Tabular Learnwares Can Be Repurposed for Seemingly Irrelevant New Tasks

Peng Tan, Fei-Fan Yang, Zhi-Hao Tan, Zhi-Hua Zhou

National Key Laboratory for Novel Software Technology, Nanjing University, China
School of Artificial Intelligence, Nanjing University, China
{tanp,yangff,tanzh,zhouzh}@lamda.nju.edu.cn

Abstract

The *learnware* paradigm aims to help users solve new tasks by reusing existing models rather than starting from scratch. A learnware consists of a model and the *specification* describing its capabilities. Numerous learnwares are accommodated by the *learnware dock system*. When users solve tasks with the system, learnwares that fully match the user task are often scarce or unavailable. This paper focuses on tabular classification tasks and explores reusing learnwares for new user tasks with significantly different feature and label spaces, leveraging the potential of numerous existing specialized tabular models developed for various tasks. Under the learnware paradigm, we find that tabular learnwares that seem semantically irrelevant can sometimes be beneficial for new user tasks. The proposed method relies solely on model-predicted probabilities and does not require gradient information, making it applicable to a wide range of tabular models. Experiments suggest that tabular learnwares can be reused beyond their original purpose across heterogeneous tasks.

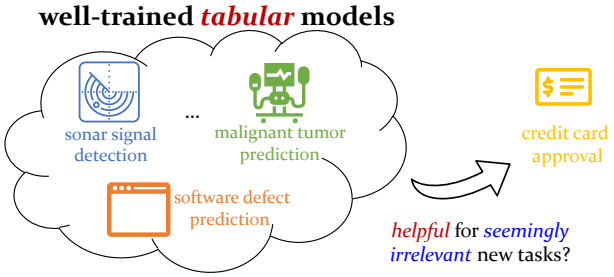

Figure 1: The significant heterogeneity of tabular data often confines a model to a single specific task. Although some research explores reusing tabular models for new tasks, these efforts are restricted to tasks that are either identical or semantically similar. This paper investigates the potential of reusing tabular models for seemingly irrelevant tasks within the learnware paradigm.

Introduction

Tabular data is one of the most common data types in real-world applications (Shwartz-Ziv and Armon 2022), supporting numerous critical applications such as medical diagnosis (Johnson et al. 2023), financial risk control (Wachowicz 2020) etc. However, building high-performing models from scratch poses significant challenges, as it requires high-quality labeled data, huge computational resources, etc. Furthermore, reusing existing tabular models is also challenging because tabular tasks are highly heterogeneous, with feature and label spaces varying significantly across tasks. Additionally, tabular data often contains privacy-sensitive information, making data sharing challenging or even impossible. This significantly complicates model reuse. Consequently, *leveraging the potential of numerous existing specialized tabular models for new tasks without exposing raw data* remains a significant challenge.

The learnware paradigm (Zhou 2016) provides a systematic way for model sharing, accommodation, and reuse without leaking raw data. *Learnware* = *model* + *specification*. The model refers to a machine learning model trained by the developer, while the specification characterizes the model without exposing raw data, allowing the model to be *automatically* identified and reused by future users who know nothing about the model in advance (Zhou and Tan 2024). Note that the specification is to be generated on the developer's side based on some statistical learning strategy, with training data preservation ability (Lei, Tan, and Zhou 2024). When future users submit task requirements to the *learnware dock system* that hosts learnwares, they do not need to disclose their own data; instead, they can generate a task specification based on data at their own side. The system will then identify helpful learnwares based on the user's task requirements and can even assemble multiple learnwares to help users solve their tasks. This mechanism of "*model sharing, reusing, and assembling without data disclosing*" naturally meets the tabular scenario. Recently, to advance research on the learnware paradigm, the first systematic open-source implementation of the learnware dock system, Beimingwu (Tan et al. 2024b), has been released.

The learnware paradigm has shown its effectiveness in scenarios no matter whether the learnwares share the same feature space (Wu et al. 2023; Zhang et al. 2021; Xie et al. 2023; Liu, Tan, and Zhou 2024, 2025a,b) or not (Tan et al. 2023, 2024a). In this paper, we disclose an amazing utility of the learnware paradigm; that is, occasionally some seemingly irrelevant learnwares, such as those with neither com-

mon features nor common labels with user task, can be helpful to solve the user task, while the learnware paradigm can effectively identify and reuse them for the user task.

In the problem illustrated in Figure 1, we start with a learnware dock system hosting tabular models from a variety of tasks. The central challenge is determining whether these seemingly irrelevant models can be helpful for a new task. It is worth noting that reusing a single, highly customized tabular model for a significantly different task is quite challenging, especially without access to the original data, a constraint that is common when dealing with highly sensitive and private tabular data. *A simple model repository would fail here*, as the model alone lacks the necessary information for such reuse. However, within the learnware paradigm, the specification provides crucial information about the model's capabilities and original task distribution without exposing the raw data, thereby offering the key information needed to enable cross-task reuse while preserving privacy. This paper indicates that the decision boundary of the model, independent of its original semantics, has the potential of benefiting some new tasks with simple transformation.

Nevertheless, a critical challenge lies in efficiently adapting the decision boundary of a trained model to a new task without resorting to an impractical exhaustive search of all possible transformations. To tackle this issue, we propose an efficient approach that generates label mappings through cross-class performance evaluation and subsequently constructs feature mappings. The proposed mechanism relies solely on the model's predicted probabilities, making it applicable across a wide range of tabular models. This paper shows that reusing models trained on seemingly irrelevant tasks can still enhance performance on new tasks sometimes. The effectiveness of tabular learnwares for seemingly irrelevant new user tasks hinges on two key factors: an effective reuse mechanism and model richness of the system. Indeed, effective learnware reuse often relies on a sufficiently rich system, as it is common that only a small fraction of learnwares can benefit a new task. This paper presents the first study on tabular learnware repurposing, with a focus on the reuse mechanism. Our contributions are summarized as:

- This paper preliminarily explores reusing tabular learnware for user tasks with significantly different feature and label spaces. It finds that tabular learnware can be reused for some seemingly irrelevant new tasks by applying simple transformations to the model boundary.
- We propose a general cross-task reuse mechanism for tabular learnware that relies solely on model-predicted probabilities, making it applicable to most tabular models. By guiding feature mapping learning with label correspondence, we effectively enhance cross-task learnware reuse performance. Building upon this mechanism, we further design a learnware recommendation criterion of estimated learnware reuse performance, which surpasses traditional distribution-based metrics.
- Binary classification experiments show that even when no existing learnware in the system can be directly applied to the user's task, seemingly irrelevant learnwares can still improve task performance sometimes.

Preliminary

After the learnware dock system is constructed, how can the system effectively recommend helpful models given user requirements without exposing model training data and facilitating model reuse? The key lies in the specification describing the model's capabilities and strengths, which is the central component of the learnware. An effective specification should: (1) capture the model's capabilities; (2) not leak raw training data; and (3) remain compact and efficient for storage and comparison.

Currently, the most widely adopted specification that achieves these objectives for the structured data task is the Reduced Kernel Mean Embedding (RKME) specification (Zhou and Tan 2024). Its core idea is to approximate the joint distribution of features and model outputs using a small set of representative samples, with the approximated distribution itself serving as the model specification. The RKME specification has been proven to prevent leakage of raw training data (Lei, Tan, and Zhou 2024). Given the model $f : \mathcal{X} \mapsto \mathcal{Y}$ and its training data $D = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^m$, the RKME specification first generates the dataset including model outputs $\hat{D} = \{z_i = (\boldsymbol{x}_i, \hat{y}_i)\}_{i=1}^m$, where $\hat{y}_i = f(\boldsymbol{x}_i)$. Then, the RKME specification is a set of weighted samples $\{(\beta_j, \boldsymbol{t}_j)\}_{j=1}^n (n \ll m)$ that approximates the joint distribution $P(X, \hat{Y})$ of features and model outputs, where $\boldsymbol{\beta}$ and $\boldsymbol{t}$ are the weights and mimic samples, respectively. The specification is generated by minimizing the distribution distance of two kernel mean embeddings (KME) (Schölkopf and Smola 2002) of raw dataset $\{\boldsymbol{z}_i\}_{i=1}^m$ and reduced set $\{\beta_j, \boldsymbol{t}_j\}_{j=1}^n$, with mimic samples $\boldsymbol{t}_j = (\boldsymbol{v}_j, y_j)$ including feature part $\boldsymbol{v}_j$ and label part y_j:

$$\min_{\boldsymbol{\beta}, \boldsymbol{t}} \left\| \frac{1}{m} \sum_{i=1}^{m} k\left(\boldsymbol{z}_i, \cdot\right) - \sum_{j=1}^{n} \beta_j k\left(\boldsymbol{t}_j, \cdot\right) \right\|_{\mathcal{H}}^2, \qquad (1)$$

where $k(\cdot, \cdot)$ is the kernel function, $\mathcal{H}$ is the associated reproducing kernel Hilbert space.

In simple cases, such as when only covariate shift (Shimodaira 2000) exists between tasks, it suffices to only approximate the feature distribution (Wu et al. 2023). Subsequently, to address the issue that label information can be overshadowed by feature information, Tan et al. (2024a) proposed generating the specification by minimizing the distribution loss of the sum of the marginal distribution $P(X)$ and the conditional distributions $p(X|\hat{Y})$, rather than measuring solely on the joint distribution $P(X, \hat{Y})$ as shown in Eq. (1). In addition, they showed that the user *task requirement* can be generated in a similar manner by optimizing the summed distribution loss over the marginal and conditional distributions. *This is the learnware specification and user requirement generation mechanism adopted in this work*. For unstructured data, such as images and text, a variant of the RKME specification, namely the PAVE specification (Tan et al. 2025), has been proposed.

In practice, the system provides a unified Gaussian kernel $k(\cdot, \cdot)$ for structured data tasks, which allows developers to generate specifications locally from training data and models, while users generate corresponding task requirements.

Problem Setup

The learnware paradigm typically involves two stages:

- *Submitting stage*: developers train a model on their own data and generate the specification, then pack it with the model as learnware and submit it to the system;
- *Deploying stage*: users submit task requirements to the system, which then recommends the most relevant learnwares and facilitates heterogeneous learnwares reuse.

For scenarios involving learnwares from diverse tabular classification tasks, the problem is formulated as:

Submitting Stage. Developers train a model f_i using the training data $D_i = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^{n}$, which are defined on $\mathcal{X}_i \times \mathcal{Y}_i$ with C classes. They then generate the RKME specification $\boldsymbol{s}_i = \{(\beta_{ij}, \boldsymbol{v}_{ij}, y_{ij})\}_{j=1}^{m_i}$ and submit the learnware $L_i := (f_i, \boldsymbol{s}_i)$ to the system. Subsequently, the system is constructed as $\mathcal{B} = \{L_i\}_{i=1}^{N}$.

Deploying Stage. The user wants to solve the task defined on $\mathcal{X}_0 \times \mathcal{Y}_0$ with limited labeled data $D_0^l = \{(\boldsymbol{x}_{0j}, y_{0j})\}_{j=1}^{n_l}$, consisting of C classes. Although the user can train a model starting from scratch, better performance can be achieved with the help of the system by reusing knowledge from other tasks. The stage has the following steps:

- *Requirement generation*: the user generates the RKME task requirement $\boldsymbol{s}_0 = \{(\beta_{0j}, \boldsymbol{v}_{0j}, y_{0j})\}_{j=1}^{m_0}$ based on the limited labeled data D_0^l and test data $D_0^u = \{(\tilde{\boldsymbol{x}}_{0j})\}_{j=1}^{n_u}$.
- *Learnware adaptation*: Given the user requirement $\boldsymbol{s}_0$, the system can adapt learnwares for the user task, enabling learnwares with different feature spaces and label spaces to be directly used on the user task. This process is formulated as $\text{Adapt}(L_i, \boldsymbol{s}_0) \mapsto L_i^*$, where $L_i^* = (f_i^*, \boldsymbol{s}_i)$, adapting f_i from the original task $\mathcal{X}_i \times \mathcal{Y}_i$ to the user task $\mathcal{X}_0 \times \mathcal{Y}_0$ as f_i^*.
- *Learnware recommendation*: the system identifies helpful learnwares which are adapted $\mathcal{B}_0^* = \{L_{i_1}^*, \cdots, L_{i_k}^*\}$ to the user with the recommendation criterion.
- *Learnware reuse*: the user reuses recommended learnwares $\mathcal{B}_0^*$ to get better task performance.

The problem formulation is illustrated in Figure 2, and the key challenge lies in effectively adapting potentially useful learnwares to new user tasks with seemingly irrelevant feature spaces and label spaces. Notably, neither the model's raw data nor the user's is exposed during this process.

Our Method

This section first describes the critical step by which the learnware dock system adapts potentially useful learnware to new user tasks with significantly different feature spaces and label spaces, without accessing to the raw data of the model or user tasks. Then, the system's workflow is outlined.

Tabular Learnware Adaptation

The most critical challenge in the problem formulation is learnware adaptation $\text{Adapt}(L_i, \boldsymbol{s}_0) \mapsto L_i^*$, which involves adapting existing learnware L_i to fit user tasks with different feature and label spaces. Our key insight is that, even if the original and new tasks are semantically irrelevant, the decision boundary of the learnware can still benefit the new task after appropriate transformations sometimes.

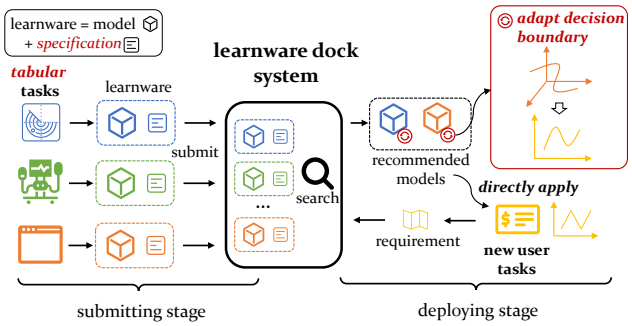

Figure 2: The problem setup of repurposing tabular learnwares for new tasks involves two stages: 1) *submitting stage*: developers submit tabular learnwares from various tasks to the system. 2) *deploying stage*: users submit task requirements, and the system recommends adapted helpful models.

Building on this idea, we formalize the learnware adaptation as the task of finding optimal feature and label mapping functions. Notably, many tabular learnwares in the system may be non-differentiable, as gradient boosting decision trees are widely used in tabular tasks (Borisov et al. 2022). *To ensure broad applicability of adaptation mechanism, we treat tabular learnware models as black boxes only accessing model outputs*. Without gradient information, end-to-end joint optimization of the two mapping functions becomes infeasible. To address this, we propose a two-stage algorithm. In the first stage, we construct a "class transfer performance matrix" using only the model's prediction outputs and efficiently solve for the label mapping. In the second stage, we proceed to solve for the feature mapping guided by determined label mapping. Details are as follows.

Learnware Adaptation Overview. Given the user task requirement $\boldsymbol{s}_0 = \{(\beta_{0j}, \boldsymbol{v}_{0j}, y_{0j})\}_{j=1}^{m_0}$ and the learnware $L_i = (f_i, \boldsymbol{s}_i)$ with $\boldsymbol{s}_i = \{(\beta_{ij}, \boldsymbol{v}_{ij}, y_{ij})\}_{j=1}^{m_i}$, the learnware adaptation process $\text{Adapt}(L_i, \boldsymbol{s}_0) \mapsto L_i^*$ adapts the model f_i, originally defined on $\mathcal{X}_i \times \mathcal{Y}_i$, to fit the user task defined on $\mathcal{X}_0 \times \mathcal{Y}_0$. This adaptation is accomplished by learning a feature mapping $T_i^X : \mathcal{X}_0 \mapsto \mathcal{X}_i$ and a label mapping $T_i^Y : \mathcal{Y}_i \mapsto \mathcal{Y}_0$, resulting in the adapted model $f_i^* = T_i^Y \circ f_i \circ T_i^X$. Ultimately, the process produces the adapted learnware $L_i^* = (f_i^*, \boldsymbol{s}_i)$. Our proposed method for learnware adaptation $\text{Adapt}(L_i, \boldsymbol{s}_0) \mapsto L_i^*$ has three steps:

- Step1: Compute the class transfer performance matrix

$$\text{ClassTransferMatrix}(L_i, \boldsymbol{s}_0) \mapsto \mathbf{C} \tag{2}$$

- Step2: Determine the label mapping heuristically

$$\text{HeuristicSearchLabelMapping}(\mathbf{C}) \mapsto T_i^Y \tag{3}$$

- Step3: Learn the feature mapping given label mapping

$$\text{FeatureMapping}(\boldsymbol{s}_i, \boldsymbol{s}_0, T_i^Y) \mapsto T_i^X \tag{4}$$

Step1: Construct Class Transfer Performance Matrix. To efficiently determine the label mapping $T_i^Y : \mathcal{Y}_i \mapsto \mathcal{Y}_0$, we first construct a class transfer performance matrix $\mathbf{C}$, which evaluates the effectiveness of using each learnware class to address each user task class. Specifically, each entry c_{kl} represents the performance of applying k-th learnware class to handle l-th user task class, which is computed as follows:

A) Feature Transformation: Given the learnware $L_i = (f_i, \boldsymbol{s}_i)$ and user requirement $\boldsymbol{s}_0$ associated with different feature spaces, class transfer performance calculation requires applying a feature transformation to the requirement. Specifically, the user requirement for the l-th class is extracted as $\boldsymbol{s}_{0,l} := \{(\beta_{0j}, \boldsymbol{v}_{0j}, y_{0j}) | y_{0j} = l\}$ (with indices j in $\mathcal{I}_{0,l}$), and the learnware specification for the k-th class is $\boldsymbol{s}_{i,k} := \{(\beta_{ij}, \boldsymbol{v}_{ij}, y_{ij}) | y_{ij} = k\}$ (with indices j in $\mathcal{I}_{i,k}$). Next, a feature mapping T^X is trained by minimizing the distribution loss between the KME of the projected user requirement $\boldsymbol{s}_{0,l}^{\text{proj}}$ and the learnware specification $\boldsymbol{s}_{i,k}$, where $\boldsymbol{s}_{0,l}^{\text{proj}} := \{(\beta_{0j}, T^X(\boldsymbol{v}_{0j}), y_{0j}) | y_{0j} = l\}$. The feature mapping T^X is trained by minimizing the following objective:

$$T^X = \underset{T^X}{\operatorname{argmin}} \|\text{KME}_{\boldsymbol{s}_{0,l}^{\text{proj}}} - \text{KME}_{\boldsymbol{s}_{i,k}}\|_{\mathcal{H}_k}, \tag{5}$$

where $\text{KME}_{\boldsymbol{s}_{0,l}^{\text{proj}}} = \sum_{j \in \mathcal{I}_{0,l}} \beta_{0j} k(T^X(\boldsymbol{v}_{0j}), \cdot)$ is the KME of the user requirement $\boldsymbol{s}_{0,l}$ after feature mapping T^X, and $\text{KME}_{\boldsymbol{s}_{i,k}} = \sum_{j \in \mathcal{I}_{i,k}} \beta_{ij} k(\boldsymbol{v}_{ij}, \cdot)$ is the KME of the learnware specification $\boldsymbol{s}_{i,k}$.

B) Performance Evaluation: Using the adapted model $\tilde{f} = T_{kl}^Y \circ f_i \circ T^X$, where T_{kl}^Y maps learnware label k to user task label l, we evaluate the performance of the learnware on the user requirement $\boldsymbol{s}_{0,l}$. The resulting performance c_{kl} is given by:

$$c_{kl} = \sum_{j \in \mathcal{I}_{0,l}} \beta_{0j} \ell(\tilde{f}(\boldsymbol{v}_{0j}), y_{0j}), \tag{6}$$

where $\ell(\cdot)$ denotes the loss function.

Step2: Solve the Label Mapping. After generating the class transfer performance matrix $\mathbf{C}$, where each entry c_{kl} represents the performance of solving the user class l using the learnware class k, we employ a greedy search strategy to find the suitable label mapping T_i^Y. Specifically, we first estimate the class ratios in the user validation set $\boldsymbol{s}_0$. Then, starting with the user class with the highest ratio, we iteratively assign it to the available learnware class that yields the highest performance according to matrix $\mathbf{C}$. Once a learnware class is assigned, it is removed from consideration. This process constructs a permutation function $T_i^Y \in \pi : \{1, \cdots, C\} \mapsto \{1, \cdots, C\}$ over the label space.

Step3: Solve the Feature Mapping. After determining the label mapping $T_i^Y : \mathcal{Y}_i \mapsto \mathcal{Y}_0$, the corresponding feature mapping $T_i^X : \mathcal{X}_0 \mapsto \mathcal{X}_i$ can be learned. Given the learnware specification $\boldsymbol{s}_i = \{(\beta_{ij}, \boldsymbol{v}_{ij}, y_{ij})\}_{j=1}^{m_i}$ and the user task requirement $\boldsymbol{s}_0 = \{(\beta_{0j}, \boldsymbol{v}_{0j}, y_{0j})\}_{j=1}^{m_0}$, the adapted learnware specification after applying the feature mapping is defined as $\boldsymbol{s}_i^* = \{(\beta_{ij}, T_i^X(\boldsymbol{v}_{ij}), y_{ij})\}_{j=1}^{m_i}$, while the adapted user requirement after applying the label mapping is defined as $\boldsymbol{s}_0^* = \{(\beta_{0j}, \boldsymbol{v}_{0j}, T_i^Y(y_{0j}))\}_{j=1}^{m_0}$. The feature mapping T_i^X is learned by minimizing the distribution losses between the adapted user requirement $\boldsymbol{s}_0^*$ and the adapted learnware specification $\boldsymbol{s}_i^*$ while preserving local structure. The objective is as follows:

$$\underset{T_i^X}{\operatorname{argmin}} \mathcal{L} = \text{MMD}_{\text{Marg}} + \alpha \text{MMD}_{\text{Cond}} + \lambda \text{L}_{\text{sim}}, \tag{7}$$

where α, λ are trade-off parameters. Objective details are:

- MMD_{Marg} is the MMD distance between the P_X of the adapted user requirement $\boldsymbol{s}_0^*$ and the adapted learnware specification $\boldsymbol{s}_i^*$, ensuring global distribution similarity. The term is defined as: $\text{MMD}_{\text{Marg}} = \|\text{KME}_{\boldsymbol{s}_0^*} - \text{KME}_{\boldsymbol{s}_i^*}\|_{\mathcal{H}_k}$, where $\text{KME}_{\boldsymbol{s}_0^*} = \sum_{j=1}^{m_0} \beta_j k(T_i^X(\boldsymbol{v}_{0j}), \cdot)$ and $\text{KME}_{\boldsymbol{s}_i^*} = \sum_{j=1}^{m_i} \beta_j k(\boldsymbol{v}_{ij}, \cdot)$ are the KMEs of the adapted user requirement and the adapted learnware specification, respectively.
- MMD_{Cond} represents the sum of MMD distances between the conditional distributions $P_{X|Y}$ of the requirement $\boldsymbol{s}_0^*$ and the specification $\boldsymbol{s}_i^*$ for each class, ensuring local distribution similarity. It is formally defined as: $\text{MMD}_{\text{Cond}} = \sum_{c=1}^{C} \|\text{KME}_{\boldsymbol{s}_{0,c}^*} - \text{KME}_{\boldsymbol{s}_{i,c}^*}\|_{\mathcal{H}_k}$, where $\text{KME}_{\boldsymbol{s}_{0,c}^*} = \sum_{j \in \mathcal{I}_{0,c}} \beta_j k(T_i^X(\boldsymbol{v}_{0j}), \cdot)$ and $\text{KME}_{\boldsymbol{s}_{i,c}^*} = \sum_{j \in \mathcal{I}_{i,c}} \beta_j k(\boldsymbol{v}_{ij}, \cdot)$ denote the KMEs of the adapted user requirement and the adapted learnware specification for class c, respectively.
- L_{sim} serves as the manifold regularizer, designed to encourage T_i^X to preserve the local structure of distribution during feature transformation. It is defined as: $\text{L}_{\text{sim}} = \sum_{j=1}^{m_0} \sum_{k=1}^{m_0} w_{jk} \|T_i^X(\boldsymbol{v}_{0j}) - T_i^X(\boldsymbol{v}_{0k})\|^2$, where $w_{jk} = \text{sim}(\boldsymbol{v}_{0j}, \boldsymbol{z}_{0k})$, and the similarity function $\text{sim}(x, y)$ is the RBF kernel: $\text{sim}(x, y) = \exp(-\gamma \|x - y\|^2)$.

Summary. When the following conditions are met: 1) raw data for the model and user task is unavailable, and 2) the model is treated as a black box, providing only output probabilities, the procedure for tabular learnware adaptation $\text{Adapt}(L_i, \boldsymbol{s}_0) \mapsto L_i^*$ can be summarized as follows: First, the system constructs a class transfer performance matrix to efficiently solve the label mapping $T^Y : \{0, 1, \cdots, C-1\} \mapsto \{0, 1, \cdots, C-1\}$. This process involves training C^2 simple feature mappings using MLP and evaluating the model's performance. Next, the label mapping is derived from the performance matrix using a greedy algorithm. Finally, the feature mapping is optimized with guidance of the determined label mapping, ensuring both the global and local distributions of user requirement and learnware specification are similar.

Discussion. This part gives more discussion on the proposed reuse mechanism.

A) Why Not Simultaneously Optimize Mappings? To adapt the model $f_i : \mathcal{X}_i \mapsto \mathcal{Y}_i$ to the user task defined on $\mathcal{X}_0 \times \mathcal{Y}_0$, it is necessary to learn both a feature mapping and a label mapping. A straightforward approach is to optimize these mappings *simultaneously* by minimizing the loss

of the adapted learnware $\tilde{f}_i = T_i^Y \circ f_i \circ T_i^X$ on the user task requirements $\boldsymbol{s}_0 = \{(\beta_{0j}, \boldsymbol{v}_{0j}, y_{0j})\}_{j=1}^{m_0}$, which can be expressed as:

$$\min_{T_i^Y, T_i^X} \sum_{j=1}^{m_0} \beta_{0j} \ell(T_i^Y \circ f_i \circ T_i^X(\boldsymbol{v}_{0j}), y_{0j}). \tag{8}$$

However, since the learnware model f_i is treated as a black box and gradient is unavailable, directly optimizing the mappings T_i^X and T_i^Y jointly becomes infeasible.

B) Why Guide the Feature Mapping Learning with Label Mapping? When joint optimization of feature mapping and label mapping is infeasible, one alternative is to first optimize the feature mapping T_i^X by minimizing the distribution loss on P_X, and then optimize the label mapping T_i^Y by minimizing loss on the user requirement $\boldsymbol{s}_0 = \{(\beta_{0j}, \boldsymbol{v}_{0j}, y_{0j})\}_{j=1}^{m_0}$. This approach generalizes the method proposed in (Tan et al. 2024b).

$$\begin{aligned} &\text{a) } T_i^X = \operatorname*{argmin}_{T_i^X} \|\text{KME}_{\boldsymbol{s}_0^{\text{proj}}} - \text{KME}_{\boldsymbol{s}_i}\|_{\mathcal{H}_k}, \\ &\text{b) } T_i^Y = \operatorname*{argmin}_{T_i^Y} \sum_{j=1}^{m_0} \beta_{0j} \ell(T_i^Y \circ f_i \circ T_i^X(\boldsymbol{v}_{0j}), y_{0j}), \end{aligned} \tag{9}$$

where $\text{KME}_{\boldsymbol{s}_0^{\text{proj}}} = \sum_{j=1}^{m_0} \beta_{0j} k(T_i^X(\boldsymbol{v}_{0j}), \cdot)$. In the first step, the feature mapping is optimized to ensure that the marginal distribution P_X of user requirement $\boldsymbol{s}_0$ after feature transformation is similar to the learnware specification $\boldsymbol{s}_i$. In the second step, the label mapping is learned based on the preliminary adapted model output $f_i \circ T_i^X(\boldsymbol{v}_{0j})$ and labels y_{0j}. However, learning a feature mapping without considering the label mapping results in the absence of supervision from the user task during the feature mapping optimization. As a result, the output of the adapted learnware $f_i \circ T_i^X(\cdot)$ is irrelevant to the user task. Essentially, this approach is equivalent to training a label mapping (e.g., a simple MLP) after applying a random projection to original features, without exploiting the model's ability.

Entire Workflow

When a user exploits the learnware dock system $\mathcal{B} = \{L_1, L_2, \cdots, L_N\}$ to better solve the task, the user first generates the task requirement $\boldsymbol{s}_0 = \{\beta_{0j}, \boldsymbol{v}_{0j}, y_{0j}\}_{j=1}^{m_0}$ based on the minor labeled data $\mathcal{D}_0^l$ and test data $\mathcal{D}_0^u$ and provides it to the system. Importantly, this requirement does not expose the raw data of the user's task. Next, the system evaluates the reusability score r_i for each learnware L_i. This evaluation involves learnware adaptation, denoted as $\text{Adapt}(L_i, \boldsymbol{s}_0) \mapsto L_i^*$, followed by calculating the performance of the adapted learnware model $f_i^* = T_i^Y \circ f_i \circ T_i^X$ on the user's task requirement $\boldsymbol{s}_0$:

$$r_i = \sum_{j=1}^{m_0} \beta_{0j} \ell(f_i^*(\boldsymbol{v}_{0j}), y_{0j}), \tag{10}$$

Based on the reusability scores, the system recommends the top-k learnwares $\mathcal{B}_k = \{L_1^*, L_2^*, \cdots, L_k^*\}$ to the user. These learnwares, although originating from different tasks, can be directly applied to the user's task after learnware adaptation conducted by the system. Finally, the user averages the predictions from the recommended learnwares and their own self-trained model to achieve improved predictions.

The learnware identification is the most time-consuming part of the workflow, as the system traverses all learnwares and computes reusability scores, which requires optimizing C^2+1 mappings for each learnware. Notably, this procedure simultaneously adapts each learnware to the new task, allowing immediate reuse of recommended learnwares on the user task. The identification efficiency can be enhanced by: (1) using learnware indexing (Liu, Tan, and Zhou 2024) or clustering (Xie et al. 2023) to avoid full traversal of the system; (2) parallelizing the optimization of the C^2 independent MLP mappings; and (3) exploring learnware recommendation criterions that do not rely on model prediction.

Experiments

Experiment Setup

Datasets. We test our method on a large tabular benchmark called Tabzilla benchmark (McElfresh et al. 2023). We filter the dataset whose size is less than 200 and select the binary classification tasks, resulting in a total of 79 tasks. The number of instances across the selected tasks ranges from 208 to 1,000,000, while the number of features varies between 3 and 4,296.

Compared Methods. Exploring the system accommodating tabular models from various tasks to assist new user tasks with significantly different feature spaces and label spaces is a new problem, we first compare with user self-trained methods, including the widely used gradient boosting decision trees (LightGBM (Ke et al. 2017), `LGB`), simple yet robust linear model (`Linear`), the state-of-the-art method `TabPFN` (Hollmann et al. 2023, 2025) on small tabular datasets (Ye, Liu, and Chao 2025) and recent competitive tabular methods like `ModernNCA` (Ye, Yin, and Zhan 2025), `TabM` (Gorishniy, Kotelnikov, and Babenko 2025), `TabCaps` (Chen et al. 2023). However, training from scratch often underperforms with limited data, while reusing knowledge from other tasks can further enhance results. `Xtab` (Zhu et al. 2023) trains a shared backbone for transfer across tasks. *Model hub approaches select reusable models from similar tasks, though they are not designed for arbitrary feature spaces*. We adapt several methods to incorporate both learnware recommendation and reuse for comparison: `MMD` (Tan et al. 2024b; Wu et al. 2023), `GW` (Mémoli 2011), `JOINT` (Tan et al. 2024a), and `FA` (Tan et al. 2024b). Since the model hub methods use ensembles, we also report top-3 ensembles of scratch-trained models ($\text{Ensemble}_{\text{top-k}}$) of typical tabular models and ensembles of strong competitor TabPFN v2 ($\text{TabPFN}_{\text{t-\{ensemble time\}}}$).

Experiment Configurations. Among 79 tasks, 15% are designated as user tasks, while the remaining tasks provide models stored in the learnware dock system. Each dataset is split into training, validation, and test sets with a 8:1:1 ratio based on (McElfresh et al. 2023). Each submitted model in the system is trained with LightGBM via cross-validation.

Category	Method	Rank ↓		Acc ↑		Wins ↑
		Mean	Min	Mean	Std	W/T/L
Single model (training from scratch/fine-tuning pre-trained model)	$\text{TabPFN}_{\text{v2-basic}}$	6.04	**1**	80.28	0.76	10/0/3
	TabM	6.88	**1**	79.13	0.09	8/2/3
	$\text{Linear}_{\text{tuned}}$	7.85	3	80.17	0.28	11/0/2
	ModernNCA	12.19	6.5	79.15	1.13	12/0/1
	$\text{LGB}_{\text{tuned}}$	13	8.5	78.91	0.64	13/0/0
	Xtab	14.54	4.5	52.30	2.72	12/0/1
	TabCaps	15.96	13	71.29	0.95	13/0/0
Multiple models (only use models from user data)	$\text{Ensemble}_{\text{top1}}$ [a]	5.85	3	80.41	0.35	11/0/2
	$\text{Ensemble}_{\text{top2}}$ [b]	6.35	**1**	80.32	0.37	9/2/2
	$\text{Ensemble}_{\text{top3}}$	6.65	3	80.23	0.33	10/0/3
	$\text{TabPFN}_{\text{t-200}}$	8.38	**1**	80.02	0.37	10/0/3
	$\text{TabPFN}_{\text{t-30}}$	9.96	5	79.78	0.64	12/0/1
Multiple models (also use models from other task)	FA	7.23	**1**	80.16	0.41	8/1/4
	JOINT	8.19	2	79.42	0.46	9/1/3
	MMD	9.92	2	76.28	0.75	10/1/2
	GW	10.19	**1**	78.87	0.67	10/0/3
	LwRepurpose	**3.81**	**1**	**81.12**	0.47	-

[a] Linear+TabPFN; [b] Linear+$\text{LGB}_{\text{tuned}}$ +TabPFN;

Table 1: Performance comparison of our method (***LwRepurpose***) and contenders under *limited labeled data* setting.

Category	Method	Rank ↓		Acc ↑		Wins ↑
		Mean	Min	Mean	Std	W/T/L
Single model (training from scratch/fine-tuning pre-trained model)	$\text{LGB}_{\text{tuned}}$	7.79	**1**	88.14	0.00	9/1/2
	$\text{TabPFN}_{\text{v2-basic}}$	8.17	2	87.95	0.19	9/1/2
	ModernNCA	9.08	**1**	87.94	0.31	9/0/3
	$\text{Linear}_{\text{tuned}}$	10.75	**1**	86.59	0.00	11/0/1
	Xtab	12.88	**1**	84.73	0.00	11/0/1
	TabCaps	14.96	2	77.81	1.03	11/0/1
	TabM	15.08	7	79.85	0.36	12/0/0
Multiple models (only use models from user data)	$\text{Ensemble}_{\text{top1}}$ [a]	5.33	**1**	88.81	0.08	6/0/6
	$\text{Ensemble}_{\text{top2}}$ [b]	6.79	**1**	88.72	0.00	9/0/3
	$\text{Ensemble}_{\text{top3}}$	7.12	2	88.74	0.22	8/3/1
	$\text{TabPFN}_{\text{t-200}}$	8.67	1.5	87.78	0.33	11/0/1
	$\text{TabPFN}_{\text{t-30}}$	8.75	2	87.86	0.45	12/0/0
Multiple models (also use models from other task)	JOINT	7.54	**1**	88.12	0.24	8/1/3
	MMD	7.67	**1**	88.26	0.18	9/1/2
	GW	8.42	4	88.05	0.43	11/0/1
	FA	10.25	**1**	86.13	0.19	9/1/2
	LwRepurpose	**3.75**	**1**	**89.17**	0.15	-

[a]Linear+$\text{LGB}_{\text{tuned}}$ +TabPFN+$\text{LGB}_{\text{default}}$; [b]Linear+$\text{LGB}_{\text{tuned}}$ +$\text{LGB}_{\text{default}}$;

Table 2: Performance comparison of our method (***LwRepurpose***) and contenders under *full labeled data* setting.

The accuracy on user task test data serves as the evaluation metric. In our method, the specification size is set as a piece-wise constant function with $M = O(\log N)$, where N is the training set size. Feature space transformation uses a single-layer network with ReLU activation (Glorot, Bordes, and Bengio 2011) and the Adam (Kingma and Ba 2015) optimizer. Training runs for 200 epochs with a learning rate of 10^{-1}, MMD coefficient 10, and manifold regularizer 10^{-2}. Five models are recommended from the system, and all experiments are repeated five times.

Baseline Comparison

Users Have Limited Labeled Data. As shown in Table 1, when the user has 100 labeled data, our method consistently outperforms all baselines, including single models, their ensembles, and model hub-based approaches. Among single models, $\text{TabPFN}_{\text{v2-basic}}$ achieves the highest accuracy. Xtab performs the worst, likely due to difficulties arising from using a shared backbone across diverse tasks. For model hub-based methods, JOINT, GW, MMD, and FA focus on adapting models by minimizing distribution distances without considering actual model performance, making them less effective than our approach. Regarding ensembles, the combination of $\text{Linear}_{\text{tuned}}$+$\text{TabPFN}_{\text{v2-basic}}$ delivers the best results among prior approaches; however, even the best-performing ensembles of scratch-trained models fall short of our method, highlighting the greater diversity and effectiveness of system-recommended models.

Users Have Full Labeled Data. Table 2 presents a comparison of our method and contenders under the scenario where the user has access to full training data. Among single-model approaches, $\text{LGB}_{\text{tuned}}$ achieves the best performance, surpassing $\text{TabPFN}_{\text{v2-basic}}$. Although ensembling these models further improves task performance, it still does not surpass our method, which integrates the user's $\text{LGB}_{\text{tuned}}$ model with system-recommended models. This indicates that system-recommended models provide valuable and unique patterns that go beyond what can be achieved when training from scratch on user data.

	Models from user data			Models from other tasks				
Scenario	Ens1st	Ens2nd	Ens3rd	FA	MMD	JOINT	GW	***LwRepurpose***
Limited	0.03	-0.01	-0.09	-0.03	-1.00	-0.21	-0.35	**0.21**
Full	0.33	0.29	0.30	-1.00	0.06	-0.01	-0.04	**0.51**

Table 3: Comparison of *accuracy gains (after normalization)* for methods involving multiple models under the limited labeled data scenario and full labeled data scenario.

Accuracy Gains of Methods Involving Multiple Models. Table 3 reveals differences between the two model sources. Methods based on *models from user data* deliver stable but moderate gains. In contrast, approaches that rely on *models from other tasks* often reduce performance due to poor heterogeneous learnware reuse. Our approach, *LwRepurpose*, despite also leveraging models from other tasks, effectively improves cross-task learnware reuse and achieves larger accuracy gains in both limited and full labeled data scenarios.

Evaluation of Key Procedures

After comparing the final performance of the proposed workflow with contenders, this section provides a detailed analysis of key procedures in the system workflow including learnware cross-task reuse and recommendation.

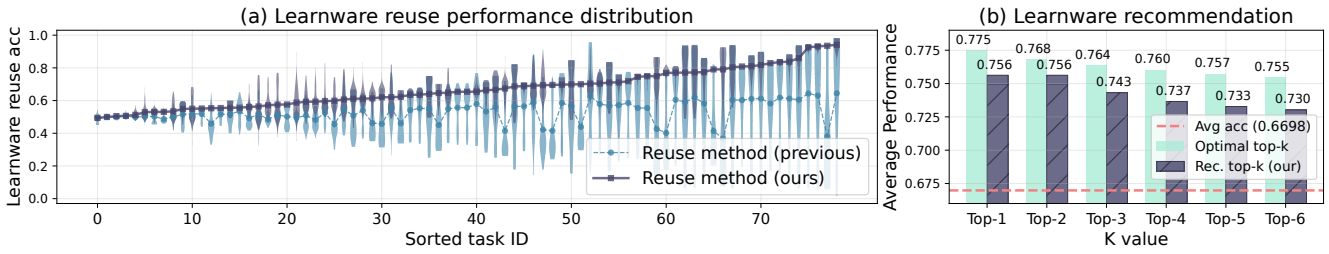

Figure 3: Superiority of our method in learnware reuse and identification.

Heterogeneous Learnware Reuse. This paper introduces a cross-task learnware reuse mechanism that heuristically identifies suitable label mappings and then guides feature mapping based on the optimized label mappings. In contrast, previous methods focused solely on optimizing feature mapping without leveraging label mapping as guidance. As illustrated in Figure 3(a), which depicts the reuse performance distribution for each task when utilizing learnwares from other tasks, our approach demonstrates a significant improvement over previous methods—both in overall distribution and average reuse performance.

Heterogeneous Learnware Identification. After achieving significant improvements in cross-task learnware reuse performance, a critical challenge emerges: effectively identifying helpful learnware from the system without accessing users' raw data. As illustrated in Figure 3(b), the average reuse performance of learnware across all user tasks is 66.98%. The top-k learnware recommended by our method closely align with those selected by the market oracle, significantly outperforming average learnware performance.

Resource Usage. Our method is highly resource-efficient because of simple mapping and small specification. On average, adapting a single binary learnware to the user task takes 2.8s, 600 FLOPs, 17.3MB memory based on a 16.21KB specification. (measured on an A100 40GB GPU).

Cross-Task Learnware Reuse Performance Gains

Figure 4 illustrates the accuracy improvements achieved by reusing learnware from another task compared to using a self-trained model, under scenarios with limited labeled data. For clarity, only results for half of the binary classification tasks are shown. In the figure, an arrow from node A to node B indicates that task B benefits from ensembling a LightGBM model trained on the full training data of task A with a self-trained TabPFN model on task B. TabPFN performs well when training data is limited, whereas LightGBM excels with abundant training data. The line width represents the magnitude of the improvement. Multiple lines connecting nodes across different scenarios and shown in various colors indicate that a task can gain from seemingly irrelevant learnware, even when its own self-trained model already performs well. The graph illustrates gains from reusing a single learnware; combining multiple learnwares can lead to further improvements.

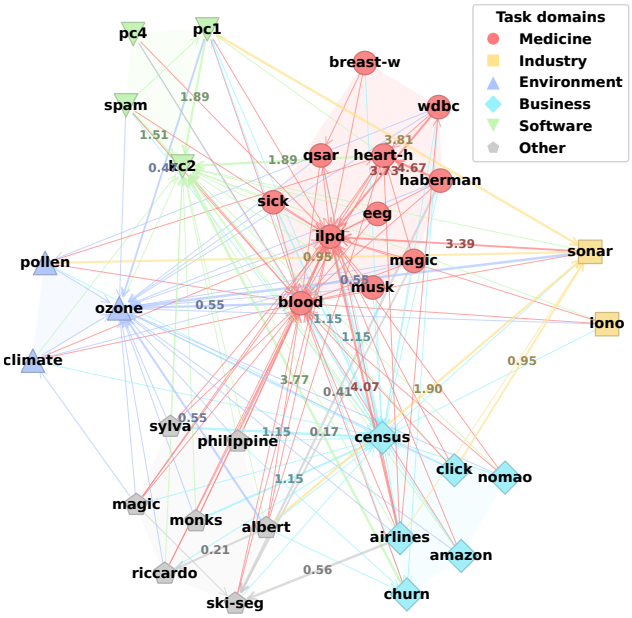

Figure 4: Performance gains by cross-task learnware reuse: this graph illustrates the accuracy improvement ($A \rightarrow B$) achieved by ensembling an adapted source learnware from task A with a self-trained model on a label-scarce target task B. Nodes are clustered by domain, with arrows indicating beneficial reuse; edges are colored by the target task's domain, while line widths and numerical labels quantify the magnitude of accuracy improvement.

Conclusion

This paper investigates repurposing tabular learnwares for new tasks with substantially different feature and label spaces *without exposing raw data*. We propose a general cross-task learnware reuse mechanism that leverages only model-predicted probabilities, making it broadly applicable to various tabular models. By supervising feature mapping learning with heuristically optimized label mappings, cross-task learnware reuse performance is substantially improved. Binary classification experiments show that even semantically irrelevant learnwares can sometimes benefit user tasks, indicating that the model decision boundary, independent of its original semantics, can be exploited for new tasks.

Acknowledgments

This research was supported by NSFC (62250069). Zhi-Hao Tan was supported by the Postdoctoral Fellowship Program of CPSF under Grant Number GZB20250396, and Jiangsu Funding Program for Excellent Postdoctoral Talent. The authors would like to thank Jia-Wei Shan for helpful discussions. We are also grateful for the anonymous reviewers for their valuable comments.

References

Borisov, V.; Leemann, T.; Seßler, K.; Haug, J.; Pawelczyk, M.; and Kasneci, G. 2022. Deep neural networks and tabular data: A survey. *IEEE Transactions on Neural Networks and Learning Systems*, 35(6): 7499–7519.

Chen, J.; Liao, K.; Fang, Y.; Chen, D.; and Wu, J. 2023. Tabcaps: A capsule neural network for tabular data classification with bow routing. In *The 11th International Conference on Learning Representations*.

Glorot, X.; Bordes, A.; and Bengio, Y. 2011. Deep sparse rectifier neural networks. In *Proceedings of the 14th International Conference on Artificial Intelligence and Statistics*, 315–323.

Gorishniy, Y.; Kotelnikov, A.; and Babenko, A. 2025. TabM: Advancing tabular deep learning with parameter-efficient ensembling. In *The 13th International Conference on Learning Representations*.

Hollmann, N.; Müller, S.; Eggensperger, K.; and Hutter, F. 2023. TabPFN: A transformer that solves small tabular classification problems in a second. In *The 11th International Conference on Learning Representations*.

Hollmann, N.; Müller, S.; Purucker, L.; Krishnakumar, A.; Körfer, M.; Hoo, S. B.; Schirrmeister, R. T.; and Hutter, F. 2025. Accurate predictions on small data with a tabular foundation model. *Nature*, 637(8044): 319–326.

Johnson, A. E.; Bulgarelli, L.; Shen, L.; Gayles, A.; Shammout, A.; Horng, S.; Pollard, T. J.; Hao, S.; Moody, B.; Gow, B.; et al. 2023. MIMIC-IV, a freely accessible electronic health record dataset. *Scientific Data*, 10(1): 1.

Ke, G.; Meng, Q.; Finley, T.; Wang, T.; Chen, W.; Ma, W.; Ye, Q.; and Liu, T. 2017. LightGBM: A highly efficient gradient boosting decision tree. In *Advances in Neural Information Processing Systems 30*, 3146–3154.

Kingma, D. P.; and Ba, J. 2015. Adam: A method for stochastic optimization. In Bengio, Y.; and LeCun, Y., eds., *The 3rd International Conference on Learning Representations*.

Lei, H.-Y.; Tan, Z.-H.; and Zhou, Z.-H. 2024. On the ability of developers' training data preservation of learnware. In *Advances in Neural Information Processing Systems 37*, 36471–36513.

Liu, J.-D.; Tan, Z.-H.; and Zhou, Z.-H. 2024. Towards making learnware specification and market evolvable. In *Proceedings of the 38th AAAI Conference on Artificial Intelligence*, 13909–13917.

Liu, J.-D.; Tan, Z.-H.; and Zhou, Z.-H. 2025a. Dynamic learnware filtering for efficient learnware identification and system slimming. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 1811–1822.

Liu, J.-D.; Tan, Z.-H.; and Zhou, Z.-H. 2025b. Identifying and Reusing Learnwares across Different Label Spaces. In *Proceedings of the 34th International Joint Conference on Artificial Intelligence (IJCAI)*, 5734–5742.

McElfresh, D.; Khandagale, S.; Valverde, J.; Prasad C, V.; Ramakrishnan, G.; Goldblum, M.; and White, C. 2023. When do neural nets outperform boosted trees on tabular data? In *Advances in Neural Information Processing Systems 36*, 76336–76369.

Mémoli, F. 2011. Gromov-Wasserstein distances and the metric approach to object matching. *Foundations of Computational Mathematics*, 11(4): 417–487.

Schölkopf, B.; and Smola, A. J. 2002. *Learning with kernels: Support vector machines, regularization, optimization, and beyond*. MIT press.

Shimodaira, H. 2000. Improving predictive inference under covariate shift by weighting the log-likelihood function. *Journal of statistical planning and inference*, 90(2): 227–244.

Shwartz-Ziv, R.; and Armon, A. 2022. Tabular data: Deep learning is not all you need. *Information Fusion*, 81: 84–90.

Tan, P.; Liu, H.-T.; Tan, Z.-H.; and Zhou, Z.-H. 2024a. Handling learnwares from heterogeneous feature spaces with explicit label exploitation. In *Advances in Neural Information Processing Systems 37*, 12767–12795.

Tan, P.; Tan, Z.-H.; Jiang, Y.; and Zhou, Z.-H. 2023. Handling learnwares developed from heterogeneous feature spaces without auxiliary data. In *Proceedings of the 32nd International Joint Conference on Artificial Intelligence*, 4235–4243.

Tan, Z.-H.; Liu, J.-D.; Bi, X.-D.; Tan, P.; Zheng, Q.-C.; Liu, H.-T.; Xie, Y.; Zou, X.-C.; Yu, Y.; and Zhou, Z.-H. 2024b. Beimingwu: A learnware dock system. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 5773–5782.

Tan, Z.-H.; Zhao, Z.-C.; Shi, H.-Y.; Zhang, X.-Y.; Tan, P.; Yu, Y.; and Zhou, Z.-H. 2025. Learnware of Language Models: Specialized Small Language Models Can Do Big. *arXiv:2505.13425*.

Wachowicz, E. 2020. Wharton research data services (WRDS). *Journal of Business & Finance Librarianship*, 25(3-4): 184–187.

Wu, X.-Z.; Xu, W.; Liu, S.; and Zhou, Z.-H. 2023. Model reuse with reduced kernel mean embedding specification. *IEEE Transactions on Knowledge and Data Engineering*, 35(1): 699–710.

Xie, Y.; Tan, Z.-H.; Jiang, Y.; and Zhou, Z.-H. 2023. Identifying helpful learnwares without examining the whole market. In *Proceedings of the 26th European Conference on Artificial Intelligence*, 2752–2759.

Ye, H.-J.; Liu, S.-Y.; and Chao, W.-L. 2025. A closer look at TabPFN v2: Strength, Limitation, and Extension. *arXiv:2502.17361*.

Ye, H.-J.; Yin, H.-H.; and Zhan, D.-C. 2025. Modern neighborhood components analysis: A deep tabular baseline two decades later. In *The 13th International Conference on Learning Representations*.

Zhang, Y.-J.; Yan, Y.-H.; Zhao, P.; and Zhou, Z.-H. 2021. Towards enabling learnware to handle unseen jobs. In *Proceedings of the 35th AAAI Conference on Artificial Intelligence*, 10964–10972.

Zhou, Z.-H. 2016. Learnware: on the future of machine learning. *Frontiers of Computer Science*, 10: 589–590.

Zhou, Z.-H.; and Tan, Z.-H. 2024. Learnware: Small models do big. *Science China Information Sciences*, 67(1): 112102.

Zhu, B.; Shi, X.; Erickson, N.; Li, M.; Karypis, G.; and Shoaran, M. 2023. XTab: Cross-table pretraining for tabular transformers. In *Proceedings of 40th International Conference on Machine Learning*, 43181–43204.