Towards Making Learnware Specification and Market Evolvable

Jian-Dong Liu, Zhi-Hao Tan, Zhi-Hua Zhou

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

Abstract

The *learnware paradigm* aims to establish a market of numerous well-performed machine learning models, enabling users to leverage existing helpful models for their tasks instead of starting from scratch. Each learnware in the market is a model submitted by its developer, associated with a specification generated with the help of learnware market, representing the model's specialty and utility and enabling it to be identified for new user tasks. As the market continuously scales up, accommodating an ever-increasing number of learnwares, the critical challenge of the learnware paradigm is to effectively and efficiently identify the most helpful learnware(s) for a new user task without accessing the user's raw data. In this paper, to achieve increasingly accurate learnware characterization and identification along with a growing number of learnwares in the market, we propose an approach called Evolvable Learnware Specification with Index (ELSI). Specifically, based on the key idea of leveraging the task information within learnware specifications, we tackle the challenge of ascertaining the capabilities of models beyond their original training tasks, thereby enabling learnware specifications and the entire market to evolve continuously. Furthermore, through organizing learnwares and constructing specification indexes, we design a practical procedure to accurately and efficiently identify helpful learnwares without examining the entire market. Theoretical analysis and extensive experiments on a learnware market prototype encompassing thousands of models and covering six real-world scenarios validate the effectiveness and efficiency of our approach.

1 Introduction

Machine learning has achieved significant success in various real-world applications, including medicine, robotics, and ecology. However, developing a well-performed model necessitates several essential conditions, such as sufficient labeled data, adequate computational resources, and proficient training skills. Without these conditions, most ordinary users can hardly produce high-quality models starting from scratch. Besides, it is difficult to identify and reuse beneficial models among different users due to data privacy concerns.

To tackle the above issues simultaneously, the *learnware paradigm* (Zhou 2016; Zhou and Tan 2024) was proposed to establish a learnware market containing numerous machine learning models, enabling users to build models by reusing existing efforts instead of starting from scratch. A learnware is a well-performed model with a specification representing its specialty and utility, enabling the model to be adequately identified for subsequent user tasks. Developers can spontaneously submit their models on various tasks to the market, which then assigns specifications to accepted ones. The *learnware market* is also called *learnware dock system*, which can organize and utilize the accepted learnwares to solve new user tasks. Given a new task, the market can identify helpful learnwares based on the submitted user requirements. It is important to note that the learnware market has access to the raw data of neither the developers nor the users.

A critical challenge in the learnware paradigm is *how to effectively and efficiently identify the most helpful learnware(s) in a continuously expanding market for a new user task without accessing raw user data?* This is particularly crucial as the number of learnwares grow exponentially. The key to the solution is the *specification*, which is the core of the learnware paradigm, playing a crucial role in learnware characterization and identification without leaking raw data. Based on specifications, recent studies about the learnware paradigm (Wu et al. 2023; Zhang et al. 2021; Tan et al. 2022, 2023; Xie et al. 2023) successfully identified helpful models whose original training distribution align with the user task. However, perfect matches are uncommon due to varying environments. Solely relying on initial training distribution for model identification would ignore numerous beneficial models trained on tasks far from the user's. Additionally, current methods necessitate examining all learnwares in the market, which is computationally infeasible in markets with continuously growing scale. Hence, two specific key issues arise:

- *How to characterize model abilities beyond models' original tasks for accurate learnware identification?*
- *How to avoid examining the entire market for efficient learnware identification?*

The fundamental difficulties of these issues stem mainly from the absence of labeled data for evaluating model performance in various domains, as well as the lack of an effective learnware distance metric to organize models for efficient retrieval. Under the learnware paradigm, our pivotal insight emerges: evaluating a model's capability beyond its initial training task could be feasible through the task infor-

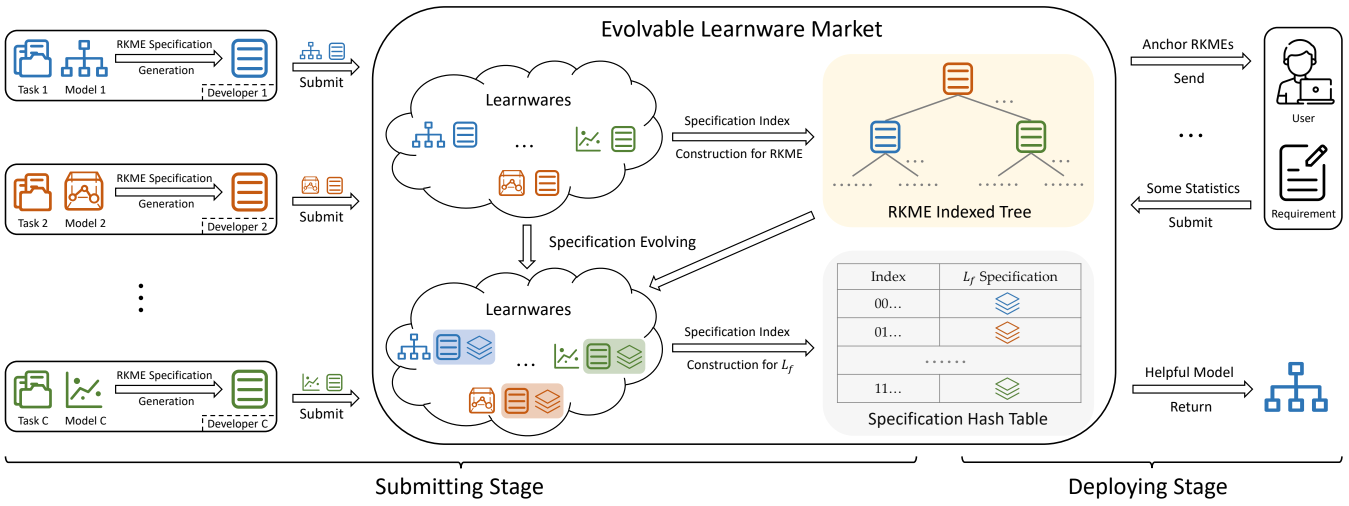

Figure 1: In our approach ELSI, the market evolves specifications by organizing learnwares from developers and building specification indexes in the submitting stage. In the deploying stage, these indexed specifications facilitate interactive approximation of user tasks and efficient identification of valuable models without accessing raw user data or traversing the entire market.

mation within the recently proposed Reduced Kernel Mean Embedding (RKME) specifications (Zhou and Tan 2024) of other learnwares. Furthermore, since specifications characterize model capabilities, they provide a pathway to systematically organize models for efficient retrieval.

In this paper, to tackle the two issues simultaneously, we propose an approach that continuously evolves indexed learnware specifications and the entire market for accurate and efficient learnware identification. As illustrated in Figure 1, the market generates evolvable specifications with indexes in the submitting stage, which are utilized to interactively approximate user tasks and effectively identify beneficial learnwares for users without accessing raw user data or examining the entire market in the deploying stage. We summarize the main contributions of our work as follows:

- We make the first attempt to establish evolvable learnware specifications, aiming for increasingly accurate characterization of model abilities beyond their original training tasks as the market continuously grows, thereby constantly facilitating the evolution and enhancement of the overall market capability. Theoretical analysis for learnware identification is provided.
- Through organizing learnwares and constructing specification indexes, we propose an approach called Evolvable Learnware Specification with Index (ELSI), which could achieve evolvable learnware specifications and corresponding efficient learnware identification for users without leaking raw data. As the key components of our approach, specification indexes are established based on the *RKME indexed tree* and the *specification hash table*.
- We design a practical and efficient procedure for implementing our approach ELSI. Extensive experimental results on a learnware market encompassing thousands of models and covering six real-world scenarios validate the effectiveness and efficiency of our approach.

Organization. The rest is structured as follows. Section 2 introduces preliminaries and Section 3 presents the learnware paradigm formulation. Section 4 details our approach ELSI. Section 5 reports the experiments. Section 6 concludes the paper.

2 Preliminary

This section introduces related techniques within this paper.

Kernel Mean Embedding. KME (Smola et al. 2007) is powerful to map a probability distribution to a point in reproducing kernel Hilbert space (RKHS). Let P be a probability distribution defined over $\mathcal{X}$ and $k : \mathcal{X} \times \mathcal{X} \mapsto \mathbb{R}$ be a kernel function. Assuming $\int_{\mathcal{X}} \sqrt{k(\boldsymbol{x}, \boldsymbol{x})} \mathrm{d}P(\boldsymbol{x}) < \infty$, the KME is $\mu_P = \int_{\mathcal{X}} k(\boldsymbol{x}, \cdot) \mathrm{d}P(\boldsymbol{x})$ with RKHS $\mathcal{H}_k$. KME captures all necessary information about the distribution P with characteristic kernels (Sriperumbudur, Fukumizu, and Lanckriet 2011), such as the Gaussian kernel. In practice, due to the inaccessibility of the true distribution P, we often approximate μ_P by the empirical KME $\widehat{\mu}_P = \frac{1}{m} \sum_{i=1}^{m} k(\boldsymbol{x}_i, \cdot)$, where the samples $\{\boldsymbol{x}_i\}_{i=1}^{m}$ are drawn i.i.d. from P. Under mild conditions, $\widehat{\mu}_P$ converges to μ_P at rate $O(1/\sqrt{m})$, measured by the RKHS norm $\|\cdot\|_{\mathcal{H}_k}$ (Smola et al. 2007).

Reduced Kernel Mean Embedding. Although KME has several beneficial properties, it requires access to raw data, conflicting with the principle of data privacy in the learnware paradigm. Based on KME, the Reduced Kernel Mean Embedding (RKME) is proposed as the specification (Zhou and Tan 2024), which preserves the abilities of KME by concisely representing the model's training data distribution without exposure of raw data. Specifically, the RKME generates a reduced set $\{(\beta_j \in \mathbb{R}, \boldsymbol{z}_j \in \mathcal{X})\}_{j=1}^{n}$ to approximate the empirical KME $\widehat{\mu}_P$ by solving

$$\min_{\boldsymbol{\beta}, \boldsymbol{z}} \left\| \frac{1}{m} \sum\nolimits_{i=1}^{m} k(\boldsymbol{x}_i, \cdot) - \sum\nolimits_{j=1}^{n} \beta_j k(\boldsymbol{z}_j, \cdot) \right\|_{\mathcal{H}_k}^2 . \quad (1)$$

Based on RKME specifications, recent research has succeeded in identifying helpful learnwares by measuring distribution similarity between RKME specifications and user tasks. Additionally, the RKME $\widetilde{\mu}_P = \sum_{j=1}^{n} \beta_j k(\boldsymbol{z}_j, \cdot)$ exhibits a linear convergence rate of $O(e^{-n})$ towards $\widehat{\mu}_P$ when $\mathcal{H}_k$ is finite-dimensional (Bach, Lacoste-Julien, and Obozinski 2012; Zhang et al. 2021).

3 Formulation

The learnware paradigm consists of two distinct stages: submitting and deploying stages. Let $\Delta^p = \{\boldsymbol{\alpha} = (\alpha_1; \ldots; \alpha_p) : \sum_{k=1}^{p} \alpha_k = 1, \alpha_k \geq 0\}$ denote the p-dimensional simplex within $\mathbb{R}^p$.

Submitting Stage. In this stage, developers can spontaneously submit their well-performed models with RKME specifications. Suppose there are C developers in this stage. The c-th developer has access to a private local dataset $D_c = \{(\boldsymbol{x}_{c,i}, y_{c,i})\}_{i=1}^{m_c}$, representing her specific task. In D_c, $\boldsymbol{x}_{c,i}$ is sampled from a distribution $\mathcal{D}_c$ on the input space $\mathcal{X}$ and $y_{c,i}$ is determined by a ground-truth function $h_c \in \mathcal{F} = \{f \mid f : \mathcal{X} \mapsto \mathcal{Y}\}$ of the distribution $\mathcal{D}_c$, i.e.,

$$\forall (\boldsymbol{x}, y) \in D_c, \boldsymbol{x} \sim \mathcal{D}_c, y = h_c(\boldsymbol{x}). \tag{2}$$

Based on the local dataset D_c, the c-th developer can train a well-performed model $f_c \in \mathcal{F}$ for the c-th task and the model f_c holds the following property:

$$\mathcal{L}_{\mathcal{D}_c}(f_c, h_c) = \mathbb{E}_{\boldsymbol{x} \sim \mathcal{D}_c}[\ell(f_c(\boldsymbol{x}), h_c(\boldsymbol{x}))] \leq \varepsilon, \tag{3}$$

where $\ell : \mathcal{Y} \times \mathcal{Y} \mapsto \mathbb{R}$ is the loss function. Besides, to approximate the c-th task without exposing the private local dataset D_c, a RKME specification $R_c = \{(\beta_{c,j}, \boldsymbol{z}_{c,j})\}_{j=1}^{n_c}$ is constructed by solving Eq. (1), where $\boldsymbol{\beta}_c \in \Delta^{n_c}$, $\boldsymbol{z}_{c,j} \in \mathcal{X}$, and $n_c \ll m_c$. Then the developer submits the learnware (f_c, R_c) to the market. After receiving a number of learnwares, the learnware market will organize them, facilitating accurate and efficient identification of helpful learnwares for users in the future. The detailed generation of RKME specifications can be found in a longer version of this paper.

Deploying Stage. In this stage, the user hopes to receive some helpful models from the market to handle her task while ensuring data security, i.e., without disclosing her raw data. Specifically, the user expects to obtain models $\{f_{U_c}\}_{c=1}^{C_u}$ from the learnware market. These models will enable the user to predict her dataset $D_u = \{\boldsymbol{x}_{u,i}\}_{i=1}^{m_u}$, which contains unknown labels and is sampled from a distribution $\mathcal{D}_u$ on $\mathcal{X}$. The ground-truth function of the distribution $\mathcal{D}_u$ is denoted as $h_u \in \mathcal{F}$. In this study, for simplicity, we consider the single model case, i.e., $C_u = 1$. Thus, we focus on the identification for a model f_u that is helpful for the user task within the learnware market, i.e.,

$$\begin{aligned} f_u = & \mathop{\arg\min}_{f \in \{f_c\}_{c=1}^{C}} \mathcal{L}_{\mathcal{D}_u}(f, h_u) \\ = & \mathop{\arg\min}_{f \in \{f_c\}_{c=1}^{C}} \mathbb{E}_{\boldsymbol{x} \sim \mathcal{D}_u}[\ell(f(\boldsymbol{x}), h_u(\boldsymbol{x}))]. \end{aligned} \tag{4}$$

The single model case combined with some existing ensemble techniques (Zhou 2012) can be directly extended to the multiple model case, i.e., $C_u > 1$, and more details will be considered in the future.

4 Our Approach

In this section, we begin by offering a high-level overview of our approach ELSI in the first two subsections, with Section 4.1 presenting the evolvable design for precise learnware characterization and identification, while Section 4.2 introducing specification indexes for accelerating the design and making it feasible. Subsequently, we delve into the detailed implementation of ELSI in Section 4.3.

4.1 Evolvable Learnware Specification and Learnware Identification

Learnware specifications represent model capabilities, forming the foundation of the learnware paradigm. When characterizing a model's ability, it is common to start from natural language descriptions or its training data distribution. However, since the model is essentially a function in a functional space, its true capability extends beyond the constraints imposed by ambiguous descriptions or restrictive data distributions. Thus, the pivotal issue emerges: *how to design specifications that precisely assess a model's capability, especially in aspects extending beyond its initial training tasks?*

The mentioned issue is a fundamental challenge due to the lack of labeled data for evaluating models in various domains. Nevertheless, in the learnware paradigm, task information from RKME specifications in other learnwares may help address this challenge. As the market grows, leveraging these diverse tasks enables continuous evolution of learnware specifications for improved learnware characterization.

In the learnware paradigm, the market rigorously evaluates submitted learnwares, admitting only high-quality ones that excel in their intended tasks. Consequently, as a starting point, it is reasonable to consider the output of each model on its corresponding RKME specification as ground truth. This enables the market to estimate model performance across existing RKME specifications, facilitating continuous evolution of learnware characterization. Specifically, the market can assign a new specification $L_f \in \mathbb{R}^C$ to each model f, where $L_{f,c}$ represents the loss of the model f on the c-th RKME specification, i.e.,

$$\forall c \in [C], L_{f,c} = \sum_{j=1}^{n_c} \beta_{c,j} \ell(f(\boldsymbol{z}_{c,j}), f_c(\boldsymbol{z}_{c,j})). \tag{5}$$

RKME specifications and L_f together constitute the evolvable learnware specification. As the market scales up, the model information captured in evolvable learnware specifications will continuously increase, bringing about enhanced accuracy in learnware characterization and identification.

For the evolvable learnware specification, we propose a new learnware identification method that evaluates model performance on user tasks, denoted as $\mathcal{L}_{\mathcal{D}_u}(f, h_u)$ in Eq. (4). Estimating this is challenging due to the fact that the models do not have access to the raw user data and the unquantifiable nature of generalization loss $\mathcal{L}_{\mathcal{D}_u}(f, h_u)$. Our main idea is to leverage all RKME specifications in the market as a bridge between user tasks and learnwares to approximate the former and subsequently evaluate the latter with a quantifiable surrogate loss. The corresponding theoretical guarantee is presented in Theorem 4.1, and its complete proof is presented in a longer version of this paper.

Theorem 4.1. *Assume $\sup_{\boldsymbol{x}\in\mathcal{X}} k(\boldsymbol{x},\boldsymbol{x}) \leq B_{\mathcal{H}}$, the loss function ℓ obeys the triangle inequality, and for all $c \in [C]$, the local dataset size $m_c = m$, the RKME specification size $n_c = n$, and the submitted model f_c satisfies Eq.* (3). *Let $\ell_{f,f'} : \boldsymbol{x} \mapsto \ell(f(\boldsymbol{x}), f'(\boldsymbol{x})) \in \mathcal{H}_k$, and suppose $\|\ell_{f,f'}\|_{\mathcal{H}_k} \leq U, \forall f, f' \in \mathcal{F}$. Then, with probability at least $1-\delta, \delta \in (0,1)$, for all $\boldsymbol{w} \in \Delta^C$ and $f \in \mathcal{F}$, we have:*

$$\mathcal{L}_{\mathcal{D}_u}(f, h_u) \leq \boldsymbol{w}^\top L_f + U \left\| \widehat{\mu}_{\mathcal{D}_u} - \sum\nolimits_{c=1}^{C} w_c \widetilde{\mu}_{\mathcal{D}_c} \right\|_{\mathcal{H}_k} + \widehat{\varepsilon} + O\left(m^{-\frac{1}{2}} + n^{-\frac{1}{2}} + m_u^{-\frac{1}{2}}\right),$$

where $\widetilde{\mu}_{\mathcal{D}_c} = \sum_{j=1}^{n_c} \beta_{c,j} k(\boldsymbol{z}_{c,j}, \cdot)$ denotes the c-th RKME, $\widehat{\mu}_{\mathcal{D}_u} = \sum_{i=1}^{m_u} k(\boldsymbol{x}_{u,i}, \cdot)/m_u$ is the empirical KME of the user data, and $\widehat{\varepsilon} = \varepsilon + \max_{c\in[C]} \mathcal{L}_{\mathcal{D}_c}(h_c, h_u)$.

Theorem 4.1 presents the upper bound of the model generalization loss on the user task, where $\boldsymbol{w}$ is the variable to be optimized. To obtain a tighter bound, we should seek a value of $\boldsymbol{w}$ by minimizing $\boldsymbol{w}^\top L_f + U\zeta_{\boldsymbol{w}}$, where $\zeta_{\boldsymbol{w}}$ is defined as $\left\| \widehat{\mu}_{\mathcal{D}_u} - \sum_{c=1}^{C} w_c \widetilde{\mu}_{\mathcal{D}_c} \right\|_{\mathcal{H}_k}$. However, this optimization is challenging due to the unavailability of L_f to the user and $\widehat{\mu}_{\mathcal{D}_u}$ to the learnware market.

To tackle these challenges, we propose a two-step approach. In the first step, the user obtains $\boldsymbol{w}_u$ by solving $\min_{\boldsymbol{w}\in\Delta^M} \zeta_{\boldsymbol{w}}$. In the second step, the learnware market calculates $\boldsymbol{w}_u^\top L_f$ to estimate the model loss over the user task for arbitrary model $f \in \mathcal{F}$. This estimation accurately measures model performance on user tasks, enabling the application of models on tasks that extend beyond original purposes. Moreover, we can substitute the generalization error in Eq. (4) with the estimated model performance, i.e.,

$$f_u = \mathop{\arg\min}_{f\in\{f_c\}_{c=1}^C} \boldsymbol{w}_u^\top L_f, \tag{6}$$

which enables the market to select the optimal model for user tasks based on its estimated performance.

4.2 Making the Design Feasible: Learnware Specification Index

In the learnware paradigm, evolvable learnware specifications facilitate increasing accuracy in learnware characterization and identification. However, achieving evolvable learnware specifications is infeasible without systematic organization of learnwares. As the market scale expands, the designed identification method presented in Section 4.1 will face significant efficiency challenges stemming from two critical factors: *the increasing dimensions of L_f and $\boldsymbol{w}_u$ which equal the number of learnwares in the market, as well as the inefficiency of traversing the entire market in Eq.* (6).

To address the above efficiency issues, we propose a general concept: *learnware specification index*, which is used to systematically organize learnware specifications and effectively accelerate various operations related to specifications and learnwares. Various data structures and algorithms can be applied to build specification indexes, such as Binary Indexed Trees, B$^+$-Trees, and hash tables. For different scenarios, selecting and designing an appropriate implementation for specification indexes must consider several factors, including market scale, specification types, and learnware identification methods.

In this paper, we construct specification indexes for RKME and L_f specifications, respectively. The overall idea is as follows: first, we establish a tree-like structure to build indexes for RKME specifications, enabling sparse representations of L_f and $\boldsymbol{w}_u$ and addressing the challenge posed by high dimensionality. Second, we build hash tables to implement L_f specification indexes, facilitating efficient retrieval during learnware identification without examining the entire learnware market.

Indexes for RKME Specifications. After receiving C learnwares $\{f_c, R_c\}_{c=1}^C$ from developers, we can organize all RKME specifications into a hierarchy called *RKME indexed tree* based on their distance metric defined with associated Reproducing Kernel Hilbert Spaces (RKHS) $\mathcal{H}_k$, i.e.,

$$d_{\mathcal{H}_k}(R_i, R_j) = \left\| \widetilde{\mu}_{\mathcal{D}_i} - \widetilde{\mu}_{\mathcal{D}_j} \right\|_{\mathcal{H}_k}^2, \tag{7}$$

where $\widetilde{\mu}_{\mathcal{D}_i}$ and $\widetilde{\mu}_{\mathcal{D}_j}$ are defined the same as $\widetilde{\mu}_{\mathcal{D}_c}$ in Theorem 4.1. Utilizing the distance metric $d_{\mathcal{H}_k}$, indexes for RKME specifications can be constructed through divisive hierarchical clustering, relying on k-medoids clustering (Kaufman and Rousseeuw 1990) as a fundamental technique. The latter involves identifying k representative objects from a set by dividing it into k clusters and choosing an object with the smallest average dissimilarity to others in each cluster. The k-medoids problem is known to be NP-hard, and thus, heuristic algorithms such as CLARANS (Ng and Han 2002) and FasterPAM (Schubert and Rousseeuw 2021) are commonly employed to provide efficient solutions. These indexes accelerate the localization of RKME specifications, enabling the sparse representations of L_f and $\boldsymbol{w}_u$, since we focus solely on the dimensions with lower losses in L_f and higher weights in $\boldsymbol{w}_u$.

Indexes for L_f Specifications. Avoiding traversing the entire market when solving Eq. (6), we use the well-known hashing technique Locality Sensitive Hashing (LSH) (Indyk and Motwani 1998) to construct indexes for L_f specifications. Specifically, inspired by Neyshabur and Srebro (2015), we define two vector transformations $p(\boldsymbol{w}_u) : \mathbb{R}^C \mapsto \mathbb{R}^{C+1}$ and $q(L_f) : \mathbb{R}^C \mapsto \mathbb{R}^{C+1}$ under the assumption of $0 \leq L_{f,c} \leq U_r, \forall f \in \{f_c\}_{c=1}^C, \forall c \in [C]$, i.e.,

$$p(\boldsymbol{w}_u) = (\boldsymbol{w}_u/\|\boldsymbol{w}_u\|_2\, ; 0)$$
$$q(L_f) = \left(U_r - 2L_f ; 2\sqrt{U_r\|L_f\|_1 - \|L_f\|_2^2}\right) \Big/ \sqrt{C}U_r\ . \tag{8}$$

Observing that $\|p(\boldsymbol{w}_u)\|_2 = \|q(L_f)\|_2 = 1$, we can convert the minimum inner product search between $\boldsymbol{w}_u$ and L_f into maximum cosine similarity search between $p(\boldsymbol{w}_u)$ and $q(L_f)$, as follows:

$$f_u = \mathop{\arg\min}_{f\in\{f_c\}_{c=1}^C} \boldsymbol{w}_u^\top L_f = \mathop{\arg\max}_{f\in\{f_c\}_{c=1}^C} p(\boldsymbol{w}_u)^\top q(L_f). \tag{9}$$

To enable effective cosine similarity search, we introduce a hash function $h_{\boldsymbol{a}}^{srp} : \mathbb{R}^{C+1} \mapsto \{0,1\}$ from a famous LSH family known as "sign random projections" (Goemans and

Algorithm 1: RKMEIndexConstruction($\mathcal{S}$, l, r, s)

```
Input: Set of RKME Specifications $\mathcal{S}$, depth in the tree $l$, constants $r, s$.
Output: RKME indexed tree $\mathcal{T}$.
 1: Initialize $\mathcal{T}.subtrees \leftarrow \{\}$, $\mathcal{T}.size = |\mathcal{S}|$, $\mathcal{T}.depth = l$, $\mathcal{T}.\boldsymbol{a} \leftarrow (a_1; \dots; a_r)$ and $\mathcal{T}.\boldsymbol{b} \leftarrow \mathcal{T}.\boldsymbol{a}$, where $a_i$ is a random value sampled from a standard normal distribution for all $i \in [r]$.
 2: if $|\mathcal{S}| \neq 0$ then
 3:     $ns \leftarrow \min(|\mathcal{S}|, s)$.    ▷ Number of subtrees.
 4:     Divide $\mathcal{S}$ into subsets $\{\mathcal{S}_i\}_{i=1}^{ns}$ with the corresponding anchor RKME specifications $\{R_{I_i}\}_{i=1}^{ns}$ by running $k$-medoids on $\mathcal{S}$ based on the distance metric $d_{\mathcal{H}_k}$ defined in Eq. (7).
 5:     for $i = 1$ to $ns$ do
 6:         $\mathcal{T}_i \leftarrow$ RKMEIndexConstruction($\mathcal{S}_i \setminus \{R_{I_i}\}, l+1, r, s$).    ▷ Generate the subtree.
 7:         $\mathcal{T}_i.anchor \leftarrow R_{I_i}$.    ▷ Set the anchor RKME specification for the subtree.
 8:         $\mathcal{T}_i.size \leftarrow \mathcal{T}_i.size + 1$.
 9:         $\mathcal{T}.\boldsymbol{b} \leftarrow \mathcal{T}.\boldsymbol{b} + \mathcal{T}_i.\boldsymbol{b}$.
10:         $\mathcal{T}.subtrees \leftarrow \mathcal{T}.subtrees \cup \{\mathcal{T}_i\}$.
11:     end for
12: end if
```

Williamson 1995; Charikar 2002). This hash function is defined as $h_{\boldsymbol{a}}^{srp}(\boldsymbol{x}) = \text{sign}(\boldsymbol{a}^\top \boldsymbol{x})$, where $\boldsymbol{a}$ is a random vector sampled from a normal distribution $\mathcal{N}(0, I)$. We show that the probability of vectors $\boldsymbol{w}_u$ and L_f having the same encoding increases with their cosine similarity increasing, i.e.,

$$\begin{aligned} &\mathbb{P}\left[h_{\boldsymbol{a}}^{srp}(p(\boldsymbol{w}_u)) = h_{\boldsymbol{a}}^{srp}(q(L_f))\right] \\ &= 1 - \cos^{-1}\left(p(\boldsymbol{w}_u)^\top q(L_f)\right)/\pi \ . \end{aligned} \tag{10}$$

Thus, we employ r random vectors $\{\boldsymbol{a}_i\}_{i=1}^r (\forall i \in [r], \boldsymbol{a}_i \sim \mathcal{N}(0, I))$ to encode $p(\boldsymbol{w}_u)$ and $q(L_f)$ as indexes. Evolved specifications with their indexes constitute the *specification hash table*, which enables efficient learnware identification through employing the classic hash lookup technique, multi-index hashing (Norouzi, Punjani, and Fleet 2014).

4.3 Detailed Procedure of ELSI

The detailed procedure of our approach ELSI comprises four main components: *RKME indexed tree* construction, L_f specifications generation and indexing, user task representation and encoding, and learnware identification.

RKME Indexed Tree Construction. To simultaneously accomplish the generation and indexing of L_f specifications, we assign an r-dimensional random vector $\mathcal{T}.\boldsymbol{a} \in \mathbb{R}^r$ to each subtree $\mathcal{T}$ in the dendrogram of RKME specifications. The sum of the random vectors $\mathcal{T}_i.\boldsymbol{a}$ for each subtree $\mathcal{T}_i$ of $\mathcal{T}$ is computed in a one-pass manner and denoted as $\mathcal{T}.\boldsymbol{b}$, which will be used to accelerate the L_f specifications indexing process. The detailed construction process is summarized in Algorithm 1.

L_f Specifications Generation and Indexing. *RKME indexed tree* can be utilized to generate L_f specifications and simultaneously hash them into r-dimensional binary codes

Algorithm 2: L_f Specifications Generation and Indexing

```
Input: Model $f$, RKME indexed tree $\mathcal{T}$, constants $U_l, U_r, \Delta_U$.
Output: Specification $L_f$ with its index $B_f$.
 1: Initialize $B_f \leftarrow (0; \dots; 0) \in \mathbb{R}^r$, $\Pi_f \leftarrow \mathcal{T}.subtrees$, $o_1 \leftarrow 0$ and $o_2 \leftarrow 0$.
 2: while $|\Pi_f| > 0$ do
 3:     Let $\mathcal{T}_i$ be the first element in $\Pi_f$, whose anchor RKME specification $\mathcal{T}_i.anchor = R_c$.
 4:     $\Pi_f \leftarrow \Pi_f \setminus \{\mathcal{T}_i\}$.
 5:     $L_{f,c} \leftarrow \sum_{j=1}^{n_c} \beta_{c,j} \ell(f(\boldsymbol{z}_{c,j}), f_c(\boldsymbol{z}_{c,j}))$.
 6:     $\widehat{L}_{f,c} \leftarrow \min(L_{f,c}, U_r)$.
 7:     $\delta_i \leftarrow \max(U_l, U_r - (\mathcal{T}_i.depth - 1)\Delta_U)$.
 8:     if $L_{f,c} \leq \delta_i$ then
 9:         $B_f \leftarrow B_f + \widehat{L}_{f,c} \mathcal{T}_i.\boldsymbol{a}$.
10:         $\Pi_f \leftarrow \Pi_f \cup \mathcal{T}_i.subtrees$.    ▷ Continue traversal.
11:         Let $o_1 \leftarrow o_1 + \widehat{L}_{f,c}$ and $o_2 \leftarrow o_2 + \widehat{L}_{f,c} \times \widehat{L}_{f,c}$.
12:     else
13:         $B_f \leftarrow B_f + \widehat{L}_{f,c} \mathcal{T}_i.\boldsymbol{b}$.
14:         $o_1 \leftarrow o_1 + \mathcal{T}_i.size \times \widehat{L}_{f,c}$.
15:         $o_2 \leftarrow o_2 + \mathcal{T}_i.size \times \widehat{L}_{f,c} \times \widehat{L}_{f,c}$.
16:     end if
17: end while
18: $B_f \leftarrow \text{sign}(U_r(\mathcal{T}.\boldsymbol{b} - \mathcal{T}.\boldsymbol{a}) - 2B_f + 2\mathcal{T}.\boldsymbol{a}\sqrt{U_r o_1 - o_2})$.
```

$B_f \in \{0, 1\}^r$ as indexes. We employ threshold parameters (U_l, U_r, Δ_U) to terminate the downward traversal process, leading to a sparse representation of L_f. During the traversal, if the model loss $L_{f,c}$ on the anchor RKME specification $\mathcal{T}_i.anchor = R_c$ exceeds the maximum between U_l and $U_r - (l-1)\Delta_U$ (which decreases with increasing depth) while reaching a subtree $\mathcal{T}_i$ at depth l, the traversal process halts. Then model losses on all subtrees within $\mathcal{T}_i$ are approximated by $L_{f,c}$ instead of accurate calculation. Simultaneously, $q(L_f)$ from Eq. (8) is encoded into binary codes B_f through random vectors $\mathcal{T}.\boldsymbol{a}$. The overall process is illustrated in Algorithm 2, and evolved specifications with binary codes constitute the *specification hash table* to accelerate subsequent learnware identification.

User Task Representation and Encoding. To avoid revealing raw user data, we employ the *RKME indexed tree* to capture user requirements interactively. At the t-th iteration, the learnware market sends a subset of RKME specifications $\{R_{I_i}\}_{i=s_{t-1}}^{s_t}$ as anchors to the user, who computes the sparse vector $\boldsymbol{w}_u^{(t)}$ using all received specifications by solving

$$\min_{\boldsymbol{w} \in \Delta^C, \sum_{i=1}^{s_t} w_{I_i} = 1} \zeta_{\boldsymbol{w}} = \left\| \widehat{\mu}_{\mathcal{D}_u} - \sum\nolimits_{i=1}^{s_t} w_{I_i} \widetilde{\mu}_{\mathcal{D}_{I_i}} \right\|_{\mathcal{H}_k}, \tag{11}$$

and returns $\boldsymbol{w}_u^{(t)}, \zeta_{\boldsymbol{w}_u}^{(t)}$ to the market. Guided by these statistics, the market sends another subset of specifications similar to the dimensions with larger weights in $\boldsymbol{w}_u^{(t)}$ to the user by traversing the *RKME indexed tree* downward one layer. The distance $\zeta_{\boldsymbol{w}}^{(t)}$ becomes smaller as the iteration continues, i.e., $\zeta_{\boldsymbol{w}_u}^{(t)} \leq \zeta_{\boldsymbol{w}_u}^{(t-1)}$. While converting the user task into $\boldsymbol{w}_u$,

Scenario (ℓ)	Ratio (%)	Random	$\mathcal{A}$-distance	RKME-task	ELSI-hash	ELSI-traverse
Postures (Error Rate)	40	52.20 ± 0.87	42.34 ± 1.57	42.44 ± 1.82	○ **33.93** ± **2.15**	○ **33.93** ± **2.15**
	60	51.72 ± 0.58	36.62 ± 1.17	36.34 ± 1.77	○ **27.45** ± **1.93**	○ **27.45** ± **1.93**
	80	51.56 ± 0.60	31.40 ± 0.63	31.16 ± 0.81	○ 22.15 ± 1.48	○ **22.03** ± **1.27**
	100	51.59	23.43	23.43	**11.08**	11.27
Bank (Error Rate)	40	15.53 ± 1.04	15.98 ± 1.70	15.00 ± 0.58	○ 12.41 ± 0.18	○ **12.38** ± **0.19**
	60	15.20 ± 0.59	15.25 ± 0.90	14.32 ± 0.49	○ 11.75 ± 0.35	○ **11.74** ± **0.35**
	80	15.06 ± 0.21	14.93 ± 0.34	14.26 ± 0.51	○ 12.19 ± 0.35	○ **12.17** ± **0.35**
	100	14.83	14.64	14.13	**12.11**	12.31
Mushroom (Error Rate)	40	44.09 ± 0.68	30.64 ± 2.47	30.47 ± 2.22	○ 22.27 ± 2.65	○ **22.22** ± **2.68**
	60	43.94 ± 0.58	26.10 ± 2.15	24.93 ± 2.02	○ **20.38** ± **1.86**	○ 21.20 ± 1.80
	80	43.67 ± 0.46	21.18 ± 1.67	19.76 ± 0.84	○ 15.74 ± 2.18	○ **15.67** ± **2.04**
	100	43.66	16.90	16.29	**6.23**	6.29
PPG-DaLiA (RMSE)	40	37.01 ± 1.19	30.74 ± 1.25	29.51 ± 0.91	○ 17.68 ± 0.44	○ **17.36** ± **0.52**
	60	36.42 ± 1.21	27.48 ± 0.79	26.30 ± 0.59	○ 16.21 ± 0.88	○ **15.17** ± **0.83**
	80	36.38 ± 0.45	23.89 ± 0.62	23.28 ± 0.36	○ 14.65 ± 0.54	○ **12.70** ± **0.35**
	100	36.43	20.62	20.62	13.88	**11.11**
PFS (RMSE)	40	2.46 ± 0.12	2.16 ± 0.10	2.11 ± 0.15	2.03 ± 0.15	**2.00** ± **0.13**
	60	2.52 ± 0.14	2.17 ± 0.10	2.18 ± 0.09	○ **1.98** ± **0.07**	○ 1.99 ± 0.06
	80	2.57 ± 0.06	2.22 ± 0.08	2.18 ± 0.09	○ 2.04 ± 0.15	○ **1.99** ± **0.10**
	100	2.58	2.21	2.21	2.03	**1.97**
M5 (RMSE)	40	3.28 ± 0.35	4.17 ± 1.78	2.33 ± 0.07	○ 2.26 ± 0.09	○ **2.22** ± **0.05**
	60	3.35 ± 0.28	4.80 ± 1.53	2.28 ± 0.06	○ 2.22 ± 0.05	○ **2.19** ± **0.04**
	80	3.28 ± 0.17	4.28 ± 1.30	2.23 ± 0.05	2.21 ± 0.06	○ **2.16** ± **0.05**
	100	3.36	5.25	2.19	**2.14**	**2.14**

Table 1: Losses (ℓ) on user data's ground truth. For all ratios except 100%, the evaluation is repeated ten times, and the results are presented as the mean and standard deviation. The best method is emphasized in bold. "○" indicates the methods that are significantly superior to all contenders by the paired t-test at a 5% significance level.

we simultaneously encode $p(\boldsymbol{w}_u)$ into binary codes B_u for subsequent retrieval. The entire process efficiency is controlled by the number of iterations T_u and the tolerance ϵ, as detailed in a longer version of this paper. Users may also upload RKME specifications of their tasks to simplify the anchor interaction process described above.

Learnware Identification. Based on the sparse vector $\boldsymbol{w}_u$ and its binary codes B_u, we can estimate model performance and use hash techniques to accelerate identification of models that perform well on user tasks. This process involves two steps. First, based on the *specification hash table*, the classical hash technique called multi-index hashing (Norouzi, Punjani, and Fleet 2014) is applied to retrieve K potentially helpful models $\{f_{J_i}\}_{i=1}^K$ from $\{f_c\}_{c=1}^C$ with a minimum Hamming distance between B_u and B_f, which significantly narrows down the search scope and is highly efficient. Then, the market identifies the model f_u from $\{f_{J_i}\}_{i=1}^K$ using estimated model loss $\boldsymbol{w}_u^\top L_{f_{J_i}}$. Additionally, the dimensions in L_f that are approximated in Algorithm 2 but are non-zero in $\boldsymbol{w}_u$ will be precisely computed when calculating the loss $\boldsymbol{w}_u^\top L_f$. The L_f specification with its index B_f will be updated in real-time. With continuous user interactions, both the learnware specifications and the entire market will continuously evolve, leading to improved accuracy in learnware characterization and identification.

5 Experiments

In this section, we have developed a learnware market prototype encompassing thousands of models with different types, spanning various real-world scenarios. We conduct comparisons against several existing methods, validating the efficacy and efficiency of our proposed approach.

5.1 Experimental Setup

Here we introduce experimental settings about a learnware market prototype, evaluation, contenders, and configuration.

Learnware Market Prototype. We have developed a learnware market prototype comprising 3090 models of various model types, each trained on different data sets, covering six real-world scenarios involving classification and regression. These scenarios correlate with several real-world datasets: Postures (Gardner et al. 2014), Bank (Moro, Cortez, and Rita 2014), Mushroom (Wagner, Heider, and Hattab 2021), PPG-DaLiA (Reiss et al. 2019), PFS (Kaggle 2018) and M5 (Makridakis, Spiliotis, and Assimakopoulos 2022). Postures involves hand postures, Bank relates to marketing campaigns of a banking institution, and Mushroom contains different mushrooms. PPG-DaLiA focuses on heart rate estimation, while PFS and M5 concern sales prediction. These datasets span various tasks and scenarios, varying in scale from 55 thousand to 46 million instances. Each dataset

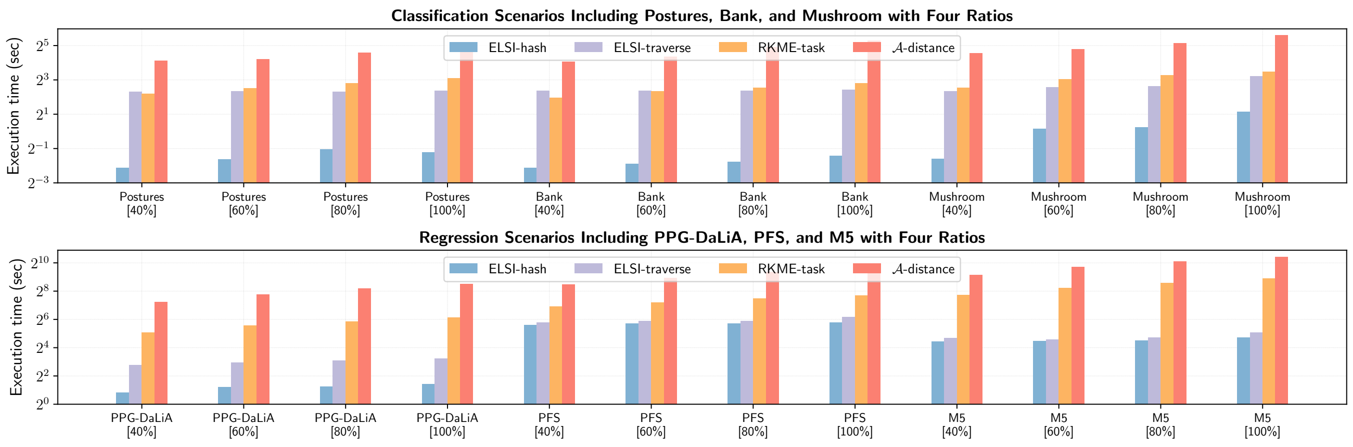

Figure 2: The comparison of full execution time in the deploying stage on six real-world scenarios.

is naturally split into multiple parts with different data distributions based on categorical attributes, and each part is then further subdivided into training and test sets. For each training set, we train various models with different model types, including linear models (Hosmer Jr, Lemeshow, and Sturdivant 2013; Hoerl and Kennard 1970), LightGBM (Ke et al. 2017), and neural networks (Glorot and Bengio 2010) with different hyperparameters. The number of models in each scenario ranges from 165 to 1050. For evaluation, we use each test set as user testing data, which does not appear in any model's training data. The various scenarios, partitions and models ensure that the market encompasses a wide array of tasks and models, significantly enhancing the diversity in the prototype and the authenticity of experimental settings. More details are presented in a longer version of this paper.

Evaluation. We access methods across six scenarios in the prototype. Given a specific scenario with n_u users, we evaluate methods using the metric $\frac{1}{n_u}\sum_{i=1}^{n_u} \text{loss}_i$, where loss_i is the loss of the identified model on the i-th user testing instances. We employ error rate and root-mean-square error (RMSE) as the loss function ℓ for classification and regression scenarios, respectively. To further verify the effectiveness and robustness of our approach, we conduct evaluations at four ratios: 40%, 60%, 80% and 100%. For instance, at a 60% ratio, we randomly select 60% of learnwares within the corresponding scenario for subsequent learnware identification. We repeat the evaluation ten times for all ratios except 100% and record both the mean and standard deviation.

Contenders. We compare our proposed approach with three other methods: a naive baseline *Random* and a related method *RKME-task* (Wu et al. 2023) with its variant $\mathcal{A}$*-distance*. *Random* selects models in a random manner and *RKME-task* identifies the model whose RKME specification is most similar to the user's RKME specification. Different from the latter, $\mathcal{A}$*-distance* chooses the metric $\mathcal{A}$-distance (Ben-David et al. 2006, 2010) to measure the similarity between RKME specifications.

Configuration. For all RKME-based methods, we set the specification size n_c to 100 and use the Gaussian kernel $k(\boldsymbol{x}_1, \boldsymbol{x}_2) = \exp(-\gamma\|\boldsymbol{x}_1 - \boldsymbol{x}_2\|_2^2)$ with $\gamma = 0.1$. In the submitting stage, we set $r = 150$ and select s from $\{3, 4\}$ for different scenarios. In the deploying stage, we set the number of iterations $T_u = 5$ and the constants $s' = s, K = 50$.

5.2 Evaluation on the Learnware Market Encompassing Thousands of Models

Learnware Identification Performance. Table 1 presents the empirical results of our proposed approach, where *ELSI-hash* only estimates the performance of a few learnwares filtered out from all learnwares utilizing hash techniques, and *ELSI-traverse* estimates the performance of all learnwares without pre-filtering. As shown in Table 1, *ELSI-traverse* outperforms other contenders in accuracy for estimating learnware performance. Furthermore, although *ELSI-hash* estimates fewer learnwares, its performance closely matches *ELSI-traverse* and still exceeds other contenders, validating the effectiveness of our learnware identification approach.

Learnware Identification Efficiency. We compare the full execution time of each method in the deploying stage. As shown in Figure 2, *ELSI-hash* achieves the highest efficiency, significantly outperforming *RKME-task* and $\mathcal{A}$*-distance*, and its execution time variation remains insignificant with the change of ratios, reflecting the effectiveness of our learnware identification approach. *ELSI-traverse* outperforms other contenders in most scenarios, indicating that inner product computation of sparse vectors $\boldsymbol{w}_u$ and L_f is efficient. Moreover, the difference between *ELSI-traverse* and *ELSI-hash* increases as the market scales up. For a medium-sized market, *ELSI-traverse* is comparable to *ELSI-hash*.

6 Conclusion

This paper presents ELSI, an approach utilizing the vast amount of learnwares shared by the community developers to solve new user tasks. By organizing these learnwares and constructing specification indexes including the *RKME indexed tree* and the *specification hash table*, ELSI continuously evolves specifications and the entire market, and effectively facilitates the reuse of learnwares beyond their original purposes without disclosing raw data. Theoretical guarantees and empirical evaluations validate ELSI's efficacy.

Acknowledgments

This research was supported by NSFC (62250069) and the Collaborative Innovation Center of Novel Software Technology and Industrialization. The authors would like to thank Peng Tan for helpful discussions. We are also grateful for the anonymous reviewers for their valuable comments.

References

Bach, F. R.; Lacoste-Julien, S.; and Obozinski, G. 2012. On the Equivalence between Herding and Conditional Gradient Algorithms. In *Proceedings of the 29th International Conference on Machine Learning*, 1355–1362.

Ben-David, S.; Blitzer, J.; Crammer, K.; Kulesza, A.; Pereira, F.; and Vaughan, J. W. 2010. A theory of learning from different domains. *Machine Learning*, 79(1-2): 151–175.

Ben-David, S.; Blitzer, J.; Crammer, K.; and Pereira, F. 2006. Analysis of Representations for Domain Adaptation. In *Advances in Neural Information Processing Systems 19*, 137–144.

Charikar, M. 2002. Similarity estimation techniques from rounding algorithms. In *Proceedings on 34th Annual ACM Symposium on Theory of Computing*, 380–388.

Gardner, A.; Kanno, J.; Duncan, C. A.; and Selmic, R. R. 2014. Measuring Distance between Unordered Sets of Different Sizes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 137–143.

Glorot, X.; and Bengio, Y. 2010. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the 13th International Conference on Artificial Intelligence and Statistics*, 249–256.

Goemans, M. X.; and Williamson, D. P. 1995. Improved Approximation Algorithms for Maximum Cut and Satisfiability Problems Using Semidefinite Programming. *Journal of the ACM*, 42(6): 1115–1145.

Hoerl, A. E.; and Kennard, R. W. 1970. Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics*, 12(1): 55–67.

Hosmer Jr, D. W.; Lemeshow, S.; and Sturdivant, R. X. 2013. *Applied logistic regression*. John Wiley & Sons.

Indyk, P.; and Motwani, R. 1998. Approximate Nearest Neighbors: Towards Removing the Curse of Dimensionality. In *Proceedings of the 30th Annual ACM Symposium on the Theory of Computing*, 604–613.

Kaggle. 2018. Predict future sales. https://kaggle.com/competitions/competitive-data-science-predict-future-sales. Accessed: 2023-05-20.

Kaufman, L.; and Rousseeuw, P. J. 1990. *Finding Groups in Data: An Introduction to Cluster Analysis*. John Wiley.

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.

Makridakis, S.; Spiliotis, E.; and Assimakopoulos, V. 2022. The M5 competition: Background, organization, and implementation. *International Journal of Forecasting*, 38(4): 1325–1336.

Moro, S.; Cortez, P.; and Rita, P. 2014. A data-driven approach to predict the success of bank telemarketing. *Decision Support System*, 62: 22–31.

Neyshabur, B.; and Srebro, N. 2015. On Symmetric and Asymmetric LSHs for Inner Product Search. In *Proceedings of the 32nd International Conference on Machine Learning*, 1926–1934.

Ng, R. T.; and Han, J. 2002. CLARANS: A Method for Clustering Objects for Spatial Data Mining. *IEEE Transactions on Knowledge and Data Engineering*, 14(5): 1003–1016.

Norouzi, M.; Punjani, A.; and Fleet, D. J. 2014. Fast Exact Search in Hamming Space With Multi-Index Hashing. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36(6): 1107–1119.

Reiss, A.; Indlekofer, I.; Schmidt, P.; and Laerhoven, K. V. 2019. Deep PPG: Large-Scale Heart Rate Estimation with Convolutional Neural Networks. *Sensors*, 19(14): 3079.

Schubert, E.; and Rousseeuw, P. J. 2021. Fast and eager k-medoids clustering: O(k) runtime improvement of the PAM, CLARA, and CLARANS algorithms. *Information Systems*, 101: 101804.

Smola, A. J.; Gretton, A.; Song, L.; and Schölkopf, B. 2007. A Hilbert Space Embedding for Distributions. In *Proceedings of the 18th International Conference on Algorithmic Learning Theory*, 13–31.

Sriperumbudur, B. K.; Fukumizu, K.; and Lanckriet, G. R. G. 2011. Universality, Characteristic Kernels and RKHS Embedding of Measures. *Journal of Machine Learning Research*, 12(70): 2389–2410.

Tan, P.; Tan, Z.-H.; Jiang, Y.; and Zhou, Z.-H. 2022. Towards enabling learnware to handle heterogeneous feature spaces. *Machine Learning*, 1–22.

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.

Wagner, D.; Heider, D.; and Hattab, G. 2021. Mushroom data creation, curation, and simulation to support classification tasks. *Scientific Reports*, 11(1): 1–12.

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.

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. 2012. *Ensemble Methods: Foundations and Algorithms*. CRC press.

Zhou, Z.-H. 2016. Learnware: on the future of machine learning. *Frontiers of Computer Science*, 10(4): 589–590.

Zhou, Z.-H.; and Tan, Z.-H. 2024. Learnware: Small Models Do Big. *SCIENCE CHINA Information Sciences*, 67(1): 1–12.