Representation Learning for Tabular Data: A Comprehensive Survey

Jun-Peng Jiang, Si-Yang Liu, Hao-Run Cai, Qi-Le Zhou, Han-Jia Ye

Abstract*—Tabular data, structured as rows and columns, is among the most prevalent data types in machine learning classification and regression applications. Models for learning from tabular data have continuously evolved, with Deep Neural Networks (DNNs) recently demonstrating promising results through their capability of representation learning. In this survey, we systematically introduce the field of tabular representation learning, covering the background, challenges, and benchmarks, along with the pros and cons of using DNNs. We organize existing methods into three main categories according to their generalization capabilities: specialized, transferable, and general models. Specialized models focus on tasks where training and evaluation occur within the same data distribution. We introduce a hierarchical taxonomy for specialized models based on the key aspects of tabular data—features, samples, and objectives—and delve into detailed strategies for obtaining high-quality feature- and sample-level representations. Transferable models are pre-trained on one or more datasets and subsequently fine-tuned on downstream tasks, leveraging knowledge acquired from homogeneous or heterogeneous sources, or even cross-modalities such as vision and language. General models, also known as tabular foundation models, extend this concept further, allowing direct application to downstream tasks without additional fine-tuning. We group these general models based on the strategies used to adapt across heterogeneous datasets. Additionally, we explore ensemble methods, which integrate the strengths of multiple tabular models. Finally, we discuss representative extensions of tabular learning, including open-environment tabular machine learning, multimodal learning with tabular data, and tabular understanding tasks. More information can be found in the following repository: https://github.com/LAMDA-Tabular/Tabular-Survey.

***Index Terms*—Tabular Data, Representation Learning, Deep Tabular Learning, Tabular Foundation Model**

I. Introduction

TABULAR data, characterized by structured rows and columns, is one of the most prevalent data formats in real-world machine learning applications, spanning diverse domains such as finance [1], healthcare [2], education [3], recommendation systems [4], and scientific research. In particular, AI for scientific research (AI4science) has increasingly relied on tabular data, as numerous prominent datasets—such as those from genomics [5], chemistry [6], and climate science [7], [8]—naturally adopt tabular forms.

Tabular data inherently organizes information in a *structured, table-like format*. In this survey, we focus primarily on supervised tabular machine learning tasks, specifically classification and regression. Beyond their structured organization, tabular datasets frequently include heterogeneous attributes [9], encompassing numerical, categorical, or mixed data types that may be dense or sparse. Additionally, many tabular datasets present quality challenges, such as noisy measurements, missing values, outliers, inaccuracies [10], and privacy constraints [11], all of which complicate the modeling process. The most common supervised tabular tasks are classification and regression, where the goal is to learn mappings from training data to discrete or continuous targets, respectively. As illustrated in Fig. 1, each row represents an instance (with its corresponding label), while each column corresponds to a specific attribute or feature [12]. Ideally, learned mappings should generalize effectively, accurately predicting outcomes for new instances drawn from the same underlying distribution.

J.-P. Jiang, S.-Y Liu, H.-R Cai, Q.-L Zhou, and H.-J. Ye are with School of Artificial Intelligence, Nanjing University, and National Key Laboratory for Novel Software Technology, Nanjing University, Nanjing, 210023, China. E-mail: {jiangjp,liusy,caihr,zhouql,yehj}@lamda.nju.edu.cn

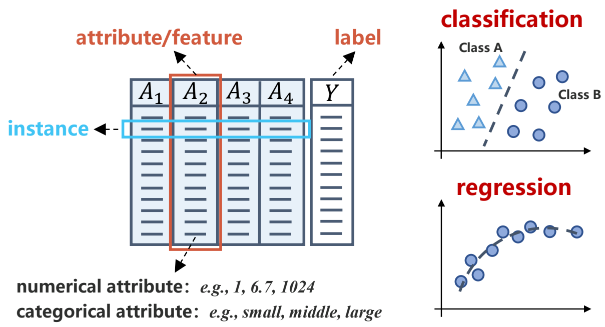
attribute/feature
label
classification
Class A
Class B
A_1 A_2 A_3 A_4 Y
instance
regression
numerical attribute: e.g., 1, 6.7, 1024
categorical attribute: e.g., small, middle, large

Fig. 1. A brief introduction to tabular data and associated learning tasks. Each row represents an instance and each column corresponds to an attribute or feature, which can be numerical or categorical. The most common tabular tasks are classification and regression as shown in the right side of the figure.

Machine learning methods for tabular data have evolved significantly over the years [13], [14], [15]. Recently, the rise of deep learning has profoundly impacted domains like computer vision [16] and natural language processing [17], where Deep Neural Networks (DNNs) extract semantic *representations* directly from raw inputs [18], [19], [20]. These learned representations have not only improved generalization but have also facilitated knowledge transfer across related tasks [21]. The flexibility of DNNs in modeling feature interactions and learning hierarchical structures has inspired interest in adapting deep learning techniques to tabular data.

Indeed, DNNs were applied to tabular data decades ago, initially targeting dimensionality reduction and visualization tasks [22], [23], [24], yet they typically struggled to match tree-based methods on standard classification and regression problems. Later advances in DNNs have led to significant improvements across various tabular-related applications, such as click-through rate prediction [25], anomaly detection [26], recommendation systems [27], and time series forecasting [28].

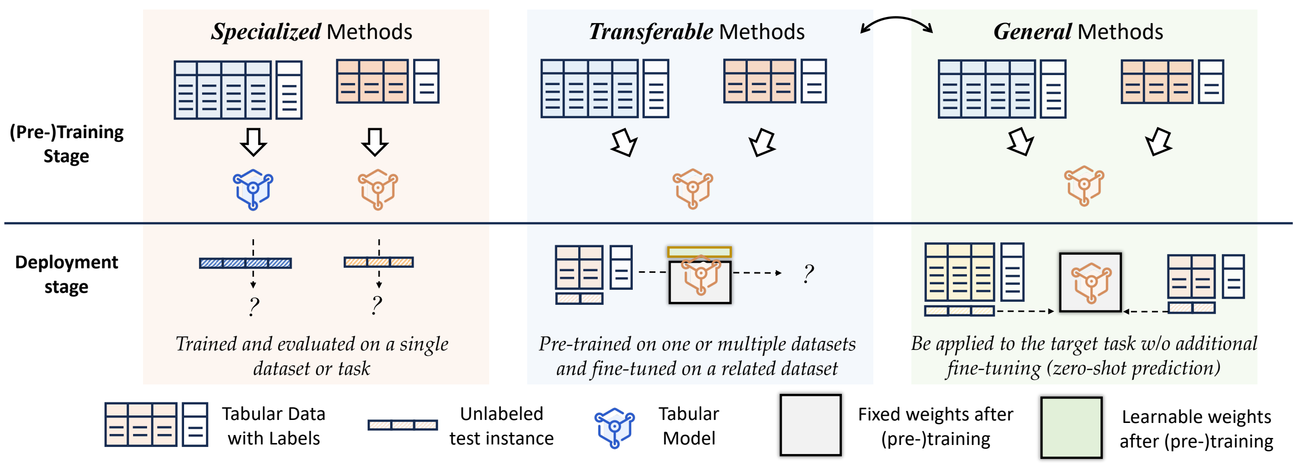
Specialized Methods
Transferable Methods
General Methods
(Pre-)Training Stage
Deployment stage
?
Trained and evaluated on a single dataset or task
Pre-trained on one or multiple datasets and fine-tuned on a related dataset
Be applied to the target task w/o additional fine-tuning (zero-shot prediction)
Tabular Data with Labels
Unlabeled test instance
Tabular Model
Fixed weights after (pre-)training
Learnable weights after (pre-)training

Fig. 2. We organize existing tabular classification/regression methods into three categories according to their generalization capabilities: specialized (left), transferable (middle), and general (right) models. Specialized models focus on tasks where training and evaluation occur within the same data distribution. Transferable models are pre-trained on one or more datasets and subsequently fine-tuned on downstream tasks. General models, also known as tabular foundation models, extend this concept further, allowing direct application to downstream tasks without additional fine-tuning. General models and transferable methods are synergistic: transferable techniques facilitate the training of general models, which in turn serve as powerful pre-trained models for downstream fine-tuning.

Modern deep learning approaches, benefiting from better-designed architectures, optimized training strategies, high-quality representations, have revitalized DNN performance on tabular data, often rivaling or surpassing traditional tree-based models [29], [30], [31]. Given the wide variety of approaches emerging in deep tabular modeling, a systematic overview that revisits critical factors and current methodologies in representation learning for tabular data becomes necessary.

This survey begins by introducing the background of tabular data learning, highlighting the challenges involved and critically examining the advantages and limitations of utilizing DNNs compared to classical—particularly tree-based—methods [32], [33], [34], [35]. Given the observed instability of method performance across different tabular datasets, we also discuss comprehensive strategies for dataset collection, evaluation, and analysis, aiming to establish robust criteria for aggregating performance metrics across multiple datasets [36], [37], [38].

We broadly categorize deep tabular methods into three types: *specialized methods, transferable methods, and general methods*, distinguished by the scope of datasets on which they are trained and deployed, as well as their corresponding generalization capabilities (illustrated in Fig. 2). Specialized tabular methods align closely with classical supervised models, typically trained and evaluated on data drawn from the same distribution. In contrast, transferable methods leverage knowledge from models pre-trained on one or multiple source datasets, subsequently fine-tuning these models on target datasets; the primary challenge here lies in addressing the heterogeneity between pre-trained sources and target tasks. The recently proposed general tabular methods—motivated by the remarkable "zero-shot" generalization abilities demonstrated by large language models (LLMs)—exhibit exceptional versatility. These general models can directly apply to downstream tabular datasets without additional fine-tuning, achieving robust generalization due to advanced pre-training strategies.

Crucially, rather than viewing these categories as a strict hierarchy, we emphasize the *synergistic relationship* between transferable and general methods. On one hand, techniques foundational to transferable models, such as self-supervised learning and heterogeneous feature alignment, serve as essential building blocks for constructing robust general models. On the other hand, general models often function as powerful starting points for transfer learning; fine-tuning a general model (*e.g.*, TabPFN variants [39], [40], [41]) on downstream tasks typically yields superior performance compared to zero-shot inference, effectively blurring the line between general and transferable approaches. Together with specialized methods, which remain dominant on large-scale, distribution-specific tasks, these paradigms form a complementary ecosystem, providing diverse tools tailored to different data scales and computational constraints.

For **specialized methods**, numerous designs have been proposed from diverse perspectives, and previous papers have often categorized these methods based primarily on their architectural characteristics or behaviors. Existing taxonomies [42], for example, group specialized methods into feature-preprocessing-based [29], [43], data-augmentation-based [44], [45], [46], [47], MLP variants [48], [30], specialized DNN architectures [49], [50], [51], [52], [53], [54], [55], [56], tree-mimic approaches [57], [58], [59], token-based techniques [60], [61], [29], [62], [63], regularization-driven methods [64], [65], and neighborhood-based strategies [66], [67], [31]. However, such categorizations can appear scattered, making it difficult to connect the core ideas between methods in distinct groups. In contrast, this survey introduces a hierarchical taxonomy based on the key aspects of tabular data—features, samples, and objectives—providing a cohesive organizational framework. Our approach emphasizes detailed strategies for obtaining high-quality representations at both feature- and sample-levels. This unified perspective helps bridge core ideas across methods, facilitating clearer comparative discussions and potentially guiding the design of more advanced tabular models.

Instead of training a model from scratch on a single tabular dataset, **transferable models** leverage knowledge encoded in a pre-trained model from another dataset, which can significantly enhance the training process, especially when data or computational resources for the target task are limited. A major challenge in transferring knowledge across tabular tasks lies in the inherent heterogeneity between the source and target datasets, particularly differences in their feature and label spaces. In this survey, we adopt a broad perspective on transferable tabular models, categorizing methods based on the sources of their pre-trained knowledge. Specifically, we discuss models pre-trained on homogeneous tabular domains, such as self-supervised methods with additional pre-training steps on the target dataset itself [68], [69]; models pre-trained across heterogeneous tabular domains [70], [71], [62]; and methods transferring knowledge from other modalities, such as vision-based pre-trained models [72], [73]. Additionally, since incorporating attribute semantics (when available) is a common strategy for bridging heterogeneous attribute spaces across tabular datasets [74], [75], [76], we also explore approaches leveraging language models in the final category. In particular, we further organize these language model-based strategies according to the methods to extract knowledge and the types of language models involved—ranging from small-scale language models to Large Language Models (LLMs) [77], [78], [79].

Inspired by recent advancements in foundation models from vision and language domains [80], [81], **general models**—also known as tabular foundation models—expand the concept of transferable tabular models by enabling direct application to downstream tasks without additional fine-tuning. This capability, commonly referred to as the model's "zero-shot" ability, significantly enhances the model's usability across diverse tabular datasets. In contrast to transferable models, which primarily focus on bridging knowledge gaps between source and target datasets, general models aim to construct highly adaptive architectures capable of handling a wide array of heterogeneous datasets simultaneously. We categorize these general models based on the strategies used to achieve adaptiveness across diverse tabular tasks, specifically examining adaptations from both data-centric [82] and model-centric perspectives [83], [84]. Furthermore, we discuss critical branches of general tabular models in detail: the TabPFN variants leveraging in-context learning [85], [86], [87], and methods utilizing attributes and semantics to unify heterogeneous tasks within a common representation framework [88], [89], [90].

Additionally, ensemble methods [91], [39], [87] are introduced, which improve the generalization ability based on the strengths of multiple tabular models. By summarizing the state of the field and discussing extensions, we aim to guide future research and applications in tabular representation learning.

II. Background

This section presents the (supervised) tabular machine learning task, including the notation of tabular data learning, the history of tabular data, the challenges of learning from tabular data, evaluation metrics, and tabular benchmarks.

A. Learning with Tabular Data

A supervised tabular dataset is formatted as N examples and d features/attributes corresponding to N rows and d columns in the table. An instance $\boldsymbol{x}_i \in \mathbb{R}^d$ is depicted by its d feature values. Assume $x_{i,j}$ as the j-th feature of instance $\boldsymbol{x}_i$, it could be a numerical (continuous) one $x_{i,j}^{\text{num}} \in \mathbb{R}$, like the temperature of a region or the density of the object. $\boldsymbol{x}_i$ can also be a categorical (discrete) value $x_{i,j}^{\text{cat}}$, like one of multiple colors, the location of a person, or even some textual descriptions of the instance. Each instance is associated with a label y_i, where $y_i \in \{1, -1\}$ in a binary classification task, $y_i \in [C] = \{1, \ldots, C\}$ in a multi-class classification task, and $y_i \in \mathbb{R}$ in a regression task. This survey primarily focuses on standard classification and regression tasks and does not specifically discuss ordinal regression [92].

Given a tabular dataset $\mathcal{D} = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^N$, we aim to learn a mapping f on $\mathcal{D}$ that maps $\boldsymbol{x}_i$ to its label y_i. In other words, the model predicts $\boldsymbol{x}_i$ with $\hat{y}_i = f(\boldsymbol{x}_i)$. The general objective learning f follows the structural risk minimization:

$$\min_f \sum_{(\boldsymbol{x}_i, y_i) \in \mathcal{D}} \ell(y,\ \hat{y}_i = f(\boldsymbol{x}_i)) + \Omega(f)\ . \tag{1}$$

$\ell(\cdot, \cdot)$ measures the discrepancy between the predicted label $\hat{y}_i$ and the true label y_i, *e.g.*, cross-entropy in classification and mean square error in regression. $\Omega(\cdot)$ is the regularization on the model, which restricts the complexity of f. We expect the learned f is able to extend its ability to *unseen* instances sampled from the same distribution as $\mathcal{D}$.

Tabular methods differ in their strategies to implement f. The "dummy" approach makes predictions based on training labels $\{y_i\}_{i=1}^N$ directly, which outputs the major class in the training set for classification and the average of all labels for regression, respectively. In a C-class classification task, classical parametric methods implement f with a linear mapping, *i.e.*, $f(\boldsymbol{x}_i) = \boldsymbol{W}^\top \boldsymbol{x}_i + \boldsymbol{b}$, where the classifier $\boldsymbol{W} \in \mathbb{R}^{d \times C}$ and $\boldsymbol{b} \in \mathbb{R}^C$ is the bias. With different loss functions, we can implement Logistic Regression, SVM, or even AdaBoost. In contrast, non-parametric methods implement the prediction via $f(\boldsymbol{x}_i) = f(\boldsymbol{x}_i, \mathcal{D})$, depending on the whole training set. For example, KNN searches neighbors in the training set $\mathcal{D}$ with the K smallest distance w.r.t. $\boldsymbol{x}_i$.

Deep tabular methods implement f with a deep neural network. Most deep models could be decomposed into two parts, *i.e.*, $f(\boldsymbol{x}_i) = \boldsymbol{W}^\top \phi(\boldsymbol{x}_i) + \boldsymbol{b}$. Similar to the linear model, $\boldsymbol{W}$ and $\boldsymbol{b}$ are the components of the classifier, with $\boldsymbol{W} \in \mathbb{R}^{d' \times C}$. ϕ maps the input vector $\boldsymbol{x}_i$ into the d' dimension space, which extracts semantic embeddings for the given input. ϕ could be implemented with an MLP or a residual network.

B. Challenges of Learning from Tabular Data

Different from other types of data sources, *e.g.*, images and texts, there exist several challenges dealing with tabular datasets due to their characteristics.

Heterogeneity of Features. Unlike continuous image data or token-based text, tabular data often contains both numerical and categorical attributes, each requiring different handling [9], [93]. Numerical features vary in range and distribution, requiring normalization or scaling. Categorical features differ in cardinality and semantics, needing encoding methods like one-hot vectors or embeddings. Models must handle these mixed types carefully to retain feature utility.

Lack of Spatial Relationships. Tabular data lacks the spatial or sequential structure present in other modalities [72], [48].

Column order has no semantic meaning, making it permutation-invariant. Rows are typically assumed to be independently and identically distributed (*i.i.d.*), eliminating temporal or sequential correlations. This lack of structure limits the applicability of deep architectures designed to exploit such dependencies.

Sensitivity to Perturbations. Unlike images, text, or time series data, tabular data often exhibits sharp decision boundaries where small variations in critical features can lead to significant shifts in the target label [32], [33]. Furthermore, when predicting with LLMs, they often struggle with precise numerical reasoning and are insensitive to small numerical changes, leading to suboptimal performance on tasks requiring high-precision arithmetic or regression [94].

Low-quality and Missing Data. Unlike image or text data, where contextual redundancy helps mitigate missing values, tabular data is more sensitive to incomplete or noisy entries [95], [96]. Missing values can introduce bias and degrade performance, while noisy data reduces reliability. Thus, preprocessing steps like cleaning and imputation are essential.

Importance of Feature Engineering. Tabular models heavily rely on input feature quality [43], [97]. Unlike in vision or NLP, where DNNs learn from raw data, tabular tasks often require domain knowledge and manual feature engineering. Modeling feature interactions usually demands expert-driven transformations, which significantly affect performance [98].

Class Imbalance. Tabular classification tasks often face label imbalance, where some classes are underrepresented [99]. This leads to biased predictions and poor performance on minority classes. Solutions include oversampling, undersampling, and loss reweighting. Metrics like AUC and F1-score help evaluate models under imbalance. Recent studies show deep and classical models handle imbalance differently, warranting careful method selection [100], [37].

Scalability to Large Datasets. Tabular datasets can be large-scale and high-dimensional, posing computational and generalization challenges [101]. As dimensionality increases, the risk of overfitting also increases. Thus, efficient training and adequate resources are essential. Scaling tabular models while preserving generalization remains a critical challenge [102].

Model Selection and Hyperparameter Tuning. Tabular models are highly sensitive to hyperparameters [103], [104]. Choosing suitable architectures and tuning parameters like learning rate or tree count is often costly and time-consuming. Although AutoML techniques [105], [106] offer automation, identifying optimal settings for deep tabular models under constraints remains difficult yet vital.

Domain-Specific Constraints. Applications in domains like healthcare or finance impose regulatory and ethical constraints [107]. Healthcare models must comply with privacy laws like HIPAA [108] and be interpretable to clinicians. Financial systems face fairness and compliance requirements. Such constraints affect algorithm choices and demand interpretability and validation [109].

C. Evaluation of a Tabular Method

We present the evaluation of tabular methods, ranging from traditional to modern, to provide a comprehensive evaluation across different aspects. For a given model on a dataset $\mathcal{D}$, we employ standard metrics that quantify the discrepancy between the predicted label $\hat{y}_i$ and the true label y_i.

Evaluation on A Single Task. For classification tasks, Accuracy (or Error Rate) is commonly employed as the primary metric. AUC and F1 scores are further used to address imbalanced label distributions, while Expected Calibration Error (ECE) [110] calculates the weighted average error of the estimated probabilities. All criteria are the higher, the better, except the error rate and ECE. For regression tasks, common metrics include Mean Squared Error (MSE), Mean Absolute Error (MAE), and Root Mean Squared Error (RMSE), with MAE and RMSE sharing the scale of the original labels. Lower values denote superior performance. Additionally, the coefficient of determination (R^2) is employed, with higher values indicating a better fit.

Evaluation on A Set of Tasks. The diversity of tabular datasets makes it hard for one model to perform best universally, so evaluation should consider both per-dataset results and aggregated metrics for overall effectiveness. Early research predominantly relied on Average Rank (Friedman Rank) [12], [35] and Critical Difference Comparisons to evaluate model performance across datasets. Models are ranked per dataset using metrics like accuracy or RMSE. Statistical tests such as Wilcoxon-Holm, Friedman, and Nemenyi [111] assess the significance of rank differences. To mitigate the influence of outliers, PAMA [12] measures the fraction of datasets where a model achieves the best accuracy, while P95 quantifies the likelihood that at least 95% of the maximum.

As research progressed, more diverse evaluation metrics were introduced, *e.g.*, Arithmetic Mean, normalized Accuracy, normalized RMSE [32], [30], Mean Normalized Error, Shifted Geometric Mean (SGM) error [30]. Beyond absolute performance, relative comparisons such as Relative Improvement [112], ELO-based evaluation [41] are also important.

D. Tabular Benchmarks and Datasets

This section introduces existing benchmarks and datasets, along with associated considerations for constructing the benchmarks and evaluation protocols.

1) Popular Tabular Benchmarks and Datasets:

We begin by introducing several benchmarks constructed from raw tabular features across various dimensions, followed by datasets enriched with semantic annotations.

Standard Benchmarks. Tabular learning methods often exhibit dataset-specific performance, and evaluations based on a small number of datasets may be biased by randomness or dataset idiosyncrasies. Therefore, constructing comprehensive benchmarks is critical for robust and generalizable evaluation.

An effective benchmark should cover a wide range of datasets to evaluate generalization across different tasks and feature characteristics. This includes binary classification, multi-class classification, and regression tasks. For example, [12] benchmarked 179 classifiers across 121 datasets and found that Random Forest variants consistently outperformed others. [48] evaluated MLPs augmented with ensembling and data augmentation across 40 classification datasets. [29] further demonstrated the competitiveness of MLPs, ResNets, and Transformer-based models on 11 datasets. [32] conducted a broad comparison on 45 datasets, analyzing the performance gap between tree-based and deep learning methods.

Benchmarks should include datasets of varying sizes to evaluate scalability and efficiency. [35] uses 176 classification datasets with different sizes to compare across methods. However, limited tuning and strict time constraints may have led to suboptimal evaluations for some deep methods [113].

To ensure generalization, datasets should come from multiple domains, *e.g.*, healthcare, biology, and finance. [114] evaluates attention and contrastive learning methods on 28 datasets. [42] uses over 300 datasets covering diverse tasks, sizes, and domains to assess the generalization of DNN-based models. TabArena [115] constructs a continuously maintained living benchmarking system for standardized and reliable evaluation.
Semantic-Enriched Datasets. Recent work has focused on tabular datasets with rich semantics, such as task-related meta-information and attribute names. UniTabE [116] introduces a 7TB dataset with 13 billion examples for pre-training. CM2 [76] proposes OpenTabs for cross-table pre-training, including 46M tables with column name semantics. TP-BERTa [75] filters OpenTabs to 101 binary and 101 regression datasets with at least 10,000 samples and no more than 32 features, totaling 10 million samples. GTL [78], TabLib [117], and T4 [88] also extract large-scale data from real-world sources such as Kaggle and GitHub. These semantic-rich datasets are primarily used for pre-training LLMs on tabular data, while others serve for evaluating standard methods. Several toolboxes support both classical and deep methods [118], [119], [120], [121]. Building a comprehensive benchmark requires considering both the diversity and quality of the dataset.

2) Evaluation Protocols: Given the strong sensitivity of tabular methods to data and the additional randomness in deep methods, robust evaluation is essential. Furthermore, due to the high computational cost of some methods, it is equally important to ensure evaluation efficiency.
Model Selection. Model selection on the validation set involves hyperparameter tuning and early stopping to ensure reliable evaluation. Given the high dimensionality of hyperparameters in deep models, automated tools like Optuna [122] are widely used for efficient search [29], [67]. Models are typically trained with multiple random seeds for stability, and early stopping [123] is applied in each trial to avoid overfitting, selecting the best epoch based on validation performance.
Performance Evaluation. To assess generalization and prevent overfitting, models are typically evaluated using train/val/test splits. However, fixed splits may lead to inconsistent results. With the rise of deep learning, more robust evaluation protocols have been proposed [124], including (1) fixing the split and running multiple trials with different random seeds [52], [57], [67], [29], [56], [125], [69]; and (2) cross-validation, where new splits are generated per fold [61], [85], [126], [30]. Hybrid approaches combining both have also been explored [127].

Recent work has highlighted that holdout-based hyperparameter tuning can be unstable and prone to overfitting [128], [113]. [113] found it ineffective on TabZilla [35] datasets, advocating for 5-fold cross-validation, which altered prior meta-feature conclusions. [42] further refined these insights by identifying more predictive meta-features. For small datasets, alternative evaluation strategies have been proposed [129]. [130] showed that simple data reshuffling can improve generalization, making holdout selection competitive with cross-validation while being more efficient.

III. From Classical to Deep Method

We present possible advantages of deep learning for tabular data, as well as the potential challenges of deep learning when compared with tree-based methods.

A. Advantages of deep representation learning

Deep tabular models offer several advantages beyond performance when compared with classical methods.
Ability to Model Complex Feature Interactions. DNNs effectively capture high-order, non-linear feature interactions, which are difficult for traditional models like linear regression or decision trees [49], [52]. Through hierarchical representations, low-level interactions are learned in early layers, while deeper layers capture complex dependencies, making DNNs suited for modeling intricate tabular relationships.
End-to-End Learning. Unlike traditional methods that separate feature engineering, preprocessing, and tuning, DNNs can learn directly from raw features without manual transformations. This end-to-end training reduces human bias and streamlines workflows [29], [131]. DNNs also support multi-task learning, enabling shared representations that improve both performance and efficiency [132], [68], [47].
Integration with Other Modalities. Deep tabular models excel in multi-modal pipelines, combining tabular data with images, audio, or text. In AI4science, for example, tabular data may be fused with images [133] (*e.g.*, medical imaging) or time series [134], [135] (*e.g.*, forecasting). DNNs naturally model such heterogeneous interactions, enabling more accurate, comprehensive predictions.
Flexibility with Dynamic Environments. DNNs benefit from gradient-based optimization, enabling efficient, iterative training and adaptability to changing objectives [9]. Unlike tree-based models, which often require task-specific adjustments, DNNs handle dynamic environments such as real-time prediction, financial analysis, and decision systems where feature relationships may shift. Their adaptability supports online or incremental learning, integrating new data without retraining from scratch [136], [137].
Long-Term Knowledge Transfer and Learning. DNNs can retain and transfer knowledge across tasks [138], reducing the need for retraining when applied to related domains [139]. This is especially valuable in AI4science, where models trained on one data type can be adapted to others, saving time and resources. Such transferability enables more efficient and sustained use of data and model capabilities.

B. Debates between Tree-Based Methods and DNNs

While deep tabular methods show promise in learning representations and nonlinear predictors, they often struggle to outperform classical models like Gradient Boosted Decision Trees (GBDT). Many studies still consider GBDT strong baselines [32], [35], and their relative advantages may diminish across diverse evaluation datasets.

Several reasons contribute to why tree-based methods retain their advantages over DNNs in many tabular tasks:

Better Handling of High-Frequency Data. Tree-based methods, especially GBDT, efficiently handle high-frequency or dense data with small variations [34]. By recursively splitting on informative features, they capture local and global patterns effectively. In contrast, DNNs may struggle with fine-grained patterns without extensive regularization [140]. To address this, [43] showed that periodic activations enhance learning of high-frequency functions.

Natural Handling of Mixed Data Types. Tree-based models naturally support mixed data types and handle categorical features without one-hot encoding [9], [42], streamlining preprocessing, whereas DNNs rely on encoding methods, adding complexity and potentially harming performance [61].

Lower Computational Requirements for Training and Inference. Tree-based models are often more computationally efficient than DNNs [29], especially for smaller datasets or rapid deployment [35]. GBDTs train quickly and require fewer resources, while DNNs typically demand more computation (*e.g.*, GPUs, time) to match performance [141], [84], making them less suitable in resource-limited settings.

Robustness to Noisy and Missing Data. Tree-based models handle noisy and missing data more effectively. Decision trees accommodate missing values through optimal splitting and tolerate inconsistent data [32]. In contrast, DNNs are more sensitive and require preprocessing (*e.g.*, imputation, noise filtering) to maintain performance [63], [85].

Interpretability and Transparency. Tree-based models are highly interpretable [58], [59], [142]. Their decision paths can be visualized, and feature importance is directly accessible [143], [144], [145], making them well-suited for domains like finance and healthcare. Although interpretability tools like LIME [146] and SHAP [147] exist for DNNs, tree-based models remain more intuitive. Recent work [57], [126] has aimed to improve neural network interpretability by mimicking tree-based behavior.

Handling Outliers and Skewed Data. Tree-based methods are more robust to outliers and skewed distributions. Decision trees split based on feature ranges, naturally isolating extreme values. In contrast, DNNs often require additional techniques (*e.g.*, outlier removal) to manage such data [148], [149].

In conclusion, despite the rapid progress of deep learning, tree-based models such as XGBoost [143], LightGBM [145], and CatBoost [144] remain the dominant solution for many tabular tasks. They offer superior training efficiency and robustness to unscaled features. Furthermore, classical probabilistic methods continue to serve as strong baselines for tasks requiring uncertainty estimation [150], [151]. Consequently, rigorous benchmarking against these established non-DNN approaches remains a critical standard for evaluating the effectiveness of deep tabular representation learning.

IV. Taxonomy of Specialized Methods

Similar to the evolution of deep learning, which progresses from specialized learning to transfer learning and ultimately to foundation models [196], we categorize deep tabular methods into three groups, as shown in Fig. 2: *specialized methods*, *transferable methods*, and *general methods*. This classification reflects both the evolutionary development of deep learning techniques and the increasing generalization capabilities.

Beyond such taxonomy, it is also insightful to view the field through the lens of input modalities. Similar to benchmarks (in Sub-Section II-D), recent deep tabular methods can also be divided into *standard methods* and *semantic-enriched methods*. Standard methods focus on extracting patterns purely from raw numerical and categorical values, modeling the structural relationships between rows, columns, and targets. In contrast, semantics-enriched methods integrate auxiliary textual information with LLMs, such as attribute names and meta descriptions. While our survey is organized primarily by generalization capability, this modality-based distinction permeates all three categories, with semantic enrichment becoming increasingly central in transferable and general models.

Specialized methods, being the earliest developed and most widely used category, will be our starting point for discussion. Tabular data consists of features (columns), samples (rows), and objectives (labels), which together define the structure and the task objectives. We emphasize detailed strategies for obtaining high-quality representations at both feature- and sample-level for the target task. Specifically, given the input data, according to the general learning objective in Equation 1, we consider how to transform the tabular input $\boldsymbol{x}_i$ (feature aspect), how to construct relationships between samples (sample aspect), how to design the objective $\ell(\cdot)$ and regularize $\Omega(\cdot)$ (objective aspect). In particular,

- **Feature Aspect.** We focus on how to transform the raw tabular input into intermediate representations. We consider two types of features: numerical and categorical. By explicitly modeling the relationships between the two features (*e.g.*, feature importance and interactions), we are able to enhance the model's understanding of the input space.
- **Sample Aspect.** In addition to features, we explore how to retrieve and utilize neighboring samples to capture inter-sample dependencies, thereby improving predictions. In order to improve the model's prediction ability, we explore the relationships between a target sample and its "neighbors."
- **Objective Aspect.** We examine how to modify the loss function and objective to introduce inductive biases. By directly guiding the learning process with the target variables, we incorporate prior knowledge or task-specific preferences into the model, improving its generalizability and interpretability.

It is worth noting that these three aspects are not mutually exclusive but rather complementary. Feature-aspect methods primarily address the heterogeneity of tabular attributes, transforming diverse data types into a unified representation space capable of capturing high-order interactions. Sample-aspect methods compensate for the lack of explicit spatial or sequential structure by leveraging relationships between instances (*e.g.*, retrieval or attention), thereby enriching the representation

TABLE I
THE TAXONOMY OF REPRESENTATION LEARNING FOR TABULAR DATA.

Algorithm Category			Reference
Specialized Methods	§ V Feature-aspect Methods	Feature Encoding	[29], [43], [62]
		Feature Selection	[57], [58], [142], [59], [126], [152], [153]
		Feature Projection	[50], [29], [30], [56]
		Feature Interaction	[52], [60], [61], [53], [63], [47], [154]
	§ IV-A Sample-aspect Methods	Sample Interaction	[68], [155], [156], [125], [65]
		Neighbor Retrieval	[157], [66], [67], [31]
	§ IV-B Objective-aspect Methods	Training Objective	[65]
		Training Regularization	[158], [48], [64]
§ VI Transferable Methods		Homogeneous	[61], [46], [68], [159], [44], [160], [161], [162], [45], [163], [164], [165], [166]
		Heterogeneous	[167], [168], [161], [70], [71], [62], [169], [170]
		Language Model	[74], [171], [116], [76], [75], [172], [173], [174], [79], [175], [176], [77], [177]
		Vision Model	[178], [179], [180], [72], [73], [181], [182], [183]
§ VII General Mehtods		Raw-Feature-based	[82], [83], [84], [184]
		TabPFN Variants	[85], [87], [185], [40], [186], [41], [187], [188], [189], [39], [190], [191], [192]
		Semantics-based	[88], [89], [90], [193], [194], [195]

with global context. Finally, Objective-aspect methods inject necessary inductive biases (such as sparsity or regularization) directly into the optimization process to guide generalization. Deep tabular models (*e.g.*, FT-Transformer [29], SAINT [68]) can integrate strategies from multiple aspects to tackle the complex challenges of tabular learning effectively.

In specialized methods, we focus solely on learning from pure data, excluding feature semantics considered in transferable methods (in Section VI). Since specialized methods cover lots of approaches—with feature-aspect methods being the largest subset—we first introduce sample-aspect and objective-aspect methods, then Feature-aspect methods in Section V.

A. Sample-aspect Specialized Methods

Sample interaction methods take a retrieval-based approach, focusing on relationships between individual samples rather than features. In a tabular dataset, each sample $\boldsymbol{x}_i$ represents a row with d features, and the goal is to leverage relationships between a target sample and its "extracted neighbors" to improve predictions. The general form for the sample interaction methods can be expressed as:

$$\hat{y}_i = f\left(\mathcal{R}(\boldsymbol{x}_i, \mathcal{D}; \Phi)\right), \tag{2}$$

where $\mathcal{D}$ is the set of all samples (training data) available for retrieval or learning. $\mathcal{R}(\cdot)$ is the sample interaction module, which retrieves or aggregates information from relevant samples in $\mathcal{S}$ for the target sample $\boldsymbol{x}_i$. Φ represents the learnable parameters of $\mathcal{R}$. $f(\cdot)$ is the prediction head that maps the aggregated information to the final output $\hat{y}_i$.

Sample aspect approaches can be broadly categorized into two main strategies. The first approach introduces the modeling of sample relationships $\mathcal{R}$ during representation training, allowing the model to learn better representations by capturing inter-sample dependencies. The second approach is retrieval-based models, which directly predict outcomes by retrieving and utilizing neighbors' relationships $\mathcal{R}$ when testing.

Sample Interaction. These methods assist in representation learning by allowing the model to capture relationships between samples, which in turn helps generate a more robust representation during training. During testing, the model becomes more sensitive to each sample without interaction.

SAINT [68] introduces inter-sample attention beyond inter-attribute attention, which improves row classification by relating each row to others in the table. NPT [155] extends this via non-parametric Transformers, whereas Hopular [156] employs Hopfield networks, sharing conceptual alignment with SAINT [68]. Unlike nearest-neighbor classification, the distance metric is learned end-to-end. Trompt [125] posits that the feature importance in tabular data is sample-dependent. During feature extraction, it treats the information between samples as prompts. PTaRL [65] identifies two issues in the representation of tabular data samples: entanglement and localization. It addresses these by modeling global sample relationships through prototype generation and representation projection, helping the model produce clear and consistent decisions.

Neighbor Retrieval. These methods construct high-quality contexts to aid prediction by retrieving valuable neighbors and designing efficient ways to utilize them based on the relationships between samples. The training data is used to assist during testing.

DNNR [66] argues that a key advantage of neighbor-based methods is the model's transparency, meaning that the model's decisions can be explained by inspecting its components. TabR [67] proposes that, compared to purely parametric (*e.g.*, retrieval-free) models, retrieval-based models can achieve superior performance while also exhibiting several practically important properties, such as the ability for incremental learning and enhanced robustness. ModernNCA [31] revitalizes the classic tabular prediction method, Neighbourhood Component Analysis (NCA) [197], by designing and incorporating deep learning architectures and strategies. The resulting method efficiently leverages neighboring samples for prediction.

B. Objective-aspect Specialized Methods

The general objective learning f follows the structural risk minimization as in Equation 1, where ℓ is the loss function

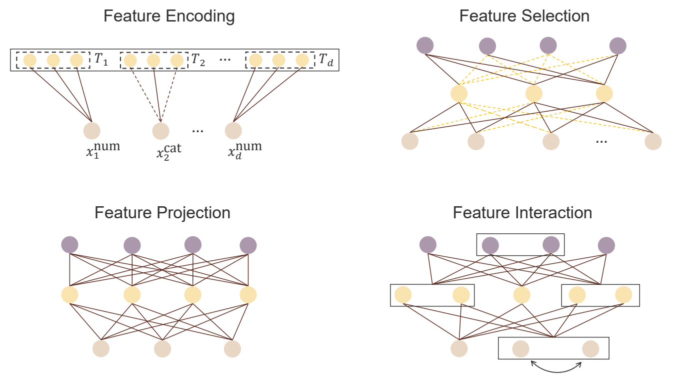

Fig. 3. Illustration of feature-aspect methods, including feature encoding, feature selection, feature projection and feature interaction.

to set the training objective between the prediction and the ground truth label. $\Omega(\cdot)$ is the regularization on the model, which directs the objective or restricts the complexity of f.

Objective-aspect methods in deep learning are an extension of these traditional regularization techniques, where inductive bias is introduced by adjusting the loss function ℓ or adding regularizers Ω. In the training progress, the goal is to leverage regularization on the model to improve predictions.

Objective-aspect approaches can be broadly categorized into two main strategies. The first approach involves training objectives, which enhance the model with a specialized ability. The second approach introduces a regularizer, allowing the model to learn strong generalized representations.

Training Objective. For training objectives, PTaRL [65] constructs prototype-based projection space and learns the disentangled representation around global prototypes. PTaRL uses a diversification constraint for representation calibration and introduces a matrix orthogonalization constraint to ensure the independence of global prototypes.

Training Regularization. For training regularization, RLNs [158] overcome the challenge of an intractable number of hyperparameters during training by introducing an efficient tuning scheme, which minimizes a new "Counterfactual Loss." In RLNs, the regularization coefficients are optimized together with learning the network weight parameters. [48] introduces "cocktails," dataset-specific combinations of 13 regularization techniques, showing that even simple neural networks can outperform tree-based architectures when optimized with these methods. TANGOS [64] introduces a regularization-based improvement. It regularizes neuron attributions to encourage neurons to specialize and become orthogonal to one another.

V. Feature-aspect Specialized Methods

Tabular data consists of various features, including categorical and numerical variables. Its complexity stems from varied feature types, interrelationships, and often high dimensionality. Traditional methods rely on manual feature engineering—such as encoding categorical variables and feature selection—to improve performance and reduce overfitting. As deep learning evolved, these techniques have been integrated and extended. Deep tabular models can automatically learn feature representations, reducing the need for manual engineering. Feature-aspect methods, like encoding, selection, projection, and interaction, transform raw inputs into informative forms, helping capture intricate relationships and improve generalization. Feature encoding and interaction methods are specifically designed to address the heterogeneity of features, transforming diverse data types into a unified latent space. Meanwhile, feature projection techniques help mitigate the high dimensionality often resulting from one-hot encoding.

A. *Feature Encoding*

Various encoding strategies have been explored for both categorical and numerical features in tabular data. Additionally, with the advancement of the attention mechanism, feature tokenization, similar to word embeddings in natural language processing, transforms all features into embeddings.

Categorical Encoding. Categorical variables represent data types divided into groups, such as race, sex, age group, and educational level [198]. These features are usually converted into integers. Two common techniques are Ordinal Encoding and One-Hot Encoding.

Ordinal Encoding assigns each category a distinct integer, useful when categories have an inherent order like "low," "medium," and "high." Its main advantage is simplicity and efficiency, transforming the variable into a single numeric column. However, it assumes an ordinal relationship that may not exist—for example, "red," "blue," and "green," with Ordinal Encoding would introduce an artificial order that does not reflect any meaningful ranking.

On the other hand, One-Hot Encoding creates a binary column for each unique category. For example, the variable "color" with categories red, blue, and green would generate three columns: "is_red," "is_blue," and "is_green," encoding red as (1,0,0), blue as (0,1,0), and green as (0,0,1). This method suits nominal categorical variables without inherent order. While it avoids ordinal assumptions, One-Hot Encoding can produce a high-dimensional feature space when many unique values exist, increasing computational costs and risking overfitting.

In some cases, more advanced encodings address these limitations. For example, Target Encoding assigns each category the mean target value, useful when categorical features strongly relate to the target. In Leave-one-out embedding, every category is replaced with the mean of the target variable of that category, which excludes the current row to avoid overfitting.

Numerical Encoding. For encoding, MLP-PLR [43] introduces two numerical encoding methods: Piecewise Linear Encoding (PLE) and Periodic Activation Functions. These encoding methods can be integrated with other differentiable layers (*e.g.*, Linear, ReLU) to enhance performance. PLE produces alternative initial representations for the original scalar values and is based on feature binning. Periodic Activation Functions take into account the fact the embedding framework where all features are computed independently of each other forbids mixing features during the embedding process and train the pre-activation coefficients instead of keeping them fixed. [34] utilizes tools from spectral analysis, showing that functions described by tabular datasets often have high irregularity, and can be smoothed by transformations such as scaling and ranking to improve performance. They propose "frequency reduction" as an inductive bias during training.

Feature Tokenization. Feature tokenizer performs a similar role to the feature extractor in traditional models. It transforms the input features to embeddings [60], [29]. Since the feature representations of features are very sparse and high-dimensional, a common way is to represent them into low-dimensional spaces (*e.g.*, word embeddings). The general form for feature tokenization can be expressed as:

$$\boldsymbol{T}_{i,j} = \boldsymbol{b}_j + \mathcal{T}(x_{i,j}; \Psi) \in \mathbb{R}^t, \tag{3}$$

where $\mathcal{T}(\cdot)$ is the feature tokenizer module, which transforms the input feature vector $\boldsymbol{x}_i \in \mathbb{R}^d$ to a token embedding $\boldsymbol{T}_{i,j} \in \mathbb{R}^t$. t is the dimension of token embedding. $\boldsymbol{b}_j$ is the j-th feature bias. $\mathcal{T}$ can be implemented with different forms. Ψ represents the learnable parameters of $\mathcal{T}$.

In AutoInt [60], both the categorical and numerical features are embedded into low-dimensional spaces, which reduces the dimension of the input features and meanwhile allows different types of features to interact with each other. TabTransformer [61] embed each categorical feature into a parametric embedding of dimension t using Column embedding. An embedding vector is assigned to each feature, and a set of embeddings is constructed for all categorical features. Unlike TabTransformer, SAINT [68] proposes projecting numerical features into a t-dimensional space before passing their embedding through the transformer encoder. FT-Transformer [29] adapts the Transformer architecture for tabular data, where all features are transformed to embeddings and applies a stack of Transformer layers to the embeddings. Specifically, the numerical tokenizer is implemented as the element-wise multiplication $\boldsymbol{T}_i^{\text{num}} = \boldsymbol{b}_i^{\text{num}} + x_i^{\text{num}} \cdot \boldsymbol{W}_i^{\text{num}}$, and the categorical tokenizer is implemented as the lookup table $\boldsymbol{T}_i^{\text{cat}} = \boldsymbol{b}_i^{\text{cat}} + \boldsymbol{e}_i^T \boldsymbol{W}_i^{\text{cat}}$, where $\boldsymbol{e}_i^T$ is a one-hot vector for the corresponding categorical feature. Other transformer-based methods, like [63], [70], [169], use the same feature tokenizer as FT-Transformer.

B. Feature Selection

High dimensionality in tabular data often leads to overfitting, where models focus on irrelevant features. Feature selection addresses this by retaining only the most informative features, improving generalization and reducing computational cost. Traditional tree-based models perform feature selection inherently by evaluating feature impact during construction. Decision trees utilize metrics such as information gain or the Gini index for feature selection, while ensemble methods like random forests determine feature importance by assessing each feature's contribution [199], [200]. Recently, modern deep learning methods for tabular data often mimic trees' structures for feature selection.

GrowNet [57] and NODE [58] mimic ensemble methods, with GrowNet stacking weak DNN learners inspired by GBDT, and NODE using differentiable oblivious trees with Bagging and Stacking. NODE-GAM [59] adapts NODE into a scalable GAM [201] for learning non-linear patterns. TabNet [142] combines DNN representation learning with tree-like interpretability and sparse feature selection, while GRANDE [126] leverages tree-style hard splits via gradient-based learning to bridge the gap with deep models. Recursive Feature Machines (RFM) [152] enables kernel machines to learn features by recursively reweighting features via a gradient-inspired mechanism without backpropagation. xRFM [153] extends feature learning kernel machines with a tree structure to both adapt to the local structure of the data and scale to unlimited amounts of training data.

In parallel, instead of mimicking tree structures, another line of work integrates differentiable feature selection into neural networks. STG [202] enhances LASSO by modeling nonlinear feature interactions and using smooth Bernoulli-based gates for regularization, while LSPIN [203] learns instance-wise gating probabilities to select the most informative features per sample.

C. Feature Projection

Feature projection methods aim to project the raw data into a middle form, enhancing the representation ability for later architectures. Feature projection methods can be broadly categorized into two main approaches: MLP variants and special designed architectures. These approaches aim to enhance the model's ability to represent complex features for underlying feature structures.

MLP Variants. For model architecture, RTDL [29] investigates both ResNet-like and Transformer-based architectures tailored for tabular data, proposing simple yet effective adaptations of these widely-used deep models. Another contemporaneous work [48] enhances the MLP architecture by equipping it with a comprehensive suite of modern regularization techniques. Instead of introducing architectural innovations, this study focuses on systematically exploring different regularization methods to identify an effective "regularization cocktail" for plain MLPs. For a more comprehensive strategy, RealMLP [30] explores multiple aspects including preprocessing, hyperparameters, architecture, regularization, and initialization.

Special Designed Architectures. For units, motivated by the observation that normalization techniques are prone to disturbances during training, SNN [50] proposes the Scaled Exponential Linear Unit (SELU) to improve deep models for tabular data. NAMs [204] uses exp-centered (ExU) hidden units to improve the learnability for fitting jumpy functions. BiSHop [56] uses a dual-component approach, sequentially processing data both column-wise and row-wise through two interconnected directional learning modules. They use layers of generalized sparse modern Hopfield layers, a sparse extension of the modern Hopfield model with learnable sparsity.

D. Feature Interaction

Feature interaction methods aim to model relationships among features to enhance the representation power of deep learning models on tabular data. In tabular datasets, each sample $\boldsymbol{x}_i \in \mathbb{R}^d$ is described by d features. The general form for feature interaction methods can be expressed as:

$$\hat{y}_i = f\left(\mathcal{H}(\boldsymbol{x}_i; \Theta)\right), \tag{4}$$

where $\boldsymbol{x}_i \in \mathbb{R}^d$ is the input feature vector for a single instance, $\mathcal{H}(\cdot)$ is the feature interaction module, which transforms the input $\boldsymbol{x}$ by capturing feature dependencies or generating higher-order feature interactions. Θ represents the learnable parameters of $\mathcal{H}$. $f(\cdot)$ is the prediction head that maps the transformed representation to the final output $\hat{y}$.

Feature interaction methods can be broadly categorized into two main approaches: the design of automatic feature interaction modules and the mining of implicit feature relationships. These approaches aim to enhance the model's ability to learn complex feature interactions and underlying feature structures within tabular data.

Automatic Feature Interaction Modules. These methods do not assume specific feature types within the tabular dataset. Instead, they focus on improving the feature interaction process, enabling the model to learn complex, high-order feature relationships autonomously.

DCNv2 [52] improves the learning of the model's feature interaction by improving the "Cross Network" structure. AutoInt [60] maps the original sparse high-dimensional feature vectors into a low-dimensional space and models high-order feature interactions by stacking interaction layers with a multi-head attention mechanism. Unlike AutoInt, the TabTransformer[61] only maps categorical features into contextual embeddings and feeds them into a Transformer model, while numerical continuous features are directly concatenated with the interacted contextual embeddings. When tabular data contains only numerical features, TabTransformer behaves in an MLP-like manner. Conversely, when the data contains only categorical features, TabTransformer operates similarly to AutoInt.

Implicit Feature Relationships. Methods in this category typically assume that features in tabular data can be abstracted into implicit types and that it is necessary to design a suitable feature learning process to adapt to the characteristics of different types of features.

DANets [53] propose the existence of underlying feature groups in tabular data, where features within each group are correlated. They learn to group input features and perform further feature abstraction. SwitchTab [47] introduces the idea of extracting sample-specific "Salient Features" and sample-shared "Mutual Information" in tabular features. It leverages self-supervised learning to assist in learning feature representations. ExcelFormer [63] argues that while DNN assigns weights to each feature, it does not actively exclude irrelevant features. To address this, it introduces Semi-Permeable Attention for feature interaction, which allows features with lower information content to access information from more informative features while preventing highly informative features from being influenced by less relevant ones. AMFormer [154] proposes the hypothesis that arithmetic feature interactions are crucial for deep tabular models. Based on the Transformer architecture, it introduces components designed to extract both additive and multiplicative interaction information.

VI. From Specialized to Transferable Model

Instead of training a tabular model from scratch, learning based on a Pre-Trained Model (PTM) may increase the learning efficacy and reduce the resource and data requirement. For example, in a house prices prediction task, training a regressor in a certain area may benefit from a well-trained predictor from its neighborhood. These methods primarily tackle the challenge of low-quality and missing data (specifically, label scarcity) by transferring knowledge from data-rich source domains.

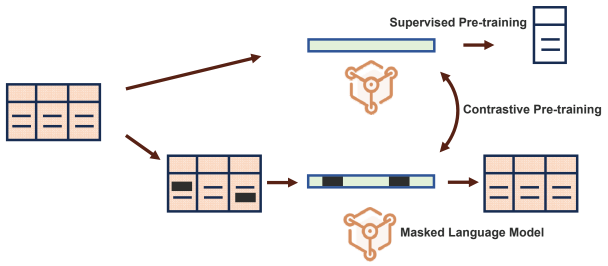

Fig. 4. Illustration of homogeneous transferable tabular methods. The pre-trained model could be constructed from supervised or self-supervised learning, including masked language model, contrastive pre-training, and hybrid methods.

Learning by reusing the PTM usually contains two stages. The first is the pre-training of a tabular model, from one or more upstream tasks. Given the PTM and a downstream task, an adaptation strategy is needed to transform the PTM to the target task or facilitate the learning of the target model. Formally, a well-trained model $g_{\boldsymbol{\Theta}}$ is often available and can be leveraged to facilitate the training of $f_{\boldsymbol{\theta}}$ over $\mathcal{D}$. Here, $g_{\boldsymbol{\Theta}}$ is pre-trained on a dataset $\mathcal{D}' = \{(\boldsymbol{x}'_j, y'_j)\}_{j=1}^{N'}$ with instances $\boldsymbol{x}'_j \in \mathbb{R}^{d'}$ and labels $y'_j \in [C']$. To reuse expert knowledge in $g_{\boldsymbol{\Theta}}$, an adaptation strategy is applied: $f_{\boldsymbol{\theta}} = \textbf{Adapt}(f_{\boldsymbol{\theta}_0} \mid \mathcal{D}, g_{\boldsymbol{\Theta}})$, where $\boldsymbol{\theta}_0$ is the initialization of the model. The notation could also be extended to cases with more than one PTM. The main challenge to reuse one or more PTMs is to bridge the gap between the PTM and the target tabular model [205]. We categorize PTMs into three kinds based on the source of PTM $g_{\boldsymbol{\Theta}}$.

Homogeneous Transferable Tabular Model. First, the PTM may come from the same form of task (with $d' = d$ and $C' = C$, but with different distributions $\Pr(\mathcal{D}') \neq \Pr(\mathcal{D})$ or model families $g \neq f$). For example, those pre-trained from other domains [69], or those unlabeled instances [46], [68].

Heterogeneous Transferable Tabular Model. In addition, we consider a PTM pre-trained from a slightly different task with $\mathcal{D}$. In addition to the previous difference, the PTM $g_{\boldsymbol{\Theta}}$ may differ from $f_{\boldsymbol{\theta}}$ in feature dimension ($d' \neq d$) or target class set ($C' \neq C$), so the adaptation method $\textbf{Adapt}(\cdot)$ must handle such heterogeneity [62], [169].

Cross-Modal Transferable Tabular Model. Moreover, the pre-trained model could also be constructed from another modality, such as vision and language domains. The cross-modality PTM is hard to be applied to the tabular prediction task in most cases, so auxiliary information from the tabular task like the semantic meaning of attributes (*i.e.*, the attribute names) are usually assumed to be available in this case, where PTM like large language models may provide the latent semantic meanings as external knowledge [74], [71].

A. Homogeneous Transferable Tabular Model

Benefiting from the strong capacity of deep neural networks, some recent studies focus on pre-training a tabular model from unsupervised instances, and then adapting the model via fine-tuning the PTM on the target (even few-shot) labeled examples.

This strategy could be applied in standard supervised learning or semi-supervised learning.

Supervised Pre-training Objectives. A straightforward way to incorporate the target variable in pre-training is to treat input corruption as augmentation for supervised objectives. [69] identifies pre-training practices for tabular deep models across datasets and architectures. They demonstrate that incorporating target labels in pre-training improves downstream performance and propose several target-aware objectives.

Self-Supervised Pre-training Objectives. The self-supervised pre-training objectives can be mainly categorized into three categories: the masked language model, contrastive pre-training, and hybrid methods.

Masked Language Model (MLM). MLM is an unsupervised pre-training objective where a random subset of features is masked and predicted in a multi-target classification manner [61]. VIME [46] estimates mask vectors from corrupted data and reconstructs features, generating multiple augmented samples via different masks and imputations. SubTab [44] reconstructs data from a subset of features instead of corrupted inputs to better capture latent representations. SEFS [160] reconstructs the input using a randomly selected feature subset and estimates a gate vector to indicate feature selection. MET [162] concatenates feature representations and adds an adversarial reconstruction loss to the standard objective.

Contrastive Pre-training. Contrastive pre-training uses data augmentations to generate positive pairs or two different augmented views of a given example, and the loss function encourages a feature extractor to map positive pairs to similar features. The key factor in contrastive learning is to generate positive and negative versions of a given instance $\boldsymbol{x}_i$. SAINT [68] utilizes cutMix in the input space and mixup in the embedding space to obtain positive pairs, where other instances $\boldsymbol{x}_{j\neq i}$ are treated as negative ones. SCARF [45] generates a view for a given input by selecting a random subset of its features and replacing them with random draws from their respective empirical marginal distributions. STab [163] minimizes the distance between the representations of the same instance processed by these two weight-sharing neural networks, with the stop-gradient operation applied to the target network, ensuring to model invariance with respect to more complicated regularizations [206], [45]. DoRA [165] incorporates domain knowledge, training by intra-sample pretext task and inter-sample contrastive learning to learn contextualized representations. DACL+ [159] uses Mixup noise to create similar and dissimilar examples by mixing data differently to overcome the reliance on a particular domain.

Hybrid Methods. [161] explores supervised and unsupervised pre-training strategies, using MLM and multi-label classification, and finds that supervised pre-training yields more transferable features. LFR [166] pre-trains models by reconstructing multiple randomly generated projections, demonstrating applicability across tabular, vision, and language data. ReConTab [164] combines self- and semi-supervised learning, using feature selection and contrastive learning to distill task-relevant information. [69] investigates whether supervised pre-training helps with fully labeled tabular data and shows that target-aware pre-training benefits downstream performance. [205] provides a systematic review and summarizes the recent progress and challenges of self-supervised learning for non-sequential tabular data.

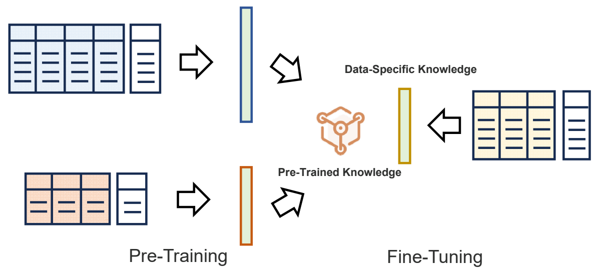

Fig. 5. Illustration of heterogeneous transferable tabular methods. During pre-training on one or multiple datasets, most of the parameters in the PTM are trained. For downstream tasks, only a small subset of parameters is fine-tuned.

B. Heterogeneous Transferable Tabular Model

The main intuition lies in the mapping f and g work in a similar fashion, *i.e.*, predicting the labels with similar mechanisms. Therefore, the main idea to transfer knowledge is to match the target model with the well-trained one, over the weight space or the prediction space.

Early methods focus on feature-level heterogeneity between f and g, assuming a shared feature set between the pre-trained task $\mathcal{D}'$ and the target task $\mathcal{D}$, allowing weight transfer for shared features [207]. Neural models are advantageous due to their ability to learn reusable features and adapt to new domains. Deep PTMs can extract generalizable features, enabling knowledge transfer from vision and language strategies. For example, most PTM parameters are frozen, and only a small subset is fine-tuned using techniques like linear probing or parameter-efficient tuning.

Reuse PTM Pre-trained from One Dataset. These methods primarily focus on the difference between the pre-trained and down-streaming datasets. TabRet [70] utilizes masked autoencoding to make the transformer work in downstream tasks. To transfer pre-trained large language models to tabular tasks, ORCA [71] trains an embedder to align the source and target distributions. TabToken [62] focuses on improving the quality of the feature tokens, which are an important component in tabular deep models. TabToken leverages a conditional contrastive loss to improve the quality of learned embeddings and demonstrates enhanced transferability of deep learning models for tabular data. Pseudo-Feature [161] trains separate models per new feature. It pre-trains on upstream data without the feature, fine-tunes on downstream data to predict it, then uses the model to assign pseudo-values in the upstream data. The enriched data is used for another pre-training round before transfer. However, this method is computationally costly for broad feature space adaptation.

Reusing PTMs Pre-trained on Multiple Datasets. XTab [169] improves transformer transferability by using independent features and federated learning to handle varying column types and quantities across tables. Other methods learn shared, attribute-agnostic components across datasets to provide strong initialization for downstream tasks. [167] addresses the challenge of differing attribute spaces by treating the problem

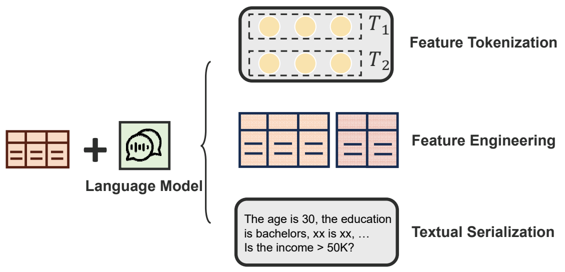

Fig. 6. Illustration of transferable tabular methods with a language model. The language model can be applied at various stages, including feature tokenization, feature engineering, and textual serialization.

as a meta-learning task. It utilizes a few labeled instances to infer latent embeddings, then applies them to unlabeled test instances for predictions, allowing the model to adapt to new tables with varying dimensions. DEN [168] adopts a three-block architecture—covariate transformation, distribution embedding, and classification—and shows that the latter two blocks can be fixed after pre-training. Meta-Transformer [170] maps raw inputs from various modalities into a shared space using a frozen encoder, allowing high-level semantic extraction without paired multimodal training data [208].

C. Reusing a Pre-trained Language Model

In some cases, the semantic meaning of features is available, making it natural to leverage pre-trained language models for tabular data. Typically, two types of semantic information can be derived from a tabular dataset $\mathcal{D}$. First, attribute names for each of the d features, $\mathcal{A} = A_1, \ldots, A_d$, provide useful context. Additionally, meta-information such as a textual description, denoted as meta_descript, can further enhance understanding. The learning process is then formulated as:

$$\hat{y}_i = f(\boldsymbol{x}_i, \mathcal{A} \mid \mathcal{D}, \text{meta\_descript}) \tag{5}$$

where the semantic information bridges the gap between feature spaces and facilitates knowledge transfer from pre-trained tasks to downstream applications.

Language Models for Feature Tokenization. When the feature space changes, language-based methods assume that semantic relationships exist between feature descriptions and rely on large-scale language models to capture these connections. For example, the feature ”occupation” in one task may share semantic similarity with the feature ”organization” in another, allowing feature-label relationships to be reused across different datasets. By extracting feature embeddings (tokens), tables of varying sizes can be transformed into a standardized set of tokens in a shared space. A pre-trained transformer then encodes transferable knowledge, aiding the fine-tuning process for downstream tasks.

TransTab [74] trains a tokenizer on column descriptions and cell values, using them as input to a gated Transformer. It is pre-trained via self-supervised or contrastive loss and evaluated on transfer and feature-incremental tasks. PTab [171] follows a similar approach, learning contextual representations from tokenized tabular datasets before fine-tuning. UniTabE [116] encodes column names, data types, and cell values into tokens, using an encoder-decoder architecture with Transformer and LSTM. It applies Multi-Cell-Masking and contrastive learning, treating sub-vectors as positives and other subsets as negatives.

CM2 [76] proposes a cross-table pre-training framework combining attribute names and feature values. It uses transformers to process feature tokens and applies a prompt-based Masked Table Modeling (pMTM) objective, where column names prompt masked feature prediction. TP-BERTa [75] adopts a similar approach with numerical discretization and magnitude tokenization, fine-tuning smaller PLMs like RoBERTa [209]. CARTE [172] models tabular data as a graph, embedding textual column names and entries. CARTE is pre-trained on YAGO3 [210] with contrastive loss on graphlets, where original and truncated variants as positives, others as negatives. Then pre-trained CARTE model is fine-tuned for downstream tasks.

Language Models for Feature Engineering. Discriminative features enhance the effectiveness of subsequent tabular learning models. Binder [173] uses LLMs to generate auxiliary features for knowledge grounding by identifying task inputs not directly answerable by the model. Since discriminative features are often manually designed, CAAFE [211] employs LLMs to generate auxiliary features from task and feature semantics, evaluating their quality with TabPFN [85]. FeatLLM [212] uses example-based prompting for LLMs to create new features from textual descriptions. TaPTaP [175], through large-scale pre-training on real-world tabular data, aims to capture generic tabular distributions and generate high-quality synthetic tables for various applications.

Language Models for Textual Serialization. A direct way to use pre-trained language models is converting tabular data into text, letting LLMs infer feature-label relationships from embedded expert knowledge, as shown in semantic parsing tasks [213]. LIFT [176] and TabLLM [77] serialize tables by combining feature names and task descriptions, treating prediction as text generation. LIFT fine-tunes on the full training set, while TabLLM uses few-shot learning. UniPredict [177] builds prompts from metadata, sample serialization, and task instructions, fine-tuning with confidence-weighted labels from an external model, validated on multiple datasets. CoT^2 [214] uses nearest neighbors and external models to guide LLMs in multi-step reasoning, advancing toward expert-level prediction.

Despite their advantages, textual serialization methods struggle as feature numbers grow, since prompts can exceed the model's context window. LLM effectiveness on tabular tasks is limited by available semantic information and external tabular model capabilities. Further discussion of LLM-based methods appears in the general tabular models in Section VII.

D. Reusing a Pre-trained Vision Model

Given the success of deep neural networks (DNNs) in visual tasks, it is natural to leverage pre-trained vision models for tabular data. Data augmentation techniques from image processing can also be applied after converting tabular data into visual formats. Similar ideas have been explored in time series forecasting [215] and irregular time series classification [216]. The main challenge is representing tabular instances as images. Unlike natural images, where neighboring pixels share semantic relationships, tabular features are permutation-invariant and lack spatial structure. Various methods have been proposed to transform tabular data into images, enabling the use of pre-trained vision models fine-tuned for tabular tasks. This subsection reviews these transformation strategies.

Various transformation strategies have been proposed to enable such reuse. One line of work uses dimensionality reduction techniques such as t-SNE [178] or Bayesian Metric Multidimensional Scaling [179] to project high-dimensional tabular features into 2D spaces, generating image-like representations. Another direction restructures tabular data into grid-like formats to introduce spatial relationships, as in TAC [180], IGTD [72], TablEye [73], and LM-IGTD [181]. Other approaches encode feature values as visual markers, such as fixed-position text [182] or colored bars [183], allowing CNNs to interpret tabular instances as images.

By transforming tabular data into images, these methods enable the application of powerful pre-trained vision models for tabular prediction tasks, leveraging established deep learning techniques to enhance tabular model performance.

VII. From Transferable to General Model

The general model (also referred to as the tabular foundation model) represents an advancement over the transferable model. It extends the generalization capabilities of a PTM to a variety of heterogeneous downstream tabular tasks, regardless of their diverse feature and class spaces, *without requiring additional fine-tuning*. In other words, given a pre-trained model g_{Θ}, it can be directly applied to a downstream tabular task $\mathcal{D}$ to predict the label of a test instance $\boldsymbol{x}^*$ as follows:

$$\hat{y}^* = g_{\Theta}(\boldsymbol{x}^* \mid \mathcal{D}) \ . \tag{6}$$

Thus, the general model shares similarities with the transferable tabular model, but with a greater emphasis on the "zero-shot" ability, aims to construct highly adaptive architectures capable of handling a wide array of heterogeneous datasets simultaneously. Importantly, it does not require an **Adapt** function, which further reduces the computational cost. General models aim to overcome the lack of spatial relationships, sensitivity to perturbations, and model selection constraints by enforcing a standardized input format or adapting architectures to handle arbitrary tabular structures.

Pre-training has transformed fields like vision and language [80], but its use in tabular data remains limited due to the inherent heterogeneity of tabular datasets. The tabular data vary greatly in the dimensionality and semantic meaning of each feature, even within the same domain. There are two main strategies to address the inherent heterogeneity in tabular datasets: improving the model's adaptability or homogenizing the diverse tabular formats. We categorize general tabular models into three parts based on their strategies for generalizability. The first focuses on raw-feature-based approaches, among which TabPFN variants represent a rapidly evolving branch and are thus discussed separately. The third category encompasses semantic-based methods that leverage attribute and task semantics to unify heterogeneous tasks.

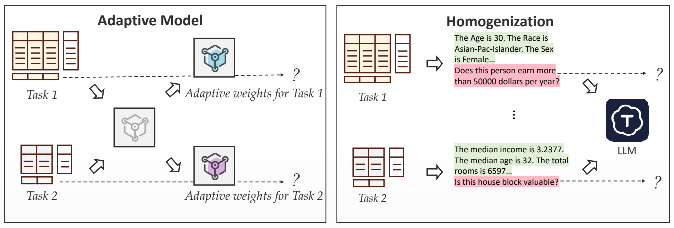
Adaptive Model
Task 1
Adaptive weights for Task 1
?
Task 2
Adaptive weights for Task 2
?
Homogenization
Task 1
The Age is 30. The Race is Asian-Pac-Islander. The Sex is Female...
Does this person earn more than 50000 dollars per year?
?
LLM
Task 2
The median income is 3.2377. The median age is 32. The total rooms is 6597...
Is this house block valuable?
?

Fig. 7. Illustration of general methods. These methods handle inherent heterogeneity by improving the model's adaptability or homogenizing the diverse tabular formats. Once pre-trained, they can be directly applied to downstream tasks without fine-tuning.

A. Raw-Feature-based General Models

To adapt general tabular models to heterogeneous datasets, two main strategies are adopted: data-centric and model-centric. From the data-centric perspective, models standardize tabular datasets into a homogeneous format. For example, TabPTM [82] uses meta-representations to transform all datasets into a uniform format, enabling pre-training. The resulting model can be directly applied or fine-tuned on downstream tasks without extra parameters. From the model-centric perspective, models are tailored to specific tasks for better adaptability. HyperFast [83] employs a Hyper Network [217] in meta-learning [218], learning a mapping from datasets to classifier weights. To handle varying input dimensions, it uses random projections. MotherNet [84] accelerates weight generation by enhancing HyperFast's architecture with Transformer-like modules. iLTM [184] further unifies tree-derived embeddings, dimensionality-agnostic representations, a metatrained hypernetwork, MLPs, and retrieval within a single architecture.

B. TabPFN Variants

The TabPFN family of models [85], [87], [185] leverages the in-context learning capabilities of transformers, directly predicting labels by adapting test instances according to the context of training examples. In the first version of TabPFN, an instance $\boldsymbol{x}_i$ is padded to a fixed dimension (*e.g.*, 100), and the features are projected to a higher dimension (*e.g.*, d') for further processing. The label y_i is processed similarly and added to the instance embeddings. These embeddings are processed through several layers of a Transformer, and the output token corresponding to the test instance is further predicted using a 10-way classifier. TabPFN is pre-trained over synthetically generated datasets with structured causal models (SCM) [219] and Bayesian Neural Networks (BNNs) [220]. Due to the high complexity of transformers, TabPFN is limited to small-scale tasks, with $N < 1000$, $d < 100$, and $C < 10$.

TabPFN v2 introduces a specialized feature tokenizer to better handle heterogeneity. Specifically, each cell in the table is projected to a k-dimensional vector using a shared mapping, and random position encoding vectors are added to differentiate features [190]. A two-way attention mechanism is used, with each feature attending to the other features in its row and then attending to the same feature across its column [221]. Several improvements have been made in TabPFN v2, including increased context size ($N < 10000$, $d < 500$), automatic feature engineering, and post-hoc ensemble methods. TabPFN v2.5 [185] further extends with up to 50000 data points and 2000 features, and introduces a new distillation engine that converts into a compact MLP or tree ensemble. Various applications have also been explored, including tabular data generation [222], anomaly detection [223], data augmentation [224], and time series forecasting [225].

The improvements of TabPFN stem from several aspects.[1]

Pre-training Improvements. TabForestPFN[189] extends TabPFN by pre-training ICL-transformers on synthetic forest datasets with complex decision boundaries. TabDPT[41] leverages real-world datasets and self-supervised objectives for pre-training, supporting both classification and regression. APT[226] enhances generalization by using adversarial synthetic data generated through adaptive agents that modify the data distribution. TabICL[40] incorporates tree-based SCMs via XGBoost and adopts curriculum learning with progressively larger synthetic datasets. Building upon masked joint-distribution modeling with an episodic, context-conditional objective, LimiX [94] introduces two scalable instantiations (LimiX-16M and LimiX-2M) of large structured-data models (LDMs) that complement language and physical world foundation models toward achieving general intelligence.

Scalable Improvements. The efficiency of TabPFN is highly sensitive to context size, prompting strategies to enhance scalability and performance [35]. These include compressing training data into a compact learned representation using sketching [188] or prompt tuning techniques [227], [187], employing adaptive data selection methods to identify the most pertinent training examples for each test instance [228], [86], [41], [229], and replacing traditional quadratic attention with computationally efficient linear attention mechanisms [230] and state-space models (SSMs) [231].

Adaptation Improvements. Some approaches improve TabPFN's performance on downstream tasks by adapting the context [86] or fine-tuning specific parts of the model [39], [189], [228], [187]. TabICL [40] employs a column-then-row attention mechanism to construct fixed-dimensional embeddings of rows, which are subsequently processed by a transformer like TabPFN v1 to facilitate efficient in-context learning. EquiTabPFN [191] introduces self-attention across target components, ensuring that the arbitrary ordering of target dimensions does not influence model predictions, enhancing the performance of TabPFN v1 to some extent. [190] adapts TabPFN v2 to multi-class, high-dimensional, and large-scale data scenarios via post-processing techniques. [192] investigates various fine-tuning strategies for TabPFN and concludes that full-model fine-tuning yields the optimal performance.

C. Semantics-based General Models

By leveraging the semantic structure of tabular data, such as column names, heterogeneous tasks can be projected into a shared language space. This allows a single language model, pre-trained on diverse tabular datasets, to handle unseen tasks in a unified manner. TabuLa-8B [88] fine-tunes a Llama 3-8B LLM for tabular data prediction (classification and binned regression) using a novel packing and attention scheme for tabular prediction. GTL [89] transforms tabular datasets into an instruction-oriented language format, facilitating the continued pre-training of LLMs on instruction-oriented tabular data, which demonstrates strong performance in few-shot scenarios. GTL-S [232] unlocks the potential of GTL from a scaling perspective, revealing that scaling datasets and prediction tasks enhance generalization. [90] extends GTL by incorporating retrieval-augmented LLMs for tabular data, combined with retrieval-guided instruction-tuning for LLMs. MediTab [193] uses a data engine that leverages LLMs to consolidate tabular samples to overcome the barrier across tables with distinct schema. MediTab aligns out-domain data with the target task using a "learn, annotate, and refinement" pipeline for arbitrary tabular input in the domain without fine-tuning.

VIII. Tabular Ensemble Methods

Ensemble learning enhances generalization by combining diverse base learners. Classical methods like Random Forest and AdaBoost use bagging and boosting to ensemble decision trees. In deep tabular learning, ensembles are either joint-training ensembles that aggregate sub-networks during training or post-hoc ensembles that combine predictions from multiple pre-trained models. A major challenge is the high computational cost of training multiple models or submodels.

Joint-Training Ensembles. Joint-training ensembles integrate diverse model architectures within a single training process to improve performance and efficiency. These often combine different models, such as linear and non-linear [233] or tree-based and deep neural networks [61], and tree-mimic methods mix predictions from multiple tree nodes [57], [126].

To balance efficiency and predictive power, parameter-efficient ensembles have been proposed. For example, TabM [112] uses MLPs with batchEnsemble to generate diverse base learners without greatly increasing parameters. Similarly, BETA applies additional tuning on pre-trained TabPFN by learning multiple feature projections and aggregating results with BatchEnsemble to reduce parameter overhead [39]. Hybrid methods like LLM-Boost and PFN-Boost integrate large language models and TabPFN with gradient-boosted decision trees [234], where LLMs and PFN serve as initial learners and additional base learners are trained via boosting, combining priors with scalability.

Post-Hoc Ensembles. Post-hoc ensemble (PHE) methods combine multiple trained models to enhance robustness and accuracy. Bagging ensembles aggregate models trained with different random seeds [29], [67], improving robustness at the cost of increased computation. Recent studies show LLM-based methods produce predictions complementary to deep tabular models without attribute names [90], making them promising ensemble candidates. Perturbation-based ensembles generate diversity from a single pre-trained model without retraining. For example, TabPFN exploits feature permutation sensitivity by randomly shuffling feature order [85]. TabPFN v2 further increases diversity via random transformations such

[1]Some variants of TabPFN are not considered general tabular models, especially the latter parts, as they require additional fine-tuning steps. We place them in this subsection due to their strong relationship with TabPFN.

as varied feature encoding, quantization, categorical shuffling, SVD compression, outlier removal, and Yeo–Johnson power transforms [87], enabling effective ensemble learning without extra training. Other methods adapt ensemble ideas to TabPFN v1 for scalability: TabPFN-Bagging splits large datasets into context groups and averages predictions [39], [235], while BoostPFN treats TabPFN v1 as weak learners trained on data subsets, outperforming standard PFNs on large-scale data [235].

IX. Extensions

In this section, we briefly introduce some extensions on deep tabular methods across different complex tasks.

Anomaly Detection. Anomaly detection in tabular data identifies irregularities such as fraud or failures. Classical methods include Isolation Forest [236] and Local Outlier Factor [237]. Recent methods capture contextual relationships in high-dimensional data [238], [239]. For example, [240] maximizes mutual information between samples and masked parts. ADBench [241] benchmarks 30 algorithms on 57 datasets. LLMs have also been applied [242].

Tabular Generation. Synthetic tabular data generation addresses privacy and data scarcity. Traditional methods like Bayesian networks and GANs capture marginal distributions; newer approaches preserve complex feature dependencies. Diffusion models [243] refine synthetic data iteratively. [244] incorporates structural causal priors and benchmarks synthesis models. To balance realism and privacy, neuro-symbolic models improve trustworthy data generation [245].

Interpretability. Traditional GBDTs offer interpretability via feature importance and decision path visualization [143], [145]. The additive nature of GBDTs enables partial dependence plots [246] to visualize feature effects. NeC4.5 [200] integrates decision tree interpretability with neural network ensembles to improve performance while maintaining clarity. Recent deep tabular models also focus on interpretability. NAMs [204] combine DNN expressivity with additive model intelligibility by learning feature-specific networks. TabNet [142] employs sequential attention with feature masks for global interpretability. Variants like TabTransformer [61] visualize cross-feature attention. NODE [58], NODE-GAM [59], and DOFEN [247] generalize ensembles of oblivious trees with gradient-based optimization and hierarchical representations.

Open-Environment Tabular Machine Learning. Real-world deployments often face distribution shifts where test data differs from training distributions. Research typically categorizes these into *domain-to-domain shifts* [248], handling scenarios with or without accessible target data via transfer learning [249] or domain generalization techniques [250]. A more challenging setting is *temporal shift*, common in financial or climate data, where patterns evolve over time. Benchmarks like TableShift [248] and TabReD [149] highlight that most standard models degrade significantly in these settings. Recent solutions focus on temporal-aware evaluation protocols [251], drift-resilient architectures [110], and robust ensemble strategies [91] to maintain performance in the context of data streams.

From Tabular Data to Structured Data. Tabular data often serves as the underlying format for more complex structured domains. *Time Series* can be modeled as tabular data with temporal indices, where recent studies apply tabular methods using sliding windows or time-aware embeddings for forecasting [149], [252]. Similarly, *Relational and Graph Data* are naturally stored as linked tables. Approaches like CARTE [172] bridge this gap by modeling tables as graphs to capture entity relationships, while Graph Neural Networks (GNNs) are increasingly adapted to model interactions between rows and columns in standard tabular tasks [253].

Multi-modal Learning with Tabular Data. Text, such as feature names, enhances tabular learning (see Section VI). We focus here on tabular–image interactions, e.g., in healthcare where medical images require expert knowledge often encoded as tabular data [254]. MMCL [133] and CHARMS [131] improving predictions without tables during inference, reducing annotation needs while TIP [255] proposes a self-supervised tabular encoder for multimodal joint representation learning.

Tabular Understanding. Tabular understanding includes tasks such as *Table Detection* (TD) [256], [257], which locates tables in images, and *Table Structure Recognition* (TSR) [258], [259], which extracts cell coordinates and spanning info. *Table Question Answering* (TQA) [260], [261], [262] answers user queries from tables. Traditional OCR-based [263] and OCR-free [264] methods have advanced TD and TSR, which are simpler tasks. More complex TQA tasks have also progressed with the help of LLMs [265]. Please refer to [260], [266] for more details.

X. Conclusion

Tabular data remains a cornerstone of real-world machine learning applications, and the advancement of deep learning has opened new possibilities for effective representation learning in this domain. In this survey, we present a comprehensive overview of deep tabular representation learning, covering its background, challenges, evaluation benchmarks, and the discussion between tree-based models and DNNs. We systematically categorize existing methods into three categories—specialized, transferable, and general models—based on their generalization capabilities. In addition, we discuss ensemble techniques, extensions, and some promising future directions, such as open-environment and multimodal tabular learning. We hope this survey serves as a valuable reference for understanding the current state of the field and inspires further progress for more robust and generalizable tabular learning methods.

References

[1] B. Kovalerchuk and E. Vityaev, *Data mining in finance: advances in relational and hybrid methods*. Springer Science & Business Media, 2005.

[2] S. L. Hyland, M. Faltys, M. Hüser, X. Lyu, T. Gumbsch, C. Esteban, C. Bock, M. Horn, M. Moor, B. Rieck *et al.*, "Early prediction of circulatory failure in the intensive care unit using machine learning," *Nature medicine*, vol. 26, no. 3, pp. 364–373, 2020.

[3] C. Romero and S. Ventura, "Educational data mining: a review of the state of the art," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 40, no. 6, pp. 601–618, 2010.

[4] X. Amatriain, A. Jaimes, N. Oliver, and J. M. Pujol, "Data mining methods for recommender systems," in *Recommender systems handbook*. Springer, 2010, pp. 39–71.

[5] R. Tibshirani, T. Hastie, B. Narasimhan, and G. Chu, "Diagnosis of multiple cancer types by shrunken centroids of gene expression," *Proceedings of the National Academy of Sciences*, vol. 99, no. 10, pp. 6567–6572, 2002.

[6] O. Ivanciuc *et al.*, "Applications of support vector machines in chemistry," *Reviews in computational chemistry*, vol. 23, p. 291, 2007.

[7] N. K. Ahmed, A. F. Atiya, N. E. Gayar, and H. El-Shishiny, "An empirical comparison of machine learning models for time series forecasting," *Econometric reviews*, vol. 29, no. 5-6, pp. 594–621, 2010.

[8] M. R. Allen and D. A. Stainforth, "Towards objective probabalistic climate forecasting," *Nature*, vol. 419, no. 6903, pp. 228–228, 2002.

[9] V. Borisov, T. Leemann, K. Seßler, J. Haug, M. Pawelczyk, and G. Kasneci, "Deep neural networks and tabular data: A survey," *IEEE Trans. Neural Networks Learn. Syst.*, vol. 35, no. 6, pp. 7499–7519, 2024.

[10] C. C. Aggarwal, *Data Mining - The Textbook*. Springer, 2015.

[11] Z. Ji, Z. C. Lipton, and C. Elkan, "Differential privacy and machine learning: a survey and review," *CoRR*, vol. abs/1412.7584, 2014.

[12] M. F. Delgado, E. Cernadas, S. Barro, and D. G. Amorim, "Do we need hundreds of classifiers to solve real world classification problems?" *Journal of Machine Learning Research*, vol. 15, no. 1, pp. 3133–3181, 2014.

[13] C. Bishop, *Pattern recognition and machine learning*. Springer, 2006.

[14] T. Hastie, R. Tibshirani, and J. H. Friedman, *The Elements of Statistical Learning: Data Mining, Inference, and Prediction, 2nd Edition*. Springer, 2009.

[15] M. Mohri, A. Rostamizadeh, and A. Talwalkar, *Foundations of Machine Learning*. MIT Press, 2012.

[16] A. Voulodimos, N. Doulamis, A. Doulamis, E. Protopapadakis *et al.*, "Deep learning for computer vision: A brief review," *Computational intelligence and neuroscience*, vol. 2018, 2018.

[17] D. W. Otter, J. R. Medina, and J. K. Kalita, "A survey of the usages of deep learning for natural language processing," *IEEE transactions on neural networks and learning systems*, vol. 32, no. 2, pp. 604–624, 2020.

[18] Y. Bengio, A. Courville, and P. Vincent, "Representation learning: A review and new perspectives," *IEEE transactions on pattern analysis and machine intelligence*, vol. 35, no. 8, pp. 1798–1828, 2013.

[19] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[20] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*. MIT press, 2016.

[21] J. Donahue, Y. Jia, O. Vinyals, J. Hoffman, N. Zhang, E. Tzeng, and T. Darrell, "Decaf: A deep convolutional activation feature for generic visual recognition," in *ICML*, 2014, pp. 647–655.

[22] G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *science*, vol. 313, no. 5786, pp. 504–507, 2006.

[23] L. Van Der Maaten, "Learning a parametric embedding by preserving local structure," in *AISTATS*, 2009, pp. 384–391.

[24] M. R. Min, L. Maaten, Z. Yuan, A. J. Bonner, and Z. Zhang, "Deep supervised t-distributed embedding," in *ICML*, 2010, pp. 791–798.

[25] W. Zhang, T. Du, and J. Wang, "Deep learning over multi-field categorical data - - A case study on user response prediction," in *ECIR*, 2016, pp. 45–57.

[26] K. G. Mehrotra, C. K. Mohan, H. Huang, K. G. Mehrotra, C. K. Mohan, and H. Huang, *Anomaly detection*. Springer, 2017.

[27] F. O. Isinkaye, Y. O. Folajimi, and B. A. Ojokoh, "Recommendation systems: Principles, methods and evaluation," *Egyptian informatics journal*, vol. 16, no. 3, pp. 261–273, 2015.

[28] B. Lim and S. Zohren, "Time-series forecasting with deep learning: a survey," *Philosophical Transactions of the Royal Society A*, vol. 379, no. 2194, p. 20200209, 2021.

[29] Y. Gorishniy, I. Rubachev, V. Khrulkov, and A. Babenko, "Revisiting deep learning models for tabular data," in *NeurIPS*, 2021, pp. 18 932–18 943.

[30] D. Holzmüller, L. Grinsztajn, and I. Steinwart, "Better by default: Strong pre-tuned mlps and boosted trees on tabular data," in *NeurIPS*, 2024, pp. 26 577–26 658.

[31] H.-J. Ye, H.-H. Yin, D.-C. Zhan, and W.-L. Chao, "Revisiting nearest neighbor for tabular data: A deep tabular baseline two decades later," in *ICLR*, 2025.

[32] L. Grinsztajn, E. Oyallon, and G. Varoquaux, "Why do tree-based models still outperform deep learning on typical tabular data?" in *NeurIPS*, 2022, pp. 507–520.

[33] R. Shwartz-Ziv and A. Armon, "Tabular data: Deep learning is not all you need," *Information Fusion*, vol. 81, pp. 84–90, 2022.

[34] E. Beyazit, J. Kozaczuk, B. Li, V. Wallace, and B. Fadlallah, "An inductive bias for tabular deep learning," in *NeurIPS*, 2023, pp. 43 108–43 135.

[35] D. C. McElfresh, S. Khandagale, J. Valverde, V. P. C., G. Ramakrishnan, M. Goldblum, and C. White, "When do neural nets outperform boosted trees on tabular data?" in *NeurIPS*, 2023, pp. 76 336–76 369.

[36] H.-J. Ye, D.-C. Zhan, N. Li, and Y. Jiang, "Learning multiple local metrics: Global consideration helps," *IEEE transactions on pattern analysis and machine intelligence*, vol. 42, no. 7, pp. 1698–1712, 2019.

[37] S. M. Jesus, J. Pombal, D. Alves, A. F. Cruz, P. Saleiro, R. P. Ribeiro, J. Gama, and P. Bizarro, "Turning the tables: Biased, imbalanced, dynamic tabular datasets for ML evaluation," in *NeurIPS*, 2022, pp. 33 563–33 575.

[38] R. Kohli, M. Feurer, K. Eggensperger, B. Bischl, and F. Hutter, "Towards quantifying the effect of datasets for benchmarking: A look at tabular machine learning," in *ICLR Workshop*, 2024.

[39] S.-Y. Liu and H.-J. Ye, "Tabpfn unleashed: A scalable and effective solution to tabular classification problems," in *ICML*, 2025, pp. 40 043–40 068.

[40] J. Qu, D. Holzmüller, G. Varoquaux, and M. L. Morvan, "Tabicl: A tabular foundation model for in-context learning on large data," in *ICML*, 2025.

[41] J. Ma, V. Thomas, R. Hosseinzadeh, H. Kamkari, A. Labach, J. C. Cresswell, K. Golestan, G. Yu, M. Volkovs, and A. L. Caterini, "Tabdpt: Scaling tabular foundation models on real data," in *NeurIPS*, 2025.

[42] H.-J. Ye, S.-Y. Liu, H.-R. Cai, Q.-L. Zhou, and D.-C. Zhan, "A closer look at deep learning on tabular data," *CoRR*, vol. abs/2407.00956, 2024.

[43] Y. Gorishniy, I. Rubachev, and A. Babenko, "On embeddings for numerical features in tabular deep learning," in *NeurIPS*, 2022, pp. 24 991–25 004.

[44] T. Ucar, E. Hajiramezanali, and L. Edwards, "Subtab: Subsetting features of tabular data for self-supervised representation learning," in *NeurIPS*, 2021, pp. 18 853–18 865.

[45] D. Bahri, H. Jiang, Y. Tay, and D. Metzler, "Scarf: Self-supervised contrastive learning using random feature corruption," in *ICLR*, 2022.

[46] J. Yoon, Y. Zhang, J. Jordon, and M. van der Schaar, "VIME: extending the success of self- and semi-supervised learning to tabular domain," in *NeurIPS*, 2020, pp. 11 033–11 043.

[47] J. Wu, S. Chen, Q. Zhao, R. Sergazinov, C. Li, S. Liu, C. Zhao, T. Xie, H. Guo, C. Ji, D. Cociorva, and H. Brunzell, "Switchtab: Switched autoencoders are effective tabular learners," in *AAAI*, 2024, pp. 15 924–15 933.

[48] A. Kadra, M. Lindauer, F. Hutter, and J. Grabocka, "Well-tuned simple nets excel on tabular datasets," in *NeurIPS*, 2021, pp. 23 928–23 941.

[49] R. Wang, B. Fu, G. Fu, and M. Wang, "Deep & cross network for ad click predictions," in *ADKDD*, 2017, pp. 1–7.

[50] G. Klambauer, T. Unterthiner, A. Mayr, and S. Hochreiter, "Self-normalizing neural networks," in *NIPS*, 2017, pp. 971–980.

[51] G. Ke, J. Zhang, Z. Xu, J. Bian, and T.-Y. Liu, "Tabnn: A universal neural network solution for tabular data," 2018.

[52] R. Wang, R. Shivanna, D. Z. Cheng, S. Jain, D. Lin, L. Hong, and E. H. Chi, "DCN V2: improved deep & cross network and practical lessons for web-scale learning to rank systems," in *WWW*, 2021, pp. 1785–1797.

[53] J. Chen, K. Liao, Y. Wan, D. Z. Chen, and J. Wu, "Danets: Deep abstract networks for tabular data classification and regression," in *AAAI*, 2022, pp. 3930–3938.

[54] J. Chen, K. Liao, Y. Fang, D. Chen, and J. Wu, "Tabcaps: A capsule neural network for tabular data classification with bow routing," in *ICLR*, 2023.

[55] J. Yan, J. Chen, Q. Wang, D. Z. Chen, and J. Wu, "Team up gbdts and dnns: Advancing efficient and effective tabular prediction with tree-hybrid mlps," in *KDD*, 2024, pp. 3679–3689.

[56] C. Xu, Y.-C. Huang, J. Y.-C. Hu, W. Li, A. Gilani, H.-S. Goan, and H. Liu, "Bishop: Bi-directional cellular learning for tabular data with generalized sparse modern hopfield model," in *ICML*, 2024, pp. 55 048–55 075.

[57] S. Badirli, X. Liu, Z. Xing, A. Bhowmik, and S. S. Keerthi, "Gradient boosting neural networks: Grownet," *CoRR*, vol. abs/2002.07971, 2020.

[58] S. Popov, S. Morozov, and A. Babenko, "Neural oblivious decision ensembles for deep learning on tabular data," in *ICLR*, 2020.

[59] C.-H. Chang, R. Caruana, and A. Goldenberg, "NODE-GAM: neural generalized additive model for interpretable deep learning," in *ICLR*, 2022.

[60] W. Song, C. Shi, Z. Xiao, Z. Duan, Y. Xu, M. Zhang, and J. Tang, "Autoint: Automatic feature interaction learning via self-attentive neural networks," in *CIKM*, 2019, pp. 1161–1170.

[61] X. Huang, A. Khetan, M. Cvitkovic, and Z. S. Karnin, "Tabtransformer: Tabular data modeling using contextual embeddings," *CoRR*, vol. abs/2012.06678, 2020.

[62] Q.-L. Zhou, H.-J. Ye, L. Wang, and D.-C. Zhan, "Unlocking the transferability of tokens in deep models for tabular data," *CoRR*, vol. abs/2310.15149, 2023.

[63] J. Chen, J. Yan, Q. Chen, D. Z. Chen, J. Wu, and J. Sun, "Can a deep learning model be a sure bet for tabular prediction?" in *KDD*, 2024, pp. 288–296.

[64] A. Jeffares, T. Liu, J. Crabbé, F. Imrie, and M. van der Schaar, "Tangos: Regularizing tabular neural networks through gradient orthogonalization and specialization," in *ICLR*, 2023.

[65] H. Ye, W. Fan, X. Song, S. Zheng, H. Zhao, D. dan Guo, and Y. Chang, "Ptarl: Prototype-based tabular representation learning via space calibration," in *ICLR*, 2024.

[66] Y. Nader, L. Sixt, and T. Landgraf, "DNNR: differential nearest neighbors regression," in *ICML*, 2022, pp. 16 296–16 317.

[67] Y. Gorishniy, I. Rubachev, N. Kartashev, D. Shlenskii, A. Kotelnikov, and A. Babenko, "Tabr: Tabular deep learning meets nearest neighbors in 2023," in *ICLR*, 2024.

[68] G. Somepalli, A. Schwarzschild, M. Goldblum, C. B. Bruss, and T. Goldstein, "SAINT: Improved neural networks for tabular data via row attention and contrastive pre-training," in *NeurIPS Workshop*, 2022.

[69] I. Rubachev, A. Alekberov, Y. Gorishniy, and A. Babenko, "Revisiting pretraining objectives for tabular deep learning," *CoRR*, vol. abs/2207.03208, 2022.

[70] S. Onishi, K. Oono, and K. Hayashi, "Tabret: Pre-training transformer-based tabular models for unseen columns," *CoRR*, vol. abs/2303.15747, 2023.

[71] J. Shen, L. Li, L. M. Dery, C. Staten, M. Khodak, G. Neubig, and A. Talwalkar, "Cross-modal fine-tuning: Align then refine," in *ICML*, 2023, pp. 31 030–31 056.

[72] Y. Zhu, T. Brettin, F. Xia, A. Partin, M. Shukla, H. Yoo, Y. A. Evrard, J. H. Doroshow, and R. L. Stevens, "Converting tabular data into images for deep learning with convolutional neural networks," *Scientific Reports*, vol. 11, no. 11325, 2021.

[73] S. Lee and S.-C. Lee, "Tableye: Seeing small tables through the lens of images," *CoRR*, vol. abs/2307.02491, 2023.

[74] Z. Wang and J. Sun, "Transtab: Learning transferable tabular transformers across tables," in *NeurIPS*, 2022, pp. 2902–2915.

[75] J. Yan, B. Zheng, H. Xu, Y. Zhu, D. Z. Chen, J. Sun, J. Wu, and J. Chen, "Making pre-trained language models great on tabular prediction," in *ICLR*, 2024.

[76] C. Ye, G. Lu, H. Wang, L. Li, S. Wu, G. Chen, and J. Zhao, "Towards cross-table masked pretraining for web data mining," in *WWW*, 2024, pp. 4449–4459.

[77] S. Hegselmann, A. Buendia, H. Lang, M. Agrawal, X. Jiang, and D. Sontag, "Tabllm: few-shot classification of tabular data with large language models," in *AISTATS*, 2023, pp. 5549–5581.

[78] X. Wen, H. Zhang, S. Zheng, W. Xu, and J. Bian, "From supervised to generative: A novel paradigm for tabular deep learning with large language models," in *SIGKDD*, 2024, pp. 3323–3333.

[79] S. Han, J. Yoon, S. Ö. Arik, and T. Pfister, "Large language models can automatically engineer features for few-shot tabular learning," in *ICML*, 2024, pp. 17 454–17 479.

[80] C. Zhou, Q. Li, C. Li, J. Yu, Y. Liu, G. Wang, K. Zhang, C. Ji, Q. Yan, L. He *et al.*, "A comprehensive survey on pretrained foundation models: A history from bert to chatgpt," *International Journal of Machine Learning and Cybernetics*, pp. 1–65, 2024.

[81] Y. Liang, H. Wen, Y. Nie, Y. Jiang, M. Jin, D. Song, S. Pan, and Q. Wen, "Foundation models for time series analysis: A tutorial and survey," in *SIGKDD*, 2024, pp. 6555–6565.

[82] H.-J. Ye, Q.-L. Zhou, H.-H. Yin, D.-C. Zhan, and W.-L. Chao, "Rethinking pre-training in tabular data: A neighborhood embedding perspective," *CoRR*, vol. abs/2311.00055, 2025.

[83] D. Bonet, D. M. Montserrat, X. G. i Nieto, and A. G. Ioannidis, "Hyperfast: Instant classification for tabular data," in *AAAI*, 2024, pp. 11 114–11 123.

[84] A. Müller, C. Curino, and R. Ramakrishnan, "Mothernet: Fast training and inference via hyper-network transformers," in *ICLR*, 2025.

[85] N. Hollmann, S. Müller, K. Eggensperger, and F. Hutter, "Tabpfn: A transformer that solves small tabular classification problems in a second," in *ICLR*, 2023.

[86] V. Thomas, J. Ma, R. Hosseinzadeh, K. Golestan, G. Yu, M. Volkovs, and A. L. Caterini, "Retrieval & fine-tuning for in-context tabular models," in *NeurIPS*, 2024, pp. 108 439–108 467.

[87] N. Hollmann, S. Müller, L. Purucker, A. Krishnakumar, M. Körfer, S. B. Hoo, R. T. Schirrmeister, and F. Hutter, "Accurate predictions on small data with a tabular foundation model," *Nature*, vol. 637, no. 8045, pp. 319–326, 2025.

[88] J. Gardner, J. C. Perdomo, and L. Schmidt, "Large scale transfer learning for tabular data via language modeling," in *NeurIPS*, 2024, pp. 45 155–45 205.

[89] X. Wen, H. Zhang, S. Zheng, W. Xu, and J. Bian, "From supervised to generative: A novel paradigm for tabular deep learning with large language models," in *SIGKDD*, 2024, pp. 3323–3333.

[90] X. Wen, S. Zheng, Z. Xu, Y. Sun, and J. Bian, "Scalable in-context learning on tabular data via retrieval-augmented large language models," *CoRR*, vol. abs/2502.03147, 2025.

[91] Y. Gorishniy, A. Kotelnikov, and A. Babenko, "Tabm: Advancing tabular deep learning with parameter-efficient ensembling," *CoRR*, vol. abs/2410.24210, 2024.

[92] P. A. Gutiérrez, M. Pérez-Ortiz, J. Sánchez-Monedero, F. Fernández-Navarro, and C. Hervás-Martínez, "Ordinal regression methods: Survey and experimental study," *IEEE Trans. Knowl. Data Eng.*, vol. 28, no. 1, pp. 127–146, 2016.

[93] D. Lane, D. Scott, M. Hebl, R. Guerra, D. Osherson, and H. Zimmer, *Introduction to statistics*. Citeseer, 2003.

[94] X. Zhang, G. Ren, H. Yu, H. Yuan, H. Wang, J. Li, J. Wu, L. Mo, L. Mao, M. Hao *et al.*, "Limix: Unleashing structured-data modeling capability for generalist intelligence," *CoRR*, vol. abs/2509.03505, 2025.

[95] A. F. Karr, A. P. Sanil, and D. L. Banks, "Data quality: A statistical perspective," *Statistical Methodology*, vol. 3, no. 2, pp. 137–173, 2006.

[96] A. Sánchez-Morales, J.-L. Sancho-Gómez, J.-A. Martínez-García, and A. R. Figueiras-Vidal, "Improving deep learning performance with missing values via deletion and compensation," *Neural Comput. Appl.*, vol. 32, pp. 13 233–13 244, 2020.

[97] D. Chicco, L. Oneto, and E. Tavazzi, "Eleven quick tips for data cleaning and feature engineering," *PLoS Comput. Biol.*, vol. 18, no. 12, p. e1010718, 2022.

[98] Y. Luo, M. Wang, H. Zhou, Q. Yao, W.-W. Tu, Y. Chen, W. Dai, and Q. Yang, "Autocross: Automatic feature crossing for tabular data in real-world applications," in *KDD*, 2019, pp. 1936–1945.

[99] H. He and Y. Ma, *Imbalanced learning: foundations, algorithms, and applications*. John Wiley & Sons, 2013.

[100] J. M. Johnson and T. M. Khoshgoftaar, "Survey on deep learning with class imbalance," *J. Big Data*, vol. 6, no. 1, pp. 1–54, 2019.

[101] Y. Xie, Z. Wang, Y. Li, B. Ding, N. M. Gürel, C. Zhang, M. Huang, W. Lin, and J. Zhou, "Fives: Feature interaction via edge search for large-scale tabular data," in *SIGKDD*, 2021, pp. 3795–3805.

[102] Y. Hu, I. Fountalis, J. Tian, and N. Vasiloglou, "Annotatedtables: A large tabular dataset with language model annotations," *CoRR*, vol. abs/2406.16349, 2024.

[103] A. Klein and F. Hutter, "Tabular benchmarks for joint architecture and hyperparameter optimization," *CoRR*, vol. abs/1905.04970, 2019.

[104] P. Pokhrel, "A comparison of automl hyperparameter optimization tools for tabular data," Ph.D. dissertation, Youngstown State University, 2023.

[105] X. He, K. Zhao, and X. Chu, "Automl: A survey of the state-of-the-art," *Knowl. Based Syst.*, vol. 212, p. 106622, 2021.

[106] M. Feurer, K. Eggensperger, S. Falkner, M. Lindauer, and F. Hutter, "Auto-sklearn 2.0: Hands-free automl via meta-learning," *J. Mach. Learn. Res.*, vol. 23, no. 261, pp. 1–61, 2022.

[107] C. Mennella, U. Maniscalco, G. De Pietro, and M. Esposito, "Ethical and regulatory challenges of ai technologies in healthcare: A narrative review," *Heliyon*, vol. 10, no. 4, 2024.

[108] W. Moore and S. Frye, "Review of hipaa, part 1: history, protected health information, and privacy and security rules," *Journal of nuclear medicine technology*, vol. 47, no. 4, pp. 269–272, 2019.

[109] B. S. Caffo, F. A. D'Asaro, A. Garcez, and E. Raffinetti, "Explainable artificial intelligence models and methods in finance and healthcare," p. 970246, 2022.

[110] K. Helli, D. Schnurr, N. Hollmann, S. Müller, and F. Hutter, "Drift-resilient tabpfn: In-context learning temporal distribution shifts on tabular data," in *NeurIPS*, 2024, pp. 98 742–98 781.

[111] J. Demsar, "Statistical comparisons of classifiers over multiple data sets," *Journal of Machine Learning Research*, vol. 7, pp. 1–30, 2006.

[112] Y. Gorishniy, A. Kotelnikov, and A. Babenko, "Tabm: Advancing tabular deep learning with parameter-efficient ensembling," in *ICLR*, 2025.

[113] A. Tschalzev, L. Purucker, S. Lüdtke, F. Hutter, C. Bartelt, and H. Stuckenschmidt, "Unreflected use of tabular data repositories can undermine research quality," in *ICLR Workshop*, 2025.
[114] S. B. Rabbani, I. V. Medri, and M. D. Samad, "Attention versus contrastive learning of tabular data - A data-centric benchmarking," *CoRR*, vol. abs/2401.04266, 2024.
[115] N. Erickson, L. Purucker, A. Tschalzev, D. Holzmüller, P. M. Desai, D. Salinas, and F. Hutter, "Tabarena: A living benchmark for machine learning on tabular data," in *NeurIPS*, 2025.
[116] Y. Yang, Y. Wang, G. Liu, L. Wu, and Q. Liu, "Unitabe: A universal pretraining protocol for tabular foundation model in data science," in *ICLR*, 2024.
[117] G. Eggert, K. Huo, M. Biven, and J. Waugh, "Tablib: A dataset of 627m tables with context," *CoRR*, vol. abs/2310.07875, 2023.
[118] H. W. Jian Yang, Xuefeng Li, "DeepTables: A Deep Learning Python Package for Tabular Data," https://github.com/DataCanvasIO/DeepTables, 2022, version 0.2.x.
[119] N. Erickson, J. Mueller, A. Shirkov, H. Zhang, P. Larroy, M. Li, and A. Smola, "Autogluon-tabular: Robust and accurate automl for structured data," *CoRR*, vol. abs/2003.06505, 2020.
[120] J. R. Zaurin and P. Mulinka, "pytorch-widedeep: A flexible package for multimodal deep learning," *Journal of Open Source Software*, vol. 8, no. 86, p. 5027, Jun. 2023.
[121] S.-Y. Liu, H.-R. Cai, Q.-L. Zhou, and H.-J. Ye, "TALENT: A tabular analytics and learning toolbox," *CoRR*, vol. abs/2407.04057, 2024.
[122] T. Akiba, S. Sano, T. Yanase, T. Ohta, and M. Koyama, "Optuna: A next-generation hyperparameter optimization framework," in *KDD*, 2019, pp. 2623–2631.
[123] N. Morgan and H. Bourlard, "Generalization and parameter estimation in feedforward nets: Some experiments," in *NeurIPS*, 1989, pp. 630–637.
[124] S. Arlot and A. Celisse, "A survey of cross-validation procedures for model selection," *CoRR*, vol. abs/0907.4728, 2009.
[125] K.-Y. Chen, P.-H. Chiang, H.-R. Chou, T.-W. Chen, and T.-H. Chang, "Trompt: Towards a better deep neural network for tabular data," in *ICML*, 2023, pp. 4392–4434.
[126] S. Marton, S. Lüdtke, C. Bartelt, and H. Stuckenschmidt, "GRANDE: gradient-based decision tree ensembles for tabular data," in *ICLR*, 2024.
[127] X. Jiang, A. Margeloiu, N. Simidjievski, and M. Jamnik, "Protogate: Prototype-based neural networks with global-to-local feature selection for tabular biomedical data," in *ICML*, 2024, pp. 21 844–21 878.
[128] G. C. Cawley and N. L. C. Talbot, "On over-fitting in model selection and subsequent selection bias in performance evaluation," *Journal of Machine Learning Research*, vol. 11, pp. 2079–2107, 2010.
[129] H. Schulz-Kümpel, S. Fischer, T. Nagler, A. Boulesteix, B. Bischl, and R. Hornung, "Constructing confidence intervals for 'the' generalization error - a comprehensive benchmark study," *CoRR*, vol. abs/2409.18836, 2024.
[130] T. Nagler, L. Schneider, B. Bischl, and M. Feurer, "Reshuffling resampling splits can improve generalization of hyperparameter optimization," in *NeurIPS*, 2024.
[131] J.-P. Jiang, H.-J. Ye, L. Wang, Y. Yang, Y. Jiang, and D.-C. Zhan, "Tabular insights, visual impacts: Transferring expertise from tables to images," in *ICML*, 2024, pp. 21 988–22 009.
[132] J. Feng, Y. Yu, and Z. Zhou, "Multi-layered gradient boosting decision trees," in *NeurIPS*, 2018, pp. 3555–3565.
[133] P. Hager, M. J. Menten, and D. Rueckert, "Best of both worlds: Multimodal contrastive learning with tabular and imaging data," in *CVPR*, 2023, pp. 23 924–23 935.
[134] I. Padhi, Y. Schiff, I. Melnyk, M. Rigotti, Y. Mroueh, P. Dognin, J. Ross, R. Nair, and E. Altman, "Tabular transformers for modeling multivariate time series," in *ICASSP*, 2021, pp. 3565–3569.
[135] F. Di Martino and F. Delmastro, "Explainable ai for clinical and remote health applications: a survey on tabular and time series data," *Artif. Intell. Rev.*, vol. 56, no. 6, pp. 5261–5315, 2023.
[136] G. M. Van de Ven, T. Tuytelaars, and A. S. Tolias, "Three types of incremental learning," *Nature Machine Intelligence*, vol. 4, no. 12, pp. 1185–1197, 2022.
[137] D.-W. Zhou, Q.-W. Wang, Z.-H. Qi, H.-J. Ye, D.-C. Zhan, and Z. Liu, "Class-incremental learning: A survey," *IEEE transactions on pattern analysis and machine intelligence*, vol. 46, no. 12, pp. 9851–9873, 2024.
[138] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson, "How transferable are features in deep neural networks?" in *NIPS*, vol. 27, 2014.
[139] S. U. H. Dar, M. Özbey, A. B. Çatlı, and T. Çukur, "A transfer-learning approach for accelerated mri using deep neural networks," *Magnetic resonance in medicine*, vol. 84, no. 2, pp. 663–685, 2020.
[140] R. Basri, M. Galun, A. Geifman, D. Jacobs, Y. Kasten, and S. Kritchman, "Frequency bias in neural networks for input of non-uniform density," in *ICML*, 2020, pp. 685–694.
[141] M. Pang, K. M. Ting, P. Zhao, and Z. Zhou, "Improving deep forest by screening," *IEEE Trans. Knowl. Data Eng.*, vol. 34, no. 9, pp. 4298–4312, 2022.
[142] S. Ö. Arik and T. Pfister, "Tabnet: Attentive interpretable tabular learning," in *AAAI*, 2021, pp. 6679–6687.
[143] T. Chen and C. Guestrin, "Xgboost: A scalable tree boosting system," in *KDD*, 2016, pp. 785–794.
[144] L. O. Prokhorenkova, G. Gusev, A. Vorobev, A. V. Dorogush, and A. Gulin, "Catboost: unbiased boosting with categorical features," in *NeurIPS*, 2018, pp. 6639–6649.
[145] G. Ke, Q. Meng, T. Finley, T. Wang, W. Chen, W. Ma, Q. Ye, and T.-Y. Liu, "Lightgbm: A highly efficient gradient boosting decision tree," in *NIPS*, 2017, pp. 3146–3154.
[146] M. T. Ribeiro, S. Singh, and C. Guestrin, ""why should I trust you?": Explaining the predictions of any classifier," in *KDD*, 2016, pp. 1135–1144.
[147] S. M. Lundberg and S. Lee, "A unified approach to interpreting model predictions," in *NIPS*, 2017, pp. 4765–4774.
[148] A. Tschalzev, S. Marton, S. Lüdtke, C. Bartelt, and H. Stuckenschmidt, "A data-centric perspective on evaluating machine learning models for tabular data," in *NeurIPS Datasets and Benchmarks Track*, 2024.
[149] I. Rubachev, N. Kartashev, Y. Gorishniy, and A. Babenko, "Tabred: A benchmark of tabular machine learning in-the-wild," *CoRR*, vol. abs/2406.19380, 2024.
[150] H. A. Chipman, E. I. George, and R. E. McCulloch, "Bart: Bayesian additive regression trees," 2010.
[151] T. Duan, A. Anand, D. Y. Ding, K. K. Thai, S. Basu, A. Ng, and A. Schuler, "Ngboost: Natural gradient boosting for probabilistic prediction," in *ICML*, 2020, pp. 2690–2700.
[152] A. Radhakrishnan, D. Beaglehole, P. Pandit, and M. Belkin, "Mechanism of feature learning in deep fully connected networks and kernel machines that recursively learn features," *Science*, vol. 383, pp. 1461–1467, 2024.
[153] D. Beaglehole, D. Holzmüller, A. Radhakrishnan, and M. Belkin, "xrfm: Accurate, scalable, and interpretable feature learning models for tabular data," *CoRR*, vol. abs/2508.10053, 2025.
[154] Y. Cheng, R. Hu, H. Ying, X. Shi, J. Wu, and W. Lin, "Arithmetic feature interaction is necessary for deep tabular learning," in *AAAI*, 2024, pp. 11 516–11 524.
[155] J. Kossen, N. Band, C. Lyle, A. N. Gomez, T. Rainforth, and Y. Gal, "Self-attention between datapoints: Going beyond individual input-output pairs in deep learning," in *NeurIPS*, 2021, pp. 28 742–28 756.
[156] B. Schäfl, L. Gruber, A. Bitto-Nemling, and S. Hochreiter, "Hopular: Modern hopfield networks for tabular data," *CoRR*, vol. abs/2206.00664, 2022.
[157] H. Kim, A. Mnih, J. Schwarz, M. Garnelo, S. M. A. Eslami, D. Rosenbaum, O. Vinyals, and Y. W. Teh, "Attentive neural processes," in *ICLR*, 2019.
[158] I. Shavitt and E. Segal, "Regularization learning networks: deep learning for tabular datasets," in *NeurIPS*, 2018, pp. 1386–1396.
[159] V. Verma, T. Luong, K. Kawaguchi, H. Pham, and Q. V. Le, "Towards domain-agnostic contrastive learning," in *ICML*, 2021, pp. 10 530–10 541.
[160] C. Lee, F. Imrie, and M. van der Schaar, "Self-supervision enhanced feature selection with correlated gates," in *ICLR*, 2022.
[161] R. Levin, V. Cherepanova, A. Schwarzschild, A. Bansal, C. B. Bruss, T. Goldstein, A. G. Wilson, and M. Goldblum, "Transfer learning with deep tabular models," in *ICLR*, 2023.
[162] K. Majmundar, S. Goyal, P. Netrapalli, and P. Jain, "MET: masked encoding for tabular data," *CoRR*, vol. abs/2206.08564, 2022.
[163] E. Hajiramezanali, N. L. Diamant, G. Scalia, and M. W. Shen, "Stab: Self-supervised learning for tabular data," in *NeurIPS Workshop*, 2022.
[164] S. Chen, J. Wu, N. Hovakimyan, and H. Yao, "Recontab: Regularized contrastive representation learning for tabular data," *CoRR*, vol. abs/2310.18541, 2023.
[165] W.-W. Du, W.-Y. Wang, and W.-C. Peng, "Dora: Domain-based self-supervised learning framework for low-resource real estate appraisal," in *CIKM*, 2023, pp. 4552–4558.
[166] Y. Sui, T. Wu, J. C. Cresswell, G. Wu, G. Stein, X. S. Huang, X. Zhang, and M. Volkovs, "Self-supervised representation learning from random data projectors," in *ICLR*, 2024.
[167] T. Iwata and A. Kumagai, "Meta-learning from tasks with heterogeneous attribute spaces," in *NeurIPS*, 2020, pp. 6053–6063.

[168] L. Liu, M. M. Fard, and S. Zhao, "Distribution embedding networks for generalization from a diverse set of classification tasks," *Transactions on Machine Learning Research*, 2022.

[169] B. Zhu, X. Shi, N. Erickson, M. Li, G. Karypis, and M. Shoaran, "Xtab: Cross-table pretraining for tabular transformers," in *ICML*, 2023, pp. 43 181–43 204.

[170] Y. Zhang, K. Gong, K. Zhang, H. Li, Y. Qiao, W. Ouyang, and X. Yue, "Meta-transformer: A unified framework for multimodal learning," *CoRR*, vol. abs/2307.10802, 2023.

[171] G. Liu, J. Yang, and L. Wu, "Ptab: Using the pre-trained language model for modeling tabular data," *CoRR*, vol. abs/2209.08060, 2022.

[172] M. J. Kim, L. Grinsztajn, and G. Varoquaux, "CARTE: pretraining and transfer for tabular learning," in *ICML*, 2024, pp. 23 843–23 866.

[173] Z. Cheng, T. Xie, P. Shi, C. Li, R. Nadkarni, Y. Hu, C. Xiong, D. Radev, M. Ostendorf, L. Zettlemoyer, N. A. Smith, and T. Yu, "Binding language models in symbolic languages," in *ICLR*, 2023.

[174] N. Hollmann, S. Müller, and F. Hutter, "Large language models for automated data science: Introducing CAAFE for context-aware automated feature engineering," in *NeurIPS*, 2023, pp. 44 753–44 775.

[175] T. Zhang, S. Wang, S. Yan, L. Jian, and Q. Liu, "Generative table pre-training empowers models for tabular prediction," in *EMNLP*, 2023.

[176] T. Dinh, Y. Zeng, R. Zhang, Z. Lin, M. Gira, S. Rajput, J. yong Sohn, D. S. Papailiopoulos, and K. Lee, "LIFT: language-interfaced fine-tuning for non-language machine learning tasks," in *NeurIPS*, 2022, pp. 11 763–11 784.

[177] R. Wang, Z. Wang, and J. Sun, "Unipredict: Large language models are universal tabular predictors," *CoRR*, vol. abs/2310.03266, 2023.

[178] A. Sharma, E. Vans, D. Shigemizu, K. A. Boroevich, and T. Tsunoda, "Deepinsight: A methodology to transform a non-image data to an image for convolution neural network architecture," *Scientific reports*, vol. 9, no. 1, p. 11399, 2019.

[179] O. Bazgir, R. Zhang, S. R. Dhruba, R. Rahman, S. Ghosh, and R. Pal, "Representation of features as images with neighborhood dependencies for compatibility with convolutional neural networks," *Nature communications*, vol. 11, no. 1, p. 4391, 2020.

[180] L. Buturović and D. Miljković, "A novel method for classification of tabular data using convolutional neural networks," *BioRxiv*, pp. 2020–05, 2020.

[181] V. Gómez-Martínez, F. J. Lara-Abelenda, P. Peiro-Corbacho, D. Chushig-Muzo, C. Granja, and C. Soguero-Ruíz, "LM-IGTD: a 2d image generator for low-dimensional and mixed-type tabular data to leverage the potential of convolutional neural networks," *CoRR*, vol. abs/2406.14566, 2024.

[182] B. Sun, L. Yang, W. Zhang, M. Lin, P. Dong, C. Young, and J. Dong, "Supertml: Two-dimensional word embedding for the precognition on structured tabular data," in *CVPR Workshops*, 2019.

[183] A. Mamdouh, M. El-Melegy, S. Ali, and R. Kikinis, "Tab2visual: Overcoming limited data in tabular data classification using deep learning with visual representations," *CoRR*, vol. abs/2502.07181, 2025.

[184] D. Bonet, M. C. Cara, A. Calafell, D. M. Montserrat, and A. G. Ioannidis, "iltm: Integrated large tabular model," *CoRR*, vol. abs/2511.15941, 2025.

[185] L. Grinsztajn, K. Flöge, O. Key, F. Birkel, P. Jund, B. Roof, B. Jäger, D. Safaric, S. Alessi, A. Hayler *et al.*, "Tabpfn-2.5: Advancing the state of the art in tabular foundation models," *CoRR*, vol. abs/2511.08667, 2025.

[186] X. Zhang, D. C. Maddix, J. Yin, N. Erickson, A. F. Ansari, B. Han, S. Zhang, L. Akoglu, C. Faloutsos, M. W. Mahoney *et al.*, "Mitra: Mixed synthetic priors for enhancing tabular foundation models," in *NeurIPS*, 2025.

[187] B. Feuer, R. T. Schirrmeister, V. Cherepanova, C. Hegde, F. Hutter, M. Goldblum, N. Cohen, and C. White, "Tunetables: Context optimization for scalable prior-data fitted networks," in *NeurIPS*, 2024, pp. 83 430–83 464.

[188] B. Feuer, C. Hegde, and N. Cohen, "Scaling tabpfn: Sketching and feature selection for tabular prior-data fitted networks," *CoRR*, vol. abs/2311.10609, 2023.

[189] F. den Breejen, S. Bae, S. Cha, and S.-Y. Yun, "Fine-tuned in-context learning transformers are excellent tabular data classifiers," *CoRR*, vol. abs/2405.13396v2, 2025.

[190] H.-J. Ye, S.-Y. Liu, and W.-L. Chao, "A closer look at tabpfn v2: Strength, limitation, and extension," 2025.

[191] M. Arbel, D. Salinas, and F. Hutter, "EquitabPFN: A target-permutation equivariant prior fitted network," in *NeurIPS*, 2025.

[192] I. Rubachev, A. Kotelnikov, N. Kartashev, and A. Babenko, "On finetuning tabular foundation models," *CoRR*, vol. abs/2024.08982, 2024.

[193] Z. Wang, C. Gao, C. Xiao, and J. Sun, "Meditab: Scaling medical tabular data predictors via data consolidation, enrichment, and refinement," in *IJCAI*, 2024, pp. 6062–6070.

[194] A. Arazi, E. Shapira, and R. Reichart, "Tabstar: A foundation tabular model with semantically target-aware representations," in *NeurIPS*, 2025.

[195] M. Spinaci, M. Polewczyk, M. Schambach, and S. Thelin, "Contexttab: A semantics-aware tabular in-context learner," in *NeurIPS*, 2025.

[196] R. Bommasani, D. A. Hudson, E. Adeli, R. Altman, S. Arora, S. von Arx, M. S. Bernstein, J. Bohg, A. Bosselut, E. Brunskill *et al.*, "On the opportunities and risks of foundation models," *CoRR*, vol. abs/2108.07258, 2021.

[197] J. Goldberger, G. E. Hinton, S. Roweis, and R. R. Salakhutdinov, "Neighbourhood components analysis," in *NIPS*, vol. 17, 2004.

[198] J. T. Hancock and T. M. Khoshgoftaar, "Survey on categorical data for neural networks," *J. Big Data*, vol. 7, no. 1, p. 28, 2020.

[199] L. Breiman, "Random forests," *Machine learning*, vol. 45, pp. 5–32, 2001.

[200] Z.-H. Zhou and Y. Jiang, "Nec4. 5: Neural ensemble based c4. 5," *IEEE Transactions on knowledge and data engineering*, vol. 16, no. 6, pp. 770–773, 2004.

[201] T. Hastie and R. Tibshirani, "Generalized additive models," *Statistical science*, vol. 1, no. 3, pp. 297–310, 1986.

[202] Y. Yamada, O. Lindenbaum, S. Negahban, and Y. Kluger, "Feature selection using stochastic gates," in *ICML*, 2020, pp. 10 648–10 659.

[203] J. Yang, O. Lindenbaum, and Y. Kluger, "Locally sparse neural networks for tabular biomedical data," in *ICML*, 2022, pp. 25 123–25 153.

[204] R. Agarwal, L. Melnick, N. Frosst, X. Zhang, B. Lengerich, R. Caruana, and G. E. Hinton, "Neural additive models: Interpretable machine learning with neural nets," in *NeurIPS*, 2021, pp. 4699–4711.

[205] W.-Y. Wang, W.-W. Du, D. Xu, W. Wang, and W.-C. Peng, "A survey on self-supervised learning for non-sequential tabular data," *Machine Learning*, vol. 114, no. 1, p. 16, 2025.

[206] Y. Gal and Z. Ghahramani, "Dropout as a bayesian approximation: Representing model uncertainty in deep learning," in *ICML*, 2016, pp. 1050–1059.

[207] H.-J. Ye, D.-C. Zhan, Y. Jiang, and Z.-H. Zhou, "Rectify heterogeneous models with semantic mapping," in *ICML*, 2018, pp. 5630–5639.

[208] H.-J. Ye, L. Han, and D.-C. Zhan, "Revisiting unsupervised meta-learning via the characteristics of few-shot tasks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 45, no. 3, pp. 3721–3737, 2022.

[209] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov, "Roberta: A robustly optimized bert pretraining approach," *CoRR*, vol. abs/1907.11692, 2019.

[210] F. Mahdisoltani, J. Biega, and F. M. Suchanek, "YAGO3: A knowledge base from multilingual wikipedias," in *CIDR*, 2015.

[211] N. Hollmann, S. Müller, and F. Hutter, "Large language models for automated data science: Introducing caafe for context-aware automated feature engineering," in *NeurIPS*, 2023, pp. 44 753–44 775.

[212] S. Han, J. Yoon, S. O. Arik, and T. Pfister, "Large language models can automatically engineer features for few-shot tabular learning," in *ICML*, 2024, pp. 17 454–17 479.

[213] P. Yin, G. Neubig, W. tau Yih, and S. Riedel, "Tabert: Pretraining for joint understanding of textual and tabular data," in *ACL*, 2020, pp. 8413–8426.

[214] S.-Y. Liu, Q. Zhou, and H.-J. Ye, "Make still further progress: Chain of thoughts for tabular data leaderboard," *CoRR*, vol. abs/2505.13421, 2025.

[215] M. Chen, L. Shen, Z. Li, X. J. Wang, J. Sun, and C. Liu, "Visionts: Visual masked autoencoders are free-lunch zero-shot time series forecasters," *CoRR*, vol. abs/2408.17253, 2024.

[216] Z. Li, S. Li, and X. Yan, "Time series as images: Vision transformer for irregularly sampled time series," in *NeurIPS*, 2023, pp. 49 187–49 204.

[217] D. Ha, A. M. Dai, and Q. V. Le, "Hypernetworks," in *ICLR*, 2017.

[218] W.-L. Chao, H.-J. Ye, D.-C. Zhan, M. E. Campbell, and K. Q. Weinberger, "Revisiting meta-learning as supervised learning," *CoRR*, vol. abs/2002.00573, 2020.

[219] J. Peters, D. Janzing, and B. Schölkopf, *Elements of causal inference: foundations and learning algorithms*. The MIT Press, 2017.

[220] R. Neal, *Bayesian Learning for Neural Networks*, ser. lncs. springer, 1996.

[221] T. Iwata and A. Kumagai, "Meta-learning of semi-supervised learning from tasks with heterogeneous attribute spaces," *CoRR*, vol. abs/2311.05088, 2023.

[222] J. Ma, A. Dankar, G. Stein, G. Yu, and A. L. Caterini, "Tabpfgen - tabular data generation with tabpfn," *CoRR*, vol. abs/2406.05216, 2024.

[223] S. Ruiz-Villafranca, J. R. Gómez, J. M. C. Gómez, J. C. Mondéjar, and J. L. Martínez, "A tabpfn-based intrusion detection system for the industrial internet of things," *The Journal of Supercomputing*, vol. 80, no. 14, pp. 20 080–20 117, 2024.

[224] A. Margeloiu, A. Bazaga, N. Simidjievski, P. Liò, and M. Jamnik, "Tabmda: Tabular manifold data augmentation for any classifier using transformers with in-context subsetting," *CoRR*, vol. abs/2406.01805, 2024.

[225] S. B. Hoo, S. Müller, D. Salinas, and F. Hutter, "The tabular foundation model tabpfn outperforms specialized time series forecasting models based on simple features," *CoRR*, vol. abs/2501.02945, 2025.

[226] Y. Wu and D. L. Bergman, "Zero-shot meta-learning for tabular prediction tasks with adversarially pre-trained transformer," in *ICML*, 2025.

[227] J. Ma, V. Thomas, G. Yu, and A. L. Caterini, "In-context data distillation with tabpfn," *CoRR*, vol. abs/2402.06971, 2024.

[228] D. Xu, O. Cirit, R. Asadi, Y. Sun, and W. Wang, "Mixture of in-context prompters for tabular pfns," *CoRR*, vol. abs/2405.16156, 2024.

[229] M. Koshil, T. Nagler, M. Feurer, and K. Eggensperger, "Towards localization via data embedding for tabPFN," in *NeurIPS Workshop*, 2024.

[230] Y. Zeng, W. Kang, and A. C. Mueller, "Tabflex: Scaling tabular learning to millions with linear attention," in *NeurIPS Workshop*, 2024.

[231] S. K. Baur and S. Kim, "Exploration of autoregressive models for in-context learning on tabular data," in *NeurIPS Workshop*, 2024.

[232] Y. Sun, X. Wen, S. Zheng, X. Jia, and J. Bian, "Scaling generative tabular learning for large language models," in *NeurIPS Workshop*, 2024.

[233] H.-T. Cheng, L. Koc, J. Harmsen, T. Shaked, T. Chandra, H. Aradhye, G. Anderson, G. Corrado, W. Chai, M. Ispir, R. Anil, Z. Haque, L. Hong, V. Jain, X. Liu, and H. Shah, "Wide & deep learning for recommender systems," in *DLRS*, 2016, pp. 7–10.

[234] M. Jayawardhana, Renbo, S. Dooley, V. Cherepanova, A. G. Wilson, F. Hutter, C. White, T. Goldstein, and M. Goldblum, "Transformers boost the performance of decision trees on tabular data across sample sizes," *CoRR*, vol. abs/2502.02672v2, 2025.

[235] Y. Wang, B. Jiang, Y. Guo, Q. Gan, D. Wipf, X. Huang, and X. Qiu, "Prior-fitted networks scale to larger datasets when treated as weak learners," *CoRR*, vol. abs/2503.01256, 2025.

[236] F. T. Liu, K. M. Ting, and Z.-H. Zhou, "Isolation forest," in *ICDM*, 2008, pp. 413–422.

[237] M. M. Breunig, H.-P. Kriegel, R. T. Ng, and J. Sander, "Lof: identifying density-based local outliers," in *SIGMOD*, 2000, pp. 93–104.

[238] O. Lindenbaum, Y. Aizenbud, and Y. Kluger, "Transductive and inductive outlier detection with robust autoencoders," in *UAI*, 2024, pp. 2271–2293.

[239] A. Rozner, B. Battash, H. Li, L. Wolf, and O. Lindenbaum, "Anomaly detection with variance stabilized density estimation," in *UAI*, 2024, pp. 3121–3137.

[240] T. Shenkar and L. Wolf, "Anomaly detection for tabular data with internal contrastive learning," in *ICLR*, 2022.

[241] S. Han, X. Hu, H. Huang, M. Jiang, and Y. Zhao, "Adbench: Anomaly detection benchmark," in *NeurIPS*, 2022, pp. 32 142–32 159.

[242] A. Li, Y. Zhao, C. Qiu, M. Kloft, P. Smyth, M. Rudolph, and S. Mandt, "Anomaly detection of tabular data using llms," *CoRR*, vol. abs/2406.16308, 2024.

[243] C. Lee, J. Kim, and N. Park, "Codi: Co-evolving contrastive diffusion models for mixed-type tabular synthesis," in *ICML*, 2023, pp. 18 940–18 956.

[244] R. Tu, Z. Senane, L. Cao, C. Zhang, H. Kjellström, and G. E. Henter, "Causality for tabular data synthesis: A high-order structure causal benchmark framework," *CoRR*, vol. abs/2406.08311, 2024.

[245] R. Feinman and B. M. Lake, "Generating new concepts with hybrid neuro-symbolic models," *CoRR*, vol. abs/2003.08978, 2020.

[246] B. M. Greenwell *et al.*, "pdp: An r package for constructing partial dependence plots," *R Journal*, vol. 9, no. 1, p. 421, 2017.

[247] K.-Y. Chen, P.-H. Chiang, H.-R. Chou, C.-S. Chen, and D. T.-H. Chang, "Dofen: Deep oblivious forest ensemble," in *NeurIPS*, 2024, pp. 44 624–44 677.

[248] J. Gardner, Z. Popovic, and L. Schmidt, "Benchmarking distribution shift in tabular data with tableshift," in *NeurIPS*, 2024, pp. 53 385–53 432.

[249] C. Kim, T. Kim, S. Woo, J. Y. Yang, and E. Yang, "Adaptable: Test-time adaptation for tabular data via shift-aware uncertainty calibrator and label distribution handler," *CoRR*, vol. abs/2407.10784, 2024.

[250] S. Sagawa, P. W. Koh, T. B. Hashimoto, and P. Liang, "Distributionally robust neural networks," in *ICLR*, 2020.

[251] H.-R. Cai and H.-J. Ye, "Understanding the limits of deep tabular methods with temporal shift," *CoRR*, vol. abs/2502.20260, 2025.

[252] ——, "Feature-aware modulation for learning from temporal tabular data," in *NeurIPS*, 2025.

[253] X. Guo, Y. Quan, H. Zhao, Q. Yao, Y. Li, and W. Tu, "Tabgnn: Multiplex graph neural network for tabular data prediction," *CoRR*, vol. abs/2108.09127, 2021.

[254] W. Huang, "Multimodal contrastive learning and tabular attention for automated alzheimer's disease prediction," in *ICCV (Workshops)*, 2023, pp. 2465–2474.

[255] S. Du, S. Zheng, Y. Wang, W. Bai, D. P. O'Regan, and C. Qin, "Tip: Tabular-image pre-training for multimodal classification with incomplete data," in *ECCV*, 2024, pp. 478–496.

[256] A. Gilani, S. R. Qasim, I. Malik, and F. Shafait, "Table detection using deep learning," in *ICDAR*, 2017, pp. 771–776.

[257] M. Li, L. Cui, S. Huang, F. Wei, M. Zhou, and Z. Li, "Tablebank: Table benchmark for image-based table detection and recognition," in *LREC*, 2020, pp. 1918–1925.

[258] S. Schreiber, S. Agne, I. Wolf, A. Dengel, and S. Ahmed, "Deepdesrt: Deep learning for detection and structure recognition of tables in document images," in *ICDAR*, 2017, pp. 1162–1167.

[259] M. s. Kasem, A. Abdallah, A. Berendeyev, E. Elkady, M. Mahmoud, M. Abdalla, M. Hamada, S. Vascon, D. Nurseitov, and I. Taj-eddin, "Deep learning for table detection and structure recognition: A survey," *ACM Computing Surveys*, vol. 56, no. 12, pp. 1–41, 2024.

[260] N. Jin, J. Siebert, D. Li, and Q. Chen, "A survey on table question answering: recent advances," in *CCKS*, 2022, pp. 174–186.

[261] J.-P. Jiang, T. Zhou, D.-C. Zhan, and H.-J. Ye, "Compositional condition question answering in tabular understanding," in *ICML*, 2025, pp. 27 831–27 850.

[262] J.-P. Jiang, Y. Xia, H.-L. Sun, S. Lu, Q.-G. Chen, W. Luo, K. Zhang, D.-C. Zhan, and H.-J. Ye, "Multimodal tabular reasoning with privileged structured information," in *NeurIPS*, 2025.

[263] S. Appalaraju, B. Jasani, B. U. Kota, Y. Xie, and R. Manmatha, "Docformer: End-to-end transformer for document understanding," in *ICCV*, 2021, pp. 993–1003.

[264] J. Wan, S. Song, W. Yu, Y. Liu, W. Cheng, F. Huang, X. Bai, C. Yao, and Z. Yang, "Omniparser: A unified framework for text spotting key information extraction and table recognition," in *CVPR*, 2024, pp. 15 641–15 653.

[265] N. Deng, Z. Sun, R. He, A. Sikka, Y. Chen, L. Ma, Y. Zhang, and R. Mihalcea, "Tables as images? exploring the strengths and limitations of llms on multimodal representations of tabular data," *CoRR*, vol. abs/2402.12424, 2024.

[266] X. Fang, W. Xu, F. A. Tan, J. Zhang, Z. Hu, Y. Qi, S. Nickleach, D. Socolinsky, S. Sengamedu, and C. Faloutsos, "Large language models (llms) on tabular data: Prediction, generation, and understanding–a survey," *CoRR*, vol. abs/2402.17944, 2024.

Jun-Peng Jiang is currently working toward the PhD degree with the National Key Lab for Novel Software Technology, School of Artificial Intelligence, Nanjing University, China. His research interests lie primarily in tabular data learning, multimodal learning, and multimodal large language models.

Si-Yang Liu is currently working toward the MSc degree with the National Key Lab for Novel Software Technology, School of Artificial Intelligence, Nanjing University, China.

Hao-Run Cai is currently working toward the PhD degree with the National Key Lab for Novel Software Technology, School of Artificial Intelligence, Nanjing University, China.

Qi-Le Zhou received the MSc degree with the National Key Lab for Novel Software Technology, School of Artificial Intelligence, Nanjing University, China.

Han-Jia Ye received the PhD degree in computer science from Nanjing University, China, in 2019. He joined the School of Artificial Intelligence at Nanjing University as a faculty member in the same year and currently holds the position of associate professor. His research focuses primarily on machine learning, with interests in representation learning, model reuse, and meta-learning. He has served as the Tutorial Co-Chair for SDM 2023. Additionally, he participates as area chairs in conferences, such as ICML, NeurIPS, and CVPR, and others.