`Latte`: Transfering LLMs' Latent-level Knowledge for Few-shot Tabular Learning

Ruxue Shi, **Hengrui Gu**, **Hangting Ye**, **Yiwei Dai**, **Xu Shen** and **Xin Wang**
Jilin University, Changchun, China
{shirx24, guhr22, yeht2118, daiyw23, shenxu23}@mails.jlu.edu.cn, xinwang@jlu.edu.cn

Abstract

Few-shot tabular learning, in which machine learning models are trained with a limited amount of labeled data, provides a cost-effective approach to addressing real-world challenges. The advent of Large Language Models (LLMs) has sparked interest in leveraging their pre-trained knowledge for few-shot tabular learning. Despite promising results, existing approaches either rely on test-time knowledge extraction, which introduces undesirable latency, or text-level knowledge, which leads to unreliable feature engineering. To overcome these limitations, we propose **`Latte`**, a training-time knowledge extraction framework that transfers the latent prior knowledge within LLMs to optimize a more generalized downstream model. **`Latte`** enables general knowledge-guided downstream tabular learning, facilitating the weighted fusion of information across different feature values while reducing the risk of overfitting to limited labeled data. Furthermore, **`Latte`** is compatible with existing unsupervised pre-training paradigms and effectively utilizes available unlabeled samples to overcome the performance limitations imposed by an extremely small labeled dataset. Extensive experiments on various few-shot tabular learning benchmarks demonstrate the superior performance of **`Latte`**, establishing it as a state-of-the-art approach in this domain. Our code is available at https://github.com/ruxueshi/Latte.git.

1 Introduction

Given the high cost of sample annotation, both in terms of financial and temporal resources, learning from a limited number of labeled samples presents a more cost-effective approach for adapting machine learning models to practical deployment [Snell *et al.*, 2017; Wang *et al.*, 2020; Oreshkin *et al.*, 2018; Wang *et al.*, 2021a]. This scenario, commonly referred to as **few-shot learning**, has recently attracted significant research attention over various domains, including computer vision (CV) [Chen *et al.*, 2019] and tabular learning [Nam *et al.*, 2023; Shi *et al.*, 2025]. However, without adequate supervisory signals, traditional supervised learning in this situation cannot effectively yield desirably parameterized models, as its performance heavily depends on the statistical convergence of a large number of labeled samples. This challenge becomes more intractable in the context of tabular data, owing to the inherent scarcity of labeled tabular data [Liu *et al.*, 2024] in many real-world applications, such as fraud detection [Cao, 2022] and disease diagnosis [Shailaja *et al.*, 2018].

With the rapid advancement of Large Language Models (LLMs), several pioneering studies have explored their potential in few-shot tabular learning and achieved promising results: The extensive knowledge encoded during pre-training makes LLMs a natural choice for analyzing tabular data, as it provides a rich prior understanding of the relationships between diverse tabular features and the task objective. For instance, knowledge regarding the increased risk of certain diseases in older individuals can be helpful in identifying truly predictive tabular features in disease prediction tasks [Han *et al.*, 2024a]. However, most existing LLM-based methods [Slack and Singh, 2023; Hegselmann *et al.*, 2023] adopt test-time knowledge extraction strategies, where each test sample is directly fed into a pre-trained or fine-tuned LLM as a query to obtain its final prediction. This approach necessitates at least one LLM inference per sample, leading to undesirable response latency and increased computational costs.

To solve above limitations, FeatLLM [Han *et al.*, 2024a] is first proposed by designing a training-time extraction strategy. Specifically, in this framework, the LLM functions as a "feature engineer" to reduce the inherent redundancy in the original tabular features: Before each tabular learning task begins, FeatLLM instructs the "feature engineer" to analyze the metadata (including task and feature descriptions) and generate insightful rules for each answer class. These rules guide the subsequent feature engineering, i.e., transforming heterogeneous column features of a tabular dataset into unified binary features. This approach reduces the reliance on complex model architectures, thereby alleviating the urgent need for a large number of labeled samples in few-shot settings.

However, FeatLLM still faces several bottlenecks that limit its performance potential: ❶ **Unreliable knowledge extraction**: FeatLLM prompts LLMs to generate textual rules as the criteria for feature engineering. However, this kind of text-level knowledge, i.e. textual responses auto-regressively generated by LLMs, often suffers from hallucinations and might include plausible yet factually wrong knowledge [Duan *et al.*, 2024; Perković *et al.*, 2024; Li *et al.*, 2024], thereby leading to unreliable feature engineering. In addition, [Zhou *et al.*, 2024; Chen *et al.*, 2024] have demonstrated that the latent-level knowledge, i.e. hidden states directly hooked from LLMs, is more informative and discriminative than the text-level knowledge due to the randomness and information loss during the next-token sampling; **❷ Overlook of unlabeled tabular data**: Constructing pseudo-supervision (e.g., by data clustering) based on unlabeled data has been proven to be an effective practice to facilitate few-shot tabular learning [Nam *et al.*, 2023]. However, the tabular learning framework of FeatLLM cannot make use of these cluster pseudo-labels, as its feature engineering process is designed exclusively for ground-truth classes, implying its performance upper-bound is strictly constrained by the extremely limited labeled samples.

Contributions: To accommodate the above bottlenecks, we propose a training-time knowledge extraction framework to transfer LLMs' **Lat**en**t**-level Knowledg**e** for few-shot tabular learning (**`Latte`**). Unlike FeatLLM, which treats the LLM as a "feature engineer", **`Latte`** positions the LLM as a "teacher" to instruct the training process of the downstream tabular learning model. Specifically, for a tabular learning task, **`Latte`** inputs its metadata (i.e., task and feature descriptions) into the LLM and retrieves the hidden states produced by the final transformer layer as task-relevant knowledge. To distill this latent-level knowledge into downstream training, we propose two key components: 1) a semantic-aware tabular encoder that integrates feature semantics into the representation of feature values; and 2) a knowledge adapter that takes this latent-level knowledge as the guideline to weightedly fuse information from different feature values, thereby constructing more predictive row embeddings. Together, these two components enable a general knowledge-guided training process, mitigating the risk of overfitting to limited labeled samples—a persistent challenge in few-shot settings—and thereby promoting convergence toward a more generalized model checkpoint. (**Bottleneck ❶**). In addition, **`Latte`** includes an additional unsupervised pre-training stage, making full use of the available unlabeled samples to provide a robust parameter initialization for the subsequent knowledge-guided training (**Bottleneck ❷**).

We conducted extensive experiments to validate **`Latte`** 's effectiveness on few-shot tabular learning tasks. **`Latte`** consistently outperforms all competing methods by a significant margin across various datasets and prediction tasks. Comprehensive ablation studies further highlight the superiority of latent-level knowledge over text-level knowledge, as well as the effectiveness of the individual components in this framework.

2 Related Work

2.1 Classical Tabular Prediction

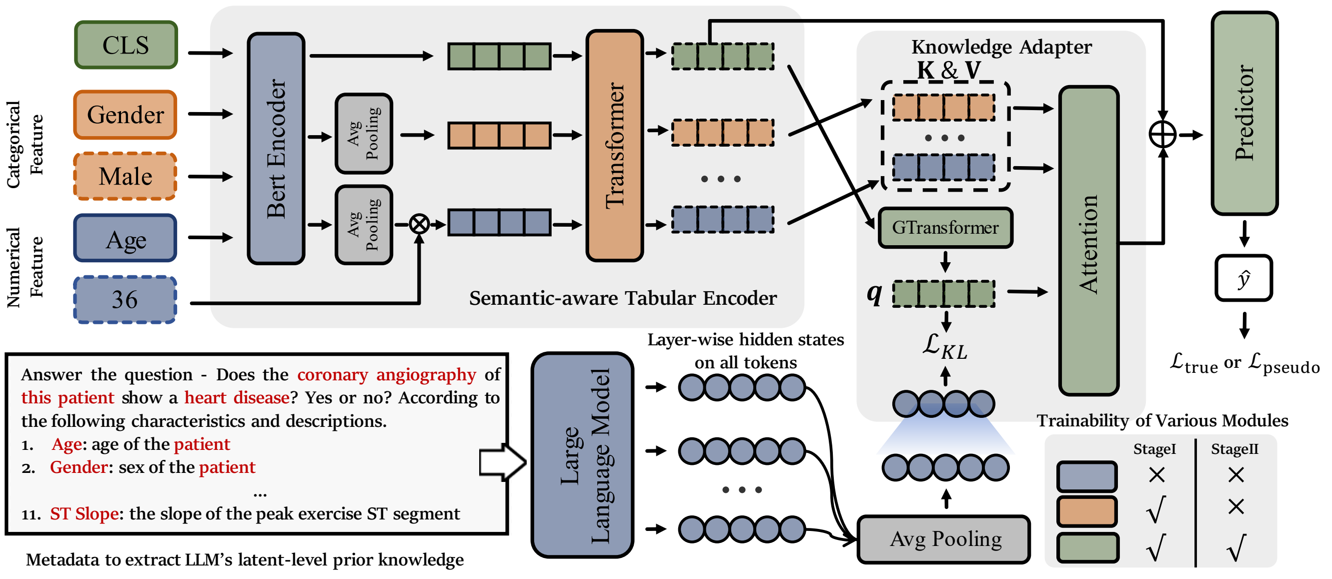
A wide range of machine learning methods have been developed for tabular data. For modeling linear relationships, logistic regression (LR) [LaValley, 2008] is commonly used. In contrast, tree-based models, such as decision trees (DT) [Loh, 2011] and ensemble methods like XGBoost [Chen and Guestrin, 2016], random forests [Breiman, 2001], CatBoost [Prokhorenkova *et al.*, 2018], and LightGBM [Ke *et al.*, 2017], are preferred for modeling non-linear relationships. With the rise of deep learning, there has been growing interest in applying neural networks to tabular data. These approaches can be broadly categorized into four types: (1) Standard Neural Networks: This group includes methods like SNN [Klambauer *et al.*, 2017], AutoInt [Song *et al.*, 2019], and DCN V2 [Wang *et al.*, 2021b]. (2) Hybrid Methods: These combine decision trees and neural networks for end-to-end training, with notable examples being NODE [Popov *et al.*, 2019], GrowNet [Badirli *et al.*, 2020], TabNN [Ke *et al.*, 2018], and DeepGBM [Ke *et al.*, 2019]. (3) Transformer-Based Methods: These models, such as TabNet [Arik and Pfister, 2021], TabTransformer [Huang *et al.*, 2020], and FT Transformer [Gorishniy *et al.*, 2021], utilize attention mechanisms to learn from tabular data. (4) Representation Learning Methods: These methods leverage self-supervised and semi-supervised learning to extract meaningful representations, with prominent examples including VIME [Yoon *et al.*, 2020], SCARF [Bahri *et al.*, 2021], SAINT [Somepalli *et al.*, 2021], and Recontab [Chen *et al.*, 2023].

2.2 Few-shot Tabular Prediction

Few-shot tabular prediction holds significant promise in fields like agriculture, industry, and finance, where obtaining and annotating data can be challenging. As a result, this area has garnered increasing attention. However, traditional tabular prediction methods often require large amounts of annotated data and tend to perform poorly in few-shot scenarios. To address these challenges, two main approaches have emerged: (1) TNNs-based methods: These approaches generate tasks from unlabeled data for meta-learning and use a small amount of labeled data to create prototypes for tabular classification tasks [Nam *et al.*, 2023]. (2) LLMs-based methods: These rely on the inference capabilities of large language models. For example, the In-context method [Wei *et al.*, 2022] directly converts tabular data into text prompts for the LLMs. TABLET [Slack and Singh, 2023] further enhances this by adding task-specific instructions to the prompts, improving the model's ability to reason about tabular. FeatLLM [Han *et al.*, 2024b] uses LLMs as a feature engineer to automatically perform feature extraction, which is then used to train specialized classification networks. Alternatively, LLMs can be fine-tuned using a small number of labeled samples, as demonstrated by methods like TabLLM [Hegselmann *et al.*, 2023]. Additionally, GTL [Wen *et al.*, 2024] fine-tunes LLMs using a large amount of tabular data to obtain a tabular foundation model.

3 Method

In this section, we innovatively propose **`Latte`** to transfer LLMs' latent-level knowledge for few-shot tabular learning. As shown in Fig. 1, **`Latte`** first utilizes a semantic-aware tabular encoder to obtain the representation of feature values integrated with their corresponding semantics (Section 3.2).



CLS
Categorical Feature
Gender
Male
Numerical Feature
Age
36
Bert Encoder
Avg Pooling
Avg Pooling
Transformer
Semantic-aware Tabular Encoder
Knowledge Adapter
K & V
GTransformer
q
Attention
Predictor
$\hat{y}$
$\mathcal{L}_{KL}$
$\mathcal{L}_{\text{true}}$ or $\mathcal{L}_{\text{pseudo}}$
Answer the question - Does the coronary angiography of this patient show a heart disease? Yes or no? According to the following characteristics and descriptions.
1. Age: age of the patient
2. Gender: sex of the patient
...
11. ST Slope: the slope of the peak exercise ST segment
Metadata to extract LLM's latent-level prior knowledge
Large Language Model
Layer-wise hidden states on all tokens
Avg Pooling
Trainability of Various Modules
StageI
StageII

	StageI	StageII
🟦	×	×
🟧	√	×
🟩	√	√

Figure 1: Overview of our proposed **Latte**. This framework begins with the extraction of task-relevant knowledge from LLMs. Knowledge adapters are then employed to guide pre-training on unlabeled data, followed by semantic-aware fine-tuning using few-shot labeled samples. Here we refer to the **unsupervised pre-training** and the **semantic-aware fine-tuning** as StageI and StageII, respectively.

Next, **Latte** leverages task metadata to extract task-relevant knowledge from LLMs (Section 3.3) and transfers this latent-level knowledge into downstream training via a specially designed knowledge adapter (Section 3.4). Finally, we outline the overall downstream training process (Section 3.5).

3.1 Problem Formulation

In this section, we formally define the problem settings for few-shot tabular learning: Consider a tabular dataset $D = \{D_l, D_u, M\}$, where D_l represents the set of labeled samples, D_u the set of unlabeled samples, and M the corresponding metadata, respectively. Specifically, each sample in the tabular dataset is in this form (x, y), where x denotes all feature values of a single row, and y represents its corresponding label. The labeled set $D_l = \left\{(x_l^i, y_l^i)\right\}_{i=1}^{N_l}$ contains N_l labeled instances, and the unlabeled set $D_u = \left\{x_u^i\right\}_{i=1}^{N_u}$ consists of N_u unlabeled instances, with $N_l \ll N_u$. Metadata $M = \{F, T\}$ provides textual descriptions of the task objective and the definitions of features in the dataset, where $F = \{f_j : d_j\}_{j=1}^{d}$ represents the column names and their descriptions, and T denotes the task description. Formally, the goal is to train a downstream model $f_\theta : \mathcal{X} \to \mathcal{Y}$, parameterized by θ, to map the row feature space $\mathcal{X}$ to the label space $\mathcal{Y}$, where $\mathcal{Y}$ could be $\{0, 1\}^C$ with C classes or $\mathbb{R}$, depending on the specific task (classification or regression).

3.2 Semantic-aware Tabular Encoder

Tabular data presents unique challenges compared to text and images due to its inherent heterogeneity, permutation invariance, and the rich semantic information provided by column names. These characteristics of tabular complicate the encoding process. Traditional feature value encoders struggle with limited labeled samples, making it difficult to infer the semantic meaning of feature names implicitly due to the scarcity of supervisory signals. To address this limitation and better handle the heterogeneity of tabular data, we propose an encoding method that explicitly incorporates feature semantic information into the representation of feature values:

$$h_j = \begin{cases} \text{Average}(\text{BERT}([f_j, x_j])), & \text{for Cat,} \\ \text{Average}(\text{BERT}([f_j])) \times x_j, & \text{for Num,} \end{cases} \tag{1}$$

Here, f_j denotes the j-th feature (e.g., "Gender"), x_j is its corresponding value (e.g., "Male"), and $\times$ represents element-wise multiplication between numerical features and values, because BERT's tokenizer typically splits numerical values into subcomponents, which can hinder its effectiveness for encoding numerical features. This approach generates effective tabular semantic representations for both categorical and numerical features, resulting in a feature set $H = \{h_j\}_{j=1}^{n}$, where n is the total number of features. To obtain a global representation of the sample, we introduce an extra token [CLS] to the feature set H. This token serves as an aggregate representation of the entire sample, capturing the overall context and facilitating downstream tasks.

The interaction between tabular features is crucial for capturing meaningful patterns. For instance, the relationship between weight and blood pressure (BP) can provide valuable insights into cardiovascular health, which is vital for heart disease classification. To model these interactions while preserving the inherent permutation invariance of tabular data, we employ a Transformer architecture without positional embeddings to further refine the quality of feature value embeddings:

$$\hat{H} = \text{Transformer}(H), \tag{2}$$

Here, $\hat{H} = \{\hat{h}_{\text{CLS}}, \hat{h}_j\}_{j=1}^{n}$, where $\hat{h}_{\text{CLS}}$ and $\{\hat{h}_j\}_{j=1}^{n}$ refers to the global representation and the set of tabular feature representations, respectively. These capture the implicit relationships between features, enabling the model to effectively understand and leverage feature interactions.

3.3 Task-relevant Prior Knowledge Extraction

The above semantic-aware tabular encoder effectively processes heterogeneous tabular data and incorporates its semantic information into the representations of feature values. However, when labeled data is limited, capturing task-specific semantics becomes more challenging. Thanks to large-scale pretraining, LLMs encode substantial general knowledge within their parameters. For a given a tabular learning task, to effectively obtain the task-relevant knowledge within LLMs, we formulate the following knowledge extraction process: Based on the metadata M of the given tabular dataset D, we elaborately craft the input prompt, describing the task objective and feature semantics (See Fig. 1 for an example of input prompt), in order to retrieve task-relevant knowledge from the pre-trained LLM. Specifically, after the LLM processes this prompt, we hook the hidden states produced at the last transformer layer of all input tokens and apply average pooling to them. This operation results in a dense vector that encapsulates the LLM's prior understanding of the task, serving as our latent-level knowledge for subsequent knowledge-guided downstream training. The process can be formally described as follows:

$$[h_1, h_2, ..., h_i, ...] = \text{LLM}(M), \tag{3}$$

where h_i represents the last-layer hidden states of the i-th token after processing the input prompt. We obtain the prompt representation, i.e. the latent-level knowledge, by applying average pooling to the hidden states of all tokens:

$$h_M = \text{Average}([h_1, h_2, ..., h_i, ...]). \tag{4}$$

where $\text{Average}(\cdot)$ represents the average pooling operator and h_M denotes the representation of the input prompt. We regard this dense vector, which contains abundant prior understandings of the given tabular dataset, as the useful latent-level knowledge for subsequent model training.

3.4 Knowledge Adapter

The semantic spaces of LLMs and the BERT-based Semantic-aware Tabular Encoder differ significantly in both dimension and representation, making it challenging to directly integrate the Latent-level knowledge from the LLM into our model. To address this, we design a knowledge adapter that aligns these two semantic spaces and facilitates the transfer of LLM-generated knowledge into our model. We introduce a novel GTransformer to generate the global query vector q:

$$q = \text{GTransformer}(\hat{h}_{CLS}, \{\hat{h}_j\}_{j=1}^{n}, \{\hat{h}_j\}_{j=1}^{n}), \tag{5}$$

where both $\hat{h}_{CLS}$ and $\{\hat{h}_j\}_{j=1}^{n}$ are representations encoded by the Semantic-Aware Tabular Encoder. To distill task-related semantic knowledge from the LLM into the global query vector q and obtain the task-relevant global query vector q_{LLM}, we apply the following knowledge distillation formula:

$$q_{\text{LLM}} = \text{KL}\left(\mathbf{W_0} h_M/\tau, q/\tau\right), \tag{6}$$

where τ is the temperature, the matrix $\mathbf{W_0}$ is initialized randomly using Kaiming initialization, which aligns the dimensions of h_M and q. Meanwhile, through the attention mechanism, we obtain the LLMs semantic knowledge guided task-relevant semantic representation h_{LLM} as:

$$a_i = \text{softmax}\left(\frac{q_{\text{LLM}}\mathbf{K}_i^{\top}}{\sqrt{d}}\right), \tag{7}$$

$$h_{\text{LLM}} = \sum_{i=0}^{n} a_i \mathbf{V}_i, \tag{8}$$

where the $\hat{H}$ generated by the Semantic-aware Tabular Encoder is directly used as $\mathbf{K}$ and $\mathbf{V}$, ensuring that the semantic information of the sample remains unaffected and d is the feature dimension of q_{LLM}. Finally, we combine the general tabular semantics $\hat{h}_{CLS}$ from the Semantic-aware Tabular Encoder with the task-relevant semantic representation h_{LLM} to obtain a high-quality representation $\hat{h}_{\text{LLM}}$, which effectively integrates both task-relevant and general tabular semantic information:

$$\hat{h}_{\text{LLM}} = \eta \cdot g(h_{\text{LLM}}; \phi) + (1-\eta) \cdot \hat{h}_{CLS}, \tag{9}$$

where $g(\cdot;\phi)$ is a learnable transformation designed to align the semantic spaces of h_{LLM} and $\hat{h}_{CLS}$ and η is a constant that allows for fine-tuning the influence of the LLM's prior knowledge on the prediction process, ensuring an optimal balance between the model's learned features and the external knowledge introduced.

3.5 Model Training

To fully leverage the available unlabeled samples and ensure robust parameter initialization for subsequent knowledge-guided training, we include an unsupervised pre-training stage, in addition to downstream task fine-tuning. During the pre-training stage, we generate N-way K-shot meta-training tasks using unlabeled datasets and perform meta-learning, guided by task-relevant knowledge extracted from LLMs. To obtain the meta-training task, we first generate pseudo-labels $\tilde{\mathbf{y}}$ for the unlabeled data using the following cluster procedure:

$$\min_{\mathbf{C}\in\mathbb{R}^{d\times k}} \frac{1}{N_u} \sum_{i=1}^{N_u} \min_{\tilde{\mathbf{y}}\in\{0,1\}^k} \left\| (\hat{\mathbf{h}}_{\text{CLS}} \odot \mathbf{m}) - \mathbf{C}\tilde{\mathbf{y}} \right\|_2^2 \quad \text{such that} \quad \tilde{\mathbf{y}}^{\top}\mathbf{1}_k = 1, \tag{10}$$

where k represents the number of centroids, $\mathbf{1}_k$ is a vector of ones, $\mathbf{C}$ is the centroid matrix, and $\mathbf{m}$ is random noise used to corrupt features, aiding in the learning of robust embeddings for downstream tasks. Next, we randomly select k samples from each cluster to create an N-way K-shot meta-training task. By introducing different noise $\mathbf{m}$, we generate a variety

Data	# of samples	# of cat/num feature	Task type
Bank	45211	8/8	classification
Blood	748	0/4	classification
Credit-g	1000	12/8	classification
Diabetes	768	0/8	classification
Heart	918	4/7	classification
Myocardial	1700	94/17	classification
Abalone	4177	1/7	regression
Boston	506	2/11	regression
Cholesterol	303	9/4	regression

Table 1: Basic information of each dataset used in our experiments.

of meta-tasks. Finally, we compute the meta-learning loss as follows:

$$\mathcal{L}_{\text{meta}} = \mathcal{L}_{KL} + \mathcal{L}_{\text{pseudo}}, \tag{11}$$

where $\mathcal{L}_{KL} = \text{KL}\left(W_0 h_M/\tau, q/\tau\right)$ denotes the KL divergence loss, which is used to guide the distillation of task-relevant knowledge from LLMs and $\mathcal{L}_{\text{pseudo}}$ is the loss associated with the pseudo-labels, given by $\mathcal{L}_{\text{pseudo}} = -\tilde{\mathbf{y}}\log(\text{MLP}(\hat{h}_{\text{LLM}}))$. This stage of training enables the model to learn task-relevant semantic information under the guidance of LLM task-relevant knowledge, preparing the model for adapting to downstream tasks. Therefore, during the downstream task fine-tuning stage, Our model is capable of rapidly generalizing to downstream tasks with the guidance of limited labeled data. The loss is computed as:

$$\mathcal{L}_{\text{pre}} = \mathcal{L}_{KL} + \mathcal{L}_{\text{true}}, \tag{12}$$

where $\mathcal{L}_{\text{true}}$ depends on the type of task:

$$\mathcal{L}_{\text{true}} = \begin{cases} -y\log(\hat{y}), & \text{for classification,} \\ (y-\hat{y})^2, & \text{for regression.} \end{cases} \tag{13}$$

This final step refines the model's performance on the specific task using the limited number of labeled examples available.

4 Experiment

In this section, we first introduce the setup for few-shot tabular data prediction and provide the implementation details. We then evaluate the proposed model against with baseline, including traditional machine learning methods, few-shot tabular algorithms, and LLM-based frameworks. Additionally, we conduct extensive ablation studies to test the effectiveness of the proposed modules.

4.1 Experimental Settings

Dataset. As shown in Table 1, we evaluate the proposed method using nine real-world datasets, including six classification tasks and three regression tasks. Each dataset provides metadata such as a clear description for each attribute and task. Refer to Appendix A for details of the dataset

Baselines. We compare the proposed method against ten baseline models. The first is conventional tabular learning approaches: (1) Logistic Regression (LogReg) [LaValley, 2008], (2) XGBoost [Chen and Guestrin, 2016], and (3) Random Forest [Breiman, 2001]. The next two baselines leverage unlabeled datasets: (4) SCARF [Bahri *et al.*, 2021] and (5) STUNT [Nam *et al.*, 2023], though it is worth noting that in real-world scenarios while obtaining unlabeled data is typically straightforward, the process of labeling it can be expensive and time-consuming. Another recent baseline, (6) TabPFN [Hollmann *et al.*, 2023], generates synthetic datasets with diverse distributions to pretrain the model. The final group of baselines utilizes LLMs: (7) In-context learning [Wei *et al.*, 2022], which embeds few-shot training samples directly into the input prompt without fine-tuning; (8) TABLET [Slack and Singh, 2023], which enriches the prompt with additional hints through an external classifier using rule sets and prototypes to improve inference quality; (9) TabLLM [Hegselmann *et al.*, 2023], which applies parameter-efficient tuning techniques such as IA3 to train the LLMs with few-shot samples; and (10) FeatLLM [Han *et al.*, 2024b], which treats the LLMs as a feature engineer, automating feature extraction and using the engineered features for few-shot tabular classification.

Implementation details. We obtain task-relevant knowledge from the LLaMA2 series models [Touvron *et al.*, 2023]. The hidden layer dimensions of the Feed Forward Network (FFN) and other components are set to 256 and 128, respectively. For the Semantic-aware Tabular Encoder, we use a 2-layer, 8-head transformer, while the knowledge adapter consists of a 2-layer, 2-head transformer. Dropout is set to 0.1. During the meta-learning phase, The Adam optimizer is used for training, with the learning rate set to 1e-4. The activation vector in the LLMs is obtained from the 30 layers. For the fine-tuning phase, the learning rate is reduced to 1e-5 due to the limited number of labeled samples. For evaluation, we use the Area Under the Curve (AUC) metric for classification tasks and Mean Squared Error (MSE) for regression tasks.

4.2 Results and Analysis

`Latte` offers an effective and comprehensive solution for few-shot tabular prediction. For **Effectiveness:** As shown in Table 2, **`Latte`** outperforms or comparable of the current state-of-the-art baseline across all datasets and shot setting. For example, in classification tasks, our method exceeds the performance of the state-of-the-art method FeatLLM in all shot settings, achieving an average performance improvement of 4.22%. This improvement is primarily due to **`Latte`**'s ability to leverage both unlabeled data and task-related knowledge from LLM, which is something that existing methods cannot achieve.

For **Comprehensiveness: `Latte`** directly applicable to regression tasks without the need for modifications. Detailed experimental results can be found in Appendix B. In contrast, existing few-shot learning methods are often designed specifically for classification tasks and cannot be directly applied to regression. While large language models (LLMs) can handle regression tasks, they tend to produce inaccurate predictions, often due to spurious patterns or "hallucination."

4.3 Ablation Study

Each module in `Latte` plays a crucial role. We conduct ablation studies to assess the contribution of key components to the model's performance. Specifically, we evaluate four factors: (1) **SaTE** explores the decision of whether to use a semantic-aware table encoder. If a semantic-aware approach is not employed, a non-semantic-aware encoder from

data	shot	LogReg	XGBoost	RandomForest	SCARF	TabPFN	STUNT	In-context	TABLET	TabLLM	FeatLLM	`Latte`
	4	$70.54_{3.83}$	$50.00_{0.00}$	$70.85_{2.02}$	$59.38_{3.42}$	$67.33_{15.29}$	$\mathbf{88.27}_{3.32}$	$60.76_{4.00}$	$68.19_{11.17}$	$59.74_{4.49}$	$75.66_{4.59}$	$\underline{86.10_{5.42}}$
	8	$78.12_{10.59}$	$55.88_{3.98}$	$79.43_{4.28}$	$74.35_{6.93}$	$77.89_{2.34}$	$\underline{88.78_{2.38}}$	$65.46_{3.77}$	$69.85_{10.82}$	$70.14_{7.91}$	$79.46_{2.16}$	$\mathbf{89.59}_{4.09}$
Heart ↑	16	$83.02_{3.70}$	$78.62_{7.14}$	$83.45_{3.95}$	$83.66_{5.91}$	$81.45_{5.05}$	$\underline{89.13_{2.10}}$	$67.00_{7.83}$	$68.39_{11.73}$	$81.72_{3.92}$	$83.71_{1.88}$	$\mathbf{92.10}_{2.79}$
	32	$84.84_{3.53}$	$87.11_{1.22}$	$88.77_{2.36}$	$88.45_{2.43}$	$88.00_{2.34}$	$\underline{89.65_{3.04}}$	$71.94_{3.88}$	$71.90_{9.07}$	$87.43_{2.32}$	$87.19_{3.66}$	$\mathbf{93.00}_{2.68}$
	64	$89.74_{3.30}$	$89.99_{3.82}$	$89.74_{2.62}$	$\underline{90.97_{1.55}}$	$90.20_{2.57}$	$89.62_{3.16}$	OOW	OOW	$89.78_{2.59}$	$88.08_{4.11}$	$\mathbf{93.27}_{2.33}$
	4	$51.25_{3.85}$	$50.00_{0.00}$	$51.91_{4.49}$	$47.70_{4.10}$	OOW	$52.77_{2.01}$	OOW	OOW	OOW	$\underline{52.87_{3.44}}$	$\mathbf{55.12}_{2.86}$
	8	$55.34_{1.11}$	$55.63_{2.92}$	$52.77_{5.83}$	$49.37_{3.41}$	OOW	$55.40_{4.41}$	OOW	OOW	OOW	$\underline{56.22_{1.64}}$	$\mathbf{59.85}_{3.25}$
Myocardial ↑	16	$60.00_{5.16}$	$56.55_{12.22}$	$54.16_{4.53}$	$54.31_{1.42}$	OOW	$\underline{61.22_{3.45}}$	OOW	OOW	OOW	$55.32_{9.15}$	$\mathbf{61.96}_{4.49}$
	32	$58.63_{0.96}$	$57.31_{5.31}$	$46.43_{5.02}$	$53.52_{0.74}$	OOW	$\underline{60.76_{1.58}}$	OOW	OOW	OOW	$60.02_{4.02}$	$\mathbf{62.40}_{6.40}$
	64	$57.04_{1.94}$	$56.18_{2.85}$	$55.00_{4.73}$	$54.41_{2.00}$	OOW	$59.79_{0.56}$	OOW	OOW	OOW	$\underline{61.47_{3.91}}$	$\mathbf{63.84}_{1.76}$
	4	$63.70_{3.87}$	$50.00_{0.00}$	$60.59_{3.90}$	$58.53_{5.49}$	$63.19_{11.60}$	$56.34_{12.82}$	$61.38_{1.30}$	$58.11_{6.29}$	$62.51_{8.95}$	$\underline{70.45_{3.69}}$	$\mathbf{74.33}_{2.57}$
	8	$72.52_{3.21}$	$58.78_{10.54}$	$61.74_{9.91}$	$55.28_{11.88}$	$62.81_{7.84}$	$63.01_{8.78}$	$69.57_{13.35}$	$69.08_{6.00}$	$63.19_{5.79}$	$\underline{75.85_{6.66}}$	$\mathbf{79.91}_{2.88}$
Bank ↑	16	$77.51_{3.09}$	$70.34_{5.86}$	$65.67_{10.43}$	$65.81_{1.79}$	$73.79_{2.21}$	$69.85_{0.95}$	$69.76_{8.55}$	$69.40_{11.28}$	$63.73_{6.43}$	$\underline{78.41_{1.08}}$	$\mathbf{82.26}_{3.56}$
	32	$\underline{79.63_{3.57}}$	$76.25_{1.26}$	$74.29_{5.32}$	$68.45_{1.06}$	$77.71_{3.56}$	$71.64_{1.65}$	$66.93_{5.67}$	$73.61_{9.28}$	$66.51_{3.92}$	$78.37_{4.50}$	$\mathbf{83.23}_{9.26}$
	64	$\underline{82.27_{1.61}}$	$81.92_{1.00}$	$79.55_{3.19}$	$68.28_{3.97}$	$82.14_{2.28}$	$72.26_{1.62}$	OOW	OOW	$70.83_{3.43}$	$81.18_{6.17}$	$\mathbf{85.17}_{7.71}$
	4	$56.79_{26.02}$	$50.00_{0.00}$	$48.50_{12.82}$	$56.22_{21.00}$	$58.72_{19.16}$	$48.57_{6.04}$	$56.30_{12.43}$	$56.45_{15.45}$	$55.87_{13.49}$	$\underline{68.34_{7.48}}$	$\mathbf{70.75}_{3.64}$
	8	$68.51_{5.16}$	$59.97_{1.36}$	$63.43_{11.03}$	$65.77_{5.00}$	$66.30_{10.01}$	$60.00_{4.84}$	$58.99_{10.12}$	$56.37_{11.56}$	$66.01_{9.25}$	$\mathbf{70.37}_{3.23}$	$\underline{69.97_{3.25}}$
Boold ↑	16	$68.30_{6.16}$	$63.28_{7.62}$	$65.98_{6.49}$	$66.27_{5.04}$	$64.14_{6.80}$	$54.76_{4.53}$	$56.59_{5.21}$	$60.62_{4.13}$	$65.14_{7.55}$	$\underline{70.07_{5.19}}$	$\mathbf{72.46}_{1.87}$
	32	$67.39_{4.46}$	$66.41_{6.37}$	$63.46_{4.43}$	$69.71_{6.24}$	$68.65_{4.37}$	$59.87_{3.72}$	$58.69_{1.53}$	$57.94_{4.16}$	$69.95_{3.39}$	$\underline{71.13_{4.38}}$	$\mathbf{74.81}_{2.67}$
	64	$71.76_{2.56}$	$69.46_{2.96}$	$68.83_{5.61}$	$72.75_{4.36}$	$\underline{73.88_{1.97}}$	$61.75_{2.19}$	$65.79_{2.05}$	$63.47_{7.36}$	$70.88_{1.58}$	$71.04_{4.36}$	$\mathbf{74.73}_{2.08}$
	4	$57.09_{18.84}$	$50.00_{0.00}$	$52.50_{7.77}$	$62.35_{7.48}$	$56.28_{13.01}$	$64.22_{6.78}$	$71.71_{5.31}$	$63.96_{3.32}$	$70.42_{3.69}$	$\mathbf{80.28}_{0.75}$	$\underline{72.06_{5.04}}$
	8	$65.52_{13.18}$	$50.86_{22.03}$	$65.34_{8.84}$	$64.69_{13.33}$	$69.08_{9.68}$	$67.39_{12.92}$	$72.21_{2.07}$	$65.47_{3.95}$	$64.30_{5.88}$	$\mathbf{79.38}_{1.66}$	$\underline{73.70_{4.94}}$
Diabetes ↑	16	$73.44_{0.52}$	$65.69_{6.54}$	$65.69_{6.33}$	$71.86_{3.16}$	$73.69_{3.21}$	$73.79_{6.48}$	$71.64_{5.05}$	$66.71_{0.76}$	$67.34_{2.79}$	$\mathbf{80.15}_{1.35}$	$\underline{76.78_{4.69}}$
	32	$73.95_{3.32}$	$72.97_{3.77}$	$71.27_{8.04}$	$72.91_{3.09}$	$75.22_{3.21}$	$76.70_{4.55}$	$73.32_{1.59}$	$66.97_{1.75}$	$69.74_{4.41}$	$\mathbf{80.06}_{1.18}$	$\underline{77.01_{2.44}}$
	64	$74.52_{1.59}$	$72.56_{3.17}$	$76.92_{2.39}$	$74.44_{4.13}$	$77.82_{3.49}$	$\underline{78.64_{3.32}}$	$70.22_{4.09}$	$69.27_{6.15}$	$71.56_{4.55}$	$\mathbf{80.91}_{1.62}$	$78.32_{2.49}$
	4	$52.68_{4.46}$	$50.00_{0.00}$	$\underline{57.00_{10.75}}$	$48.92_{4.60}$	$54.00_{7.34}$	$48.80_{6.76}$	$52.99_{4.08}$	$54.33_{6.54}$	$51.90_{9.40}$	$55.94_{1.10}$	$\mathbf{58.21}_{0.93}$
	8	$55.52_{8.88}$	$52.22_{4.90}$	$\underline{59.84_{7.33}}$	$55.26_{3.92}$	$52.58_{11.27}$	$54.50_{8.25}$	$52.43_{4.36}$	$52.90_{5.79}$	$56.42_{12.89}$	$57.42_{3.10}$	$\mathbf{62.90}_{4.52}$
Credit-g ↑	16	$58.26_{5.17}$	$56.23_{4.37}$	$58.42_{8.36}$	$59.22_{11.38}$	$58.91_{8.04}$	$57.63_{7.58}$	$55.29_{4.80}$	$51.65_{4.02}$	$\underline{60.38_{14.03}}$	$56.60_{2.22}$	$\mathbf{67.11}_{1.22}$
	32	$67.85_{5.78}$	$65.33_{6.28}$	$57.48_{5.73}$	$\underline{72.60_{7.18}}$	$66.27_{5.06}$	$63.24_{5.47}$	OOW	OOW	$68.64_{3.86}$	$61.79_{10.25}$	$\mathbf{73.13}_{1.49}$
	64	$72.77_{9.29}$	$70.79_{2.34}$	$68.53_{4.99}$	$\underline{73.12_{6.23}}$	$68.95_{6.14}$	$70.97_{4.95}$	OOW	OOW	$70.80_{4.09}$	$66.43_{2.90}$	$\mathbf{75.65}_{1.05}$

Table 2: Evaluation results, including the AUC scores across six classification datasets. The best performances are highlighted in bold, and second-best are underlined. For consistency and clarity, all subsequent tables in this paper adhere to the same format: metric values are averaged over three random seeds, with standard deviation provided after it. Out of In-context Window (OOW) indicates that the input length exceeds the maximum context window limit of the model and cannot be processed.

SaTE	LLM	Meta	Size	4	8	16	32	64
✓				$85.16_{4.44}$	$84.86_{6.73}$	$88.59_{8.54}$	$90.03_{3.60}$	$91.07_{2.75}$
✓		✓		$81.27_{10.13}$	$86.26_{7.29}$	$90.43_{5.07}$	$90.52_{2.59}$	$91.99_{2.46}$
✓	✓		13B	$84.64_{4.16}$	$84.71_{7.93}$	$89.63_{6.21}$	$90.63_{4.90}$	$91.52_{4.53}$
	✓	✓	13B	$76.57_{6.56}$	$83.13_{1.79}$	$86.52_{2.34}$	$83.09_{5.95}$	$89.68_{5.81}$
✓	✓	✓	7B	$85.48_{5.85}$	$87.39_{6.01}$	$92.08_{3.24}$	$92.93_{2.71}$	$93.13_{2.60}$
✓	✓	✓	13B	$\mathbf{86.10}_{5.42}$	$\mathbf{89.59}_{4.09}$	$\mathbf{92.10}_{2.79}$	$\mathbf{93.00}_{2.68}$	$\mathbf{93.27}_{2.33}$

Table 3: Ablation experiments conducted on Heart dataset

STUNT [Nam *et al.*, 2023] is used as an alternative, (2) **LLM** indicates whether task-relevant knowledge is extracted from the LLMs, (3) **Meta** refers to the process of applying meta-learning techniques to unlabeled data, and (4) **Size** represents the effect of varying LLM scales on performance.

Table 3 presents the impact of various ablation experiments on the AUC across the Heart datasets. In all cases, modifying any of the ablated components leads to a decline in performance. Specifically, we replaced **SaTE** with the encoder from STUNT [Nam *et al.*, 2023] and the result reveal a maximum performance drop, particularly with few-shot labeled samples. For instance, with just 4 labeled samples, performance decreased by nearly 10%. This highlights the critical role of capturing the unique properties of tabular data and the importance of incorporating semantic information from column names, especially when working with limited labeled data. Besides, without extracting task-relevant knowledge from the LLMs using metadata, or meta-learning with unlabeled data, also decrease the model's performance. This highlights the importance of learning task-related semantic knowledge in the meta-learning process. Interestingly, increasing the size of the selected LLMs does not result in significant performance gains, possibly due to the 7B model has already contained all the necessary task-related semantic knowledge.

4.4 Impact of Labeled Sample Size and LLM Activation Layers on `Latte`

The deeper of LLM, the more sufficient knowledge it contains. Fig. 2 (a-b) illustrates the effect of activation in different layers L of the LLM and the number of labeled samples on model performance. Fig. 2 (a) presents AUC values for classification tasks on the Blood dataset, while Fig. 2 (b) shows MSE values for regression tasks on the Abalone dataset. As the number of labeled samples increases, model performance improves across both tasks. In classification tasks, performance improves as deeper layers of the LLM are activated. This suggests that higher-order knowledge is accumulated in the later layers. In contrast, for regression tasks, optimal performance is achieved at around 30 layers. In summary, adding more layers may not improve the model's performance and could instead introduce noise, leading to a decline in the quality of learned representations.

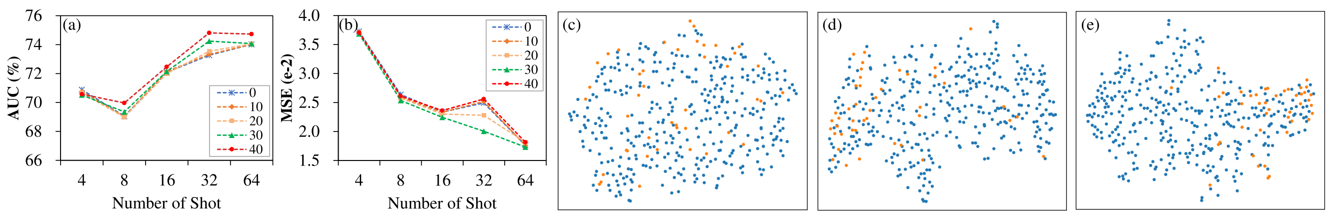
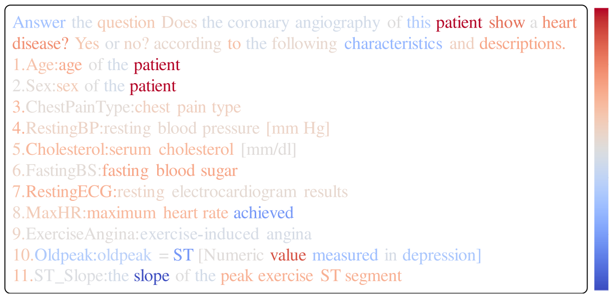


Figure 2: **Latte**'s parameter experiments and representation visualizations. (a-b) Impact of the number of labeled samples and LLM activation layers on **Latte**'s performance in classification and regression tasks. (c-e) Visualization of **Latte**'s learned representations: (c) original representations, and **Latte** representations trained on (d) 4 and (e) 64 labeled samples.



Answer the question Does the coronary angiography of this patient show a heart disease? Yes or no? according to the following characteristics and descriptions.
1.Age:age of the patient
2.Sex:sex of the patient
3.ChestPainType:chest pain type
4.RestingBP:resting blood pressure [mm Hg]
5.Cholesterol:serum cholesterol [mm/dl]
6.FastingBS:fasting blood sugar
7.RestingECG:resting electrocardiogram results
8.MaxHR:maximum heart rate achieved
9.ExerciseAngina:exercise-induced angina
10.Oldpeak:oldpeak = ST [Numeric value measured in depression]
11.ST_Slope:the slope of the peak exercise ST segment

Figure 3: The heatmap visualizes the task-related semantic information learned by the model during the meta-learning stage. Higher values (indicated by redder colors) represent a greater semantic similarity between the model's learned representations and the semantic knowledge associated with the LLM task.

4.5 What Latent Knowledge Does Latte Transfer From the LLM

Latte can retrieve task-related semantic information from metadata. To examine the task-related semantic information embedded in the representations learned by Latte during the unlabeled data meta-learning stage, we measured the semantic similarity between the learned representations and the LLM activation at the word level using metadata. This was visualized through a heatmap, as shown in Fig. 3. Higher values, indicated by redder colors, reflect greater semantic similarity. As an example of a downstream task, this involves predicting whether a patient has heart disease. The results of the heatmap demonstrate that our model, after meta-learning on unlabeled data, captures task-relevant semantic information. For example, the representations learned by **Latte** exhibit high semantic similarity with "patient" and "heart disease," indicating its effective learning of task-relevant semantic knowledge. Furthermore, the model identifies and emphasizes key features such as "age," "cholesterol," "fast blood sugar," "maximum heart rate," and "peak exercise ST segment," which are crucial for determining heart disease.

4.6 Comparison of LLM Invocation Costs

Latte has an efficiency advantage in calling LLM. Table 4 compares the number of LLM calls required by **Latte**and other LLM-based baselines across the data preprocessing, training, and inference stages. Specifically, TABLET, In-context and TabLLM require calling the LLM once for each test sample during inference, resulting in significant overhead. While FeatLLM limits LLM calls to the training phase to assist in learning, the number of calls still scales with the size of the training dataset. Our method calls the LLM only once during preprocessing to gather task-relevant knowledge, which is then stored locally. As a result, our approach significantly reduces the overall number of LLM calls throughout the entire process.

Stage	Methods				
	In-context	TABLET	TabLLM	FeatLLM	**Latte**
Preprocessing	0	0	0	0	1
Training	0	1	1	30	0
Inference	9043	9043	9043	0	0
Totle	9043	9044	9044	30	1

Table 4: Compare our method with the LLM baseline in terms of the number of LLM calls made during the preprocessing, training, and inference stages.

4.7 Visualization Experiments

Latte can generate high-quality representations with limited labeled data. The visualization of representations on the Bank dataset is shown in Fig. 2 (c-e), where the original representations refer to those generated without training. **Latte** learns effective representations in few-shot settings. With just four labeled samples, it produces a separable representation, and this separability improves as the number of labeled samples increases to 64. This demonstrates the model's strong ability to learn useful features with minimal supervision.

5 Conclusion

This paper introduces a novel framework, **Latte**, aimed at addressing the challenges of few-shot tabular learning by extracting latent-level knowledge from large language models (LLMs) during the training process. **Latte** positions the LLM as a "teacher" that provides task-relevant guidance to the downstream tabular model and enable a knowledge-guided training process that mitigates overfitting, a common issue in few-shot settings, and promotes convergence toward a generalized model. Additionally, **Latte** incorporates an unsupervised pre-training stage that leverages unlabeled data for robust parameter initialization, further enhancing the framework's performance. Extensive experiments demonstrate the effectiveness of **Latte** on a variety of few-shot tabular learning tasks.

Acknowledgments

This work was supported by a grant from the National Natural Science Foundation of China under grants (No.62372211, 62272191), and the Science and Technology Development Program of Jilin Province (No.20250102216JC).

Contribution Statement

Ruxue Shi and Hengrui Gu have equal contributions, with Xin Wang as the corresponding author.

References

[Arik and Pfister, 2021] Sercan Ö Arik and Tomas Pfister. Tabnet: Attentive interpretable tabular learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, pages 6679–6687, 2021.

[Badirli *et al.*, 2020] Sarkhan Badirli, Xuanqing Liu, Zhengming Xing, Avradeep Bhowmik, Khoa Doan, and Sathiya S Keerthi. Gradient boosting neural networks: Grownet. *arXiv preprint arXiv:2002.07971*, 2020.

[Bahri *et al.*, 2021] Dara Bahri, Heinrich Jiang, Yi Tay, and Donald Metzler. Scarf: Self-supervised contrastive learning using random feature corruption. *arXiv preprint arXiv:2106.15147*, 2021.

[Breiman, 2001] Leo Breiman. Random forests. *Machine learning*, 45:5–32, 2001.

[Cao, 2022] Longbing Cao. Ai in finance: challenges, techniques, and opportunities. *ACM Computing Surveys (CSUR)*, 55(3):1–38, 2022.

[Chen and Guestrin, 2016] Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pages 785–794, 2016.

[Chen *et al.*, 2019] Wei-Yu Chen, Yen-Cheng Liu, Zsolt Kira, Yu-Chiang Frank Wang, and Jia-Bin Huang. A closer look at few-shot classification. *arXiv preprint arXiv:1904.04232*, 2019.

[Chen *et al.*, 2023] Suiyao Chen, Jing Wu, Naira Hovakimyan, and Handong Yao. Recontab: Regularized contrastive representation learning for tabular data. *arXiv preprint arXiv:2310.18541*, 2023.

[Chen *et al.*, 2024] Chao Chen, Kai Liu, Ze Chen, Yi Gu, Yue Wu, Mingyuan Tao, Zhihang Fu, and Jieping Ye. Inside: Llms' internal states retain the power of hallucination detection. *arXiv preprint arXiv:2402.03744*, 2024.

[Duan *et al.*, 2024] Hanyu Duan, Yi Yang, and Kar Yan Tam. Do llms know about hallucination? an empirical investigation of llm's hidden states. *arXiv preprint arXiv:2402.09733*, 2024.

[Gorishniy *et al.*, 2021] Yury Gorishniy, Ivan Rubachev, Valentin Khrulkov, and Artem Babenko. Revisiting deep learning models for tabular data. *Advances in Neural Information Processing Systems*, 34:18932–18943, 2021.

[Han *et al.*, 2024a] Sungwon Han, Jinsung Yoon, Sercan O Arik, and Tomas Pfister. Large language models can automatically engineer features for few-shot tabular learning. *arXiv preprint arXiv:2404.09491*, 2024.

[Han *et al.*, 2024b] Sungwon Han, Jinsung Yoon, Sercan O Arik, and Tomas Pfister. Large language models can automatically engineer features for few-shot tabular learning. *arXiv preprint arXiv:2404.09491*, 2024.

[Hegselmann *et al.*, 2023] Stefan Hegselmann, Alejandro Buendia, Hunter Lang, Monica Agrawal, Xiaoyi Jiang, and David Sontag. Tabllm: Few-shot classification of tabular data with large language models. In *International Conference on Artificial Intelligence and Statistics*, pages 5549–5581. PMLR, 2023.

[Hollmann *et al.*, 2023] Noah Hollmann, Samuel Müller, Katharina Eggensperger, and Frank Hutter. Tabpfn: A transformer that solves small tabular classification problems in a second. In *The Eleventh International Conference on Learning Representations*, 2023.

[Huang *et al.*, 2020] Xin Huang, Ashish Khetan, Milan Cvitkovic, and Zohar Karnin. Tabtransformer: Tabular data modeling using contextual embeddings. *arXiv preprint arXiv:2012.06678*, 2020.

[Ke *et al.*, 2017] Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. Lightgbm: A highly efficient gradient boosting decision tree. *Advances in neural information processing systems*, 30, 2017.

[Ke *et al.*, 2018] Guolin Ke, Jia Zhang, Zhenhui Xu, Jiang Bian, and Tie-Yan Liu. Tabnn: A universal neural network solution for tabular data. 2018.

[Ke *et al.*, 2019] Guolin Ke, Zhenhui Xu, Jia Zhang, Jiang Bian, and Tie-Yan Liu. Deepgbm: A deep learning framework distilled by gbdt for online prediction tasks. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 384–394, 2019.

[Klambauer *et al.*, 2017] Günter Klambauer, Thomas Unterthiner, Andreas Mayr, and Sepp Hochreiter. Self-normalizing neural networks. *Advances in neural information processing systems*, 30, 2017.

[LaValley, 2008] Michael P LaValley. Logistic regression. *Circulation*, 117(18):2395–2399, 2008.

[Li *et al.*, 2024] Yingji Li, Mengnan Du, Rui Song, Xin Wang, Mingchen Sun, and Ying Wang. Mitigating social biases of pre-trained language models via contrastive self-debiasing with double data augmentation. *Artificial Intelligence*, 332:104143, 2024.

[Liu *et al.*, 2024] Ruoxue Liu, Linjiajie Fang, Wenjia Wang, and Bingyi Jing. D2r2: Diffusion-based representation with random distance matching for tabular few-shot learning. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024.

[Loh, 2011] Wei-Yin Loh. Classification and regression trees. *Wiley interdisciplinary reviews: data mining and knowledge discovery*, 1(1):14–23, 2011.

[Nam *et al.*, 2023] Jaehyun Nam, Jihoon Tack, Kyungmin Lee, Hankook Lee, and Jinwoo Shin. Stunt: Few-shot tabular learning with self-generated tasks from unlabeled tables. In *The Eleventh International Conference on Learning Representations*, 2023.

[Oreshkin *et al.*, 2018] Boris Oreshkin, Pau Rodríguez López, and Alexandre Lacoste. Tadam: Task dependent adaptive metric for improved few-shot learning. *Advances in neural information processing systems*, 31, 2018.

[Perković *et al.*, 2024] Gabrijela Perković, Antun Drobnjak, and Ivica Botički. Hallucinations in llms: Understanding and addressing challenges. In *2024 47th MIPRO ICT and Electronics Convention (MIPRO)*, pages 2084–2088. IEEE, 2024.

[Popov *et al.*, 2019] Sergei Popov, Stanislav Morozov, and Artem Babenko. Neural oblivious decision ensembles for deep learning on tabular data. *arXiv preprint arXiv:1909.06312*, 2019.

[Prokhorenkova *et al.*, 2018] Liudmila Prokhorenkova, Gleb Gusev, Aleksandr Vorobev, Anna Veronika Dorogush, and Andrey Gulin. Catboost: unbiased boosting with categorical features. *Advances in neural information processing systems*, 31, 2018.

[Shailaja *et al.*, 2018] K Shailaja, Banoth Seetharamulu, and MA Jabbar. Machine learning in healthcare: A review. In *2018 Second international conference on electronics, communication and aerospace technology (ICECA)*, pages 910–914. IEEE, 2018.

[Shi *et al.*, 2025] Ruxue Shi, Yili Wang, Mengnan Du, Xu Shen, and Xin Wang. A comprehensive survey of synthetic tabular data generation. *arXiv preprint arXiv:2504.16506*, 2025.

[Slack and Singh, 2023] Dylan Slack and Sameer Singh. Tablet: Learning from instructions for tabular data. *arXiv preprint arXiv:2304.13188*, 2023.

[Snell *et al.*, 2017] Jake Snell, Kevin Swersky, and Richard Zemel. Prototypical networks for few-shot learning. *Advances in neural information processing systems*, 30, 2017.

[Somepalli *et al.*, 2021] Gowthami Somepalli, Micah Goldblum, Avi Schwarzschild, C Bayan Bruss, and Tom Goldstein. Saint: Improved neural networks for tabular data via row attention and contrastive pre-training. *arXiv preprint arXiv:2106.01342*, 2021.

[Song *et al.*, 2019] Weiping Song, Chence Shi, Zhiping Xiao, Zhijian Duan, Yewen Xu, Ming Zhang, and Jian Tang. Autoint: Automatic feature interaction learning via self-attentive neural networks. In *Proceedings of the 28th ACM international conference on information and knowledge management*, pages 1161–1170, 2019.

[Touvron *et al.*, 2023] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.

[Wang *et al.*, 2020] Yaqing Wang, Quanming Yao, James T Kwok, and Lionel M Ni. Generalizing from a few examples: A survey on few-shot learning. *ACM computing surveys (csur)*, 53(3):1–34, 2020.

[Wang *et al.*, 2021a] Bo Wang, Tao Shen, Guodong Long, Tianyi Zhou, Ying Wang, and Yi Chang. Structure-augmented text representation learning for efficient knowledge graph completion. In *Proceedings of the Web Conference 2021*, pages 1737–1748, 2021.

[Wang *et al.*, 2021b] Ruoxi Wang, Rakesh Shivanna, Derek Cheng, Sagar Jain, Dong Lin, Lichan Hong, and Ed Chi. Dcn v2: Improved deep & cross network and practical lessons for web-scale learning to rank systems. In *Proceedings of the web conference 2021*, pages 1785–1797, 2021.

[Wei *et al.*, 2022] Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, et al. Emergent abilities of large language models. *arXiv preprint arXiv:2206.07682*, 2022.

[Wen *et al.*, 2024] Xumeng Wen, Han Zhang, Shun Zheng, Wei Xu, and Jiang Bian. From supervised to generative: A novel paradigm for tabular deep learning with large language models. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 3323–3333, 2024.

[Yoon *et al.*, 2020] Jinsung Yoon, Yao Zhang, James Jordon, and Mihaela van der Schaar. Vime: Extending the success of self-and semi-supervised learning to tabular domain. *Advances in Neural Information Processing Systems*, 33:11033–11043, 2020.

[Zhou *et al.*, 2024] Zhenhong Zhou, Haiyang Yu, Xinghua Zhang, Rongwu Xu, Fei Huang, and Yongbin Li. How alignment and jailbreak work: Explain llm safety through intermediate hidden states. *arXiv preprint arXiv:2406.05644*, 2024.