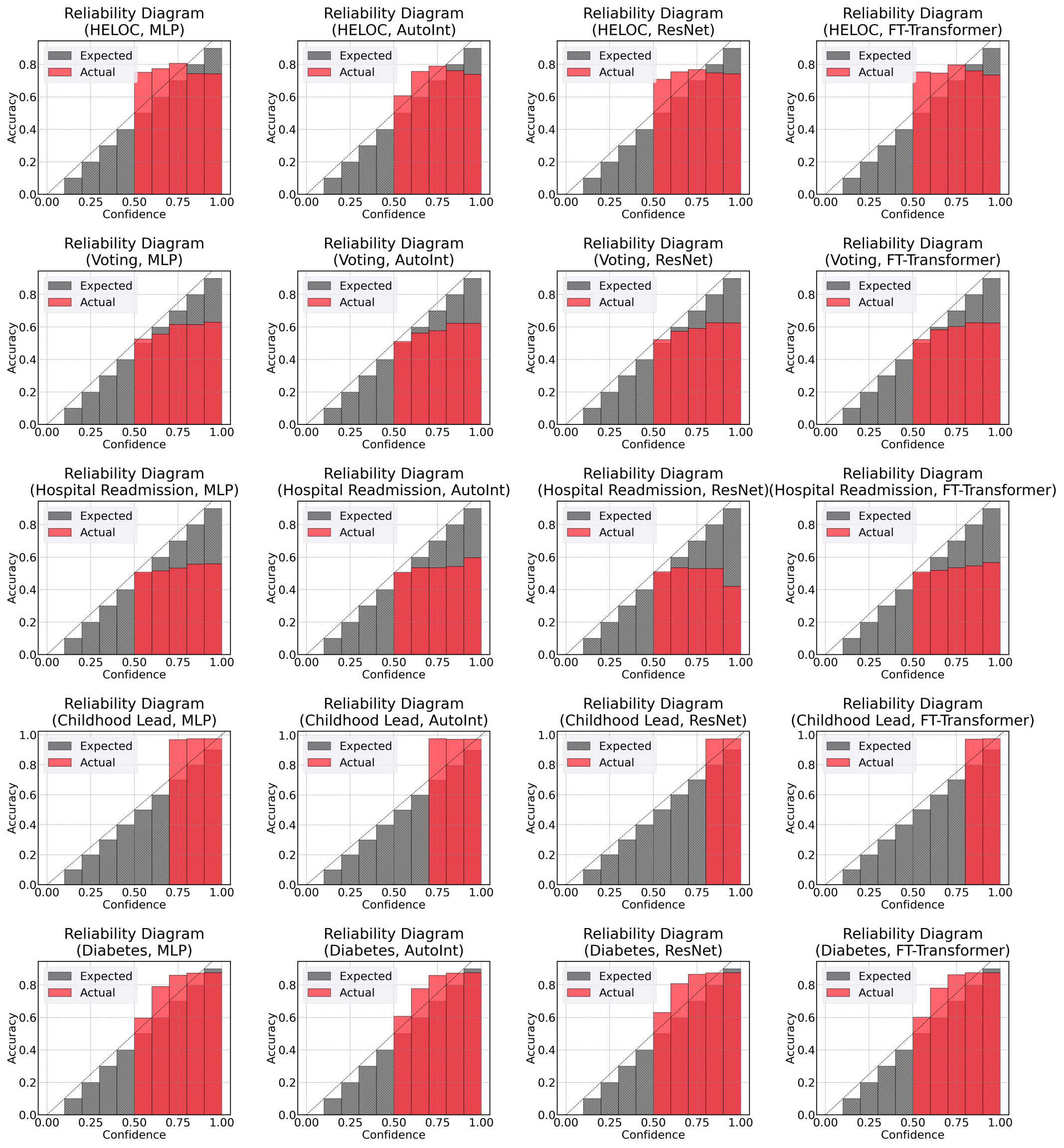

Figure 10: Reliability diagrams for test instances across five different tabular datasets (HELOC, Voting, Hospital Readmission, Childhood Lead, and Diabetes) and four representative deep tabular learning architectures (MLP, AutoInt, ResNet, FT-Transformer).

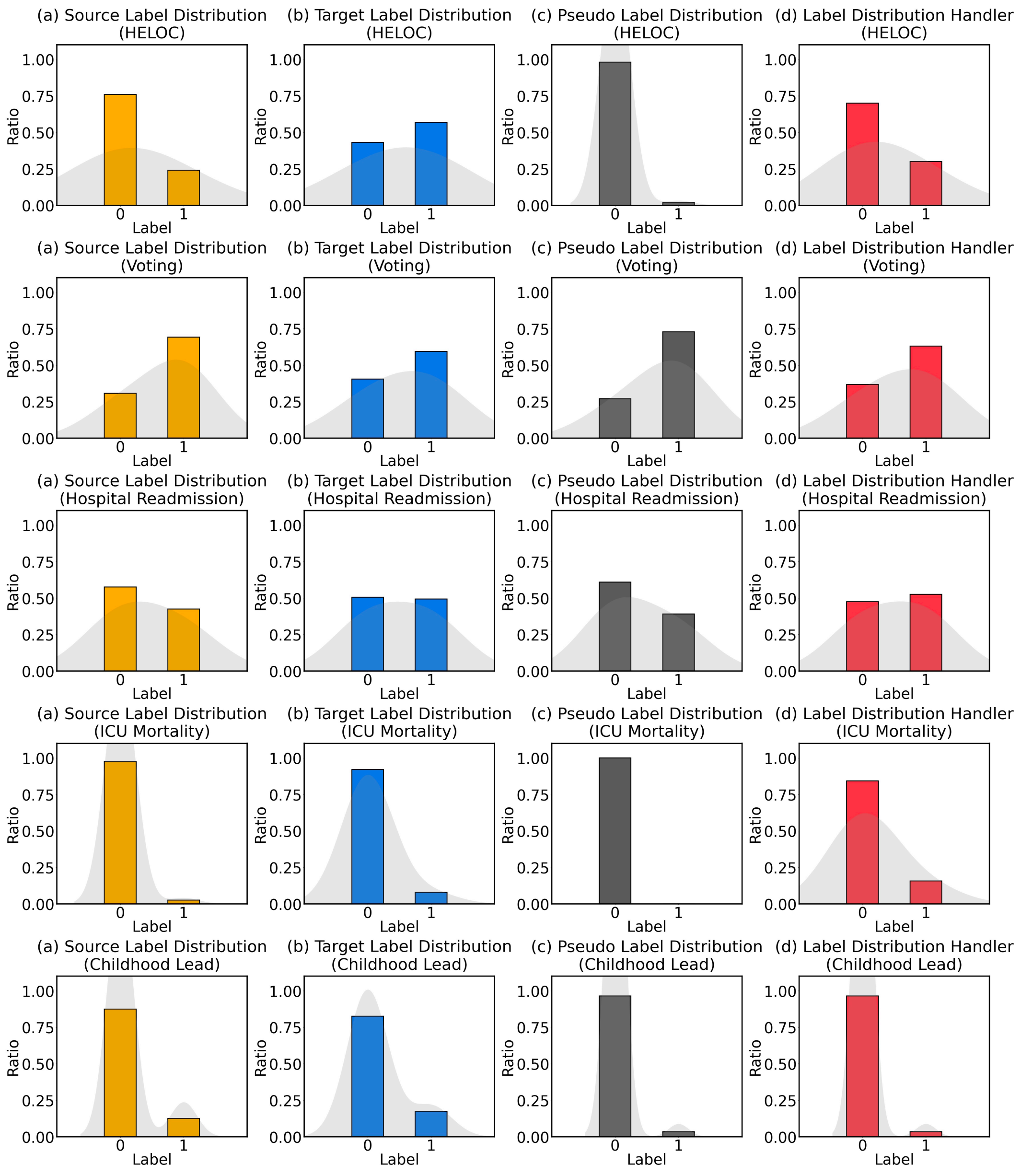

Figure 11: Label distribution histograms for test instances showing (a) source label distribution, (b) target label distribution, (c) pseudo label distribution, and (d) estimated target label distribution after applying our label distribution handler, across five tabular datasets (HELOC, Voting, Hospital Readmission, Childhood Lead, and Diabetes) using MLP.

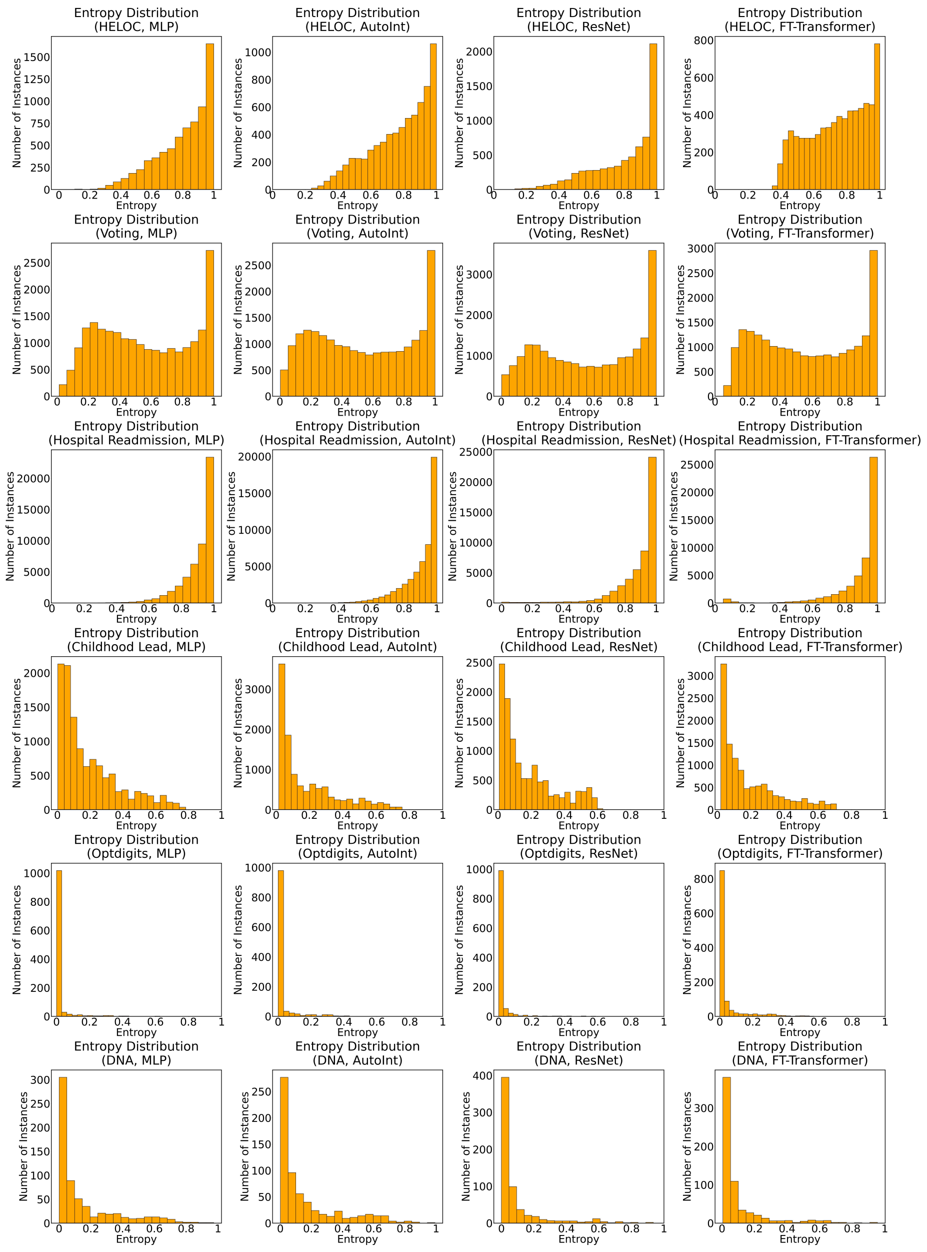

Figure 12: Entropy distribution histograms of test samples across six diverse datasets, including tabular datasets (HELOC, Voting, Hospital Readmission, and Childhood Lead) and non-tabular datasets (Optdigits, DNA), applied to four deep tabular learning architectures. Prediction entropies are normalized by dividing by the maximum entropy, $\log C$, where C represents the number of classes for each dataset.