Large Language Models Can Automatically Engineer Features for Few-Shot Tabular Learning

Sungwon Han [1]   **Jinsung Yoon** [2]   **Sercan Ö. Arik** [2]   **Tomas Pfister** [2]

Abstract

Large Language Models (LLMs), with their remarkable ability to tackle challenging and unseen reasoning problems, hold immense potential for tabular learning, that is vital for many real-world applications. In this paper, we propose a novel in-context learning framework, FeatLLM, which employs LLMs as feature engineers to produce an input data set that is optimally suited for tabular predictions. The generated features are used to infer class likelihood with a simple downstream machine learning model, such as linear regression and yields high performance few-shot learning. The proposed FeatLLM framework only uses this simple predictive model with the discovered features at inference time. Compared to existing LLM-based approaches, FeatLLM eliminates the need to send queries to the LLM for each sample at inference time. Moreover, it merely requires API-level access to LLMs, and overcomes prompt size limitations. As demonstrated across numerous tabular datasets from a wide range of domains, FeatLLM generates high-quality rules, significantly (10% on average) outperforming alternatives such as TabLLM and STUNT.

1. Introduction

In recent years, Large Language Models (LLMs) have shown impressive abilities for generating suitable responses for previously unseen tasks without requiring task-specific fine-tuning (Creswell et al., 2022; Imani et al., 2023; Taylor et al., 2022). Trained on extensive text corpora, the rich prior knowledge of LLMs can substitute the need for expert domain knowledge in many areas (Gen, 2021; Singhal et al.,

2023; Wilcox & Hripcsak, 2003), playing a crucial role in the automation of diverse applications, including dialogue analysis (Finch et al., 2023) and program repair (Fan et al., 2023). Particularly, by incorporating a task description and few examples into prompts, LLMs can understand the context of a task and "attend" to important features and examples (Brown et al., 2020; Zhao et al., 2023). This demonstrates the capability of LLMs to analyze data, highlighting their potential to perform as an expert feature engineer.

Utilization of LLMs' prior knowledge and reasoning capabilities have been extended to the domain of tabular learning. For instance, knowledge such as related to the increased risk of certain diseases in older individuals can be beneficial at disease prediction tasks. In this context, recent works define task and feature descriptions in natural language, serialize the data, and then feed it into an LLM (Dinh et al., 2022; Wang et al., 2023). This is often followed by either using in-context learning (Nam et al., 2023a) or parameter-efficient fine-tuning (Dinh et al., 2022; Hegselmann et al., 2023). Prior knowledge provided by LLMs has proven effective, outperforming conventional tabular learning baselines in low-shot regimes (Hegselmann et al., 2023).

Current LLM-based tabular learning methods have some limitations. For end-to-end predictions, at least one LLM inference per sample is required, making it computationally expensive. Most methods require fine-tuning the LLM for high accuracy, making them infeasible to the application to the LLMs that do not come with accessible training codebases or tuning APIs, a prominent concern given that recently-proposed top-performance LLMs only permit limited access via inference APIs (Anil et al., 2023; OpenAI, 2023). In addition, most approaches are not suitable with lengthy prompts (Liu et al., 2023) – when the number of features in tabular data grows to the point where the serialized text length becomes long, the approach often turns infeasible or experiences a decline in performance. These constraints pose fundamental challenges to the real-world application of LLM-based tabular learning models.

Our proposed approach goes beyond treating LLMs to yield end-to-end predictions. Instead, we assign them the role of feature engineering, with them proposing a more efficient way to utilize their prior knowledge. Specifically, we

---

[1]Work done at Google as a research intern. School of Computing, Korea Advanced Institute of Science and Technology, Daejeon, Republic of Korea [2]Google Cloud AI, Sunnyvale, California, USA. Correspondence to: Jinsung Yoon <jinsungyoon@google.com>.

introduce a novel in-context learning framework, FeatLLM, which aims to understand the "criteria" underlying LLM predictions. For instance, for the task of predicting a particular disease, the LLM can directly infer and generate rules that determine which feature conditions result in identifying the disease. The criteria (or rules) are used to create new features that can replace existing ones in solving tasks. This approach shifts the capability of the LLM to feature generation, yielding a simpler downstream model – once the features are discovered, a complex machine learning model is no longer needed for inference, improving the inference latency. By leveraging the in-context learning ability of LLMs, it's possible to extract features without the need for model training, by adding example demonstrations to the prompts. Furthermore, by employing techniques from ensemble methods (Breiman, 2001), like feature bagging, we can mitigate the challenge of excessive prompt size.

FeatLLM structures the input prompt for engineering new features into the following two sub-tasks: (1) understanding the problem and inferring the relationship between features and targets; and (2) based on the knowledge from the previous step and few-shot examples, deriving decisive rules to distinguish classes. This structured reasoning approach leads the LLMs to filter out uninformative features for rule extraction. The deduced rules are then applied to the data (parsed them as program code), and the data are transformed into a binary feature indicating whether or not the samples adheres to each rule. Finally, a model with low complexity is fitted to estimate class probabilities from these new features. The robustness is further improved by employing an ensemble approach, repeatedly generating features in conjunction with feature bagging. By appropriately adjusting the number of features and samples chosen in bagging, the challenge of excessive prompt size is mitigated. FeatLLM is free from training constraints and yields low inference cost, rendering the application of the LLM feasible for various real-world problems.

We evaluate FeatLLM on 13 different tabular datasets in low-shot regimes, showing its strong and robust performance. We show that LLMs proficiently utilize both prior knowledge and the knowledge from the data, extracting high-quality features. Our framework outperforms contemporary few-shot learning baselines across various settings. The code is released via anonymous GitHub link at `https://github.com/Sungwon-Han/FeatLLM`.

2. Related Work

2.1. Few-Shot Learning with Tabular Data

The advancement of few-shot learning has enabled the extraction of generalizable representations from data, even with minimal labeling costs (Chen et al., 2018). Initially focusing on image data, few-shot learning has proven its strong generalizability across various data modalities including text, audio, and more recently, tabular data (Majumder et al., 2022; Nam et al., 2023b; Schick & Schütze, 2021). Learning from a small number of labeled samples presents the risk of learning spurious correlations (Bartlett et al., 2021). Consequently, recent efforts have employed additional sources of information or introduced domain-specific prior knowledge to impart appropriate inductive biases for model training. For example, utilizing large quantities of more readily-available unlabeled datasets with judicious augmentation for semi-supervised learning (Ucar et al., 2021; Yoon et al., 2020), or seeking good model initialization through task-agnostic unsupervised pre-training (Somepalli et al., 2021). Such unsupervised objectives can involve masking or corrupting part of the data and then using a reconstruction objective (Arik & Pfister, 2021; Majmundar et al., 2022) or employing data augmentation to apply contrastive learning objectives (Bahri et al., 2022; Chen et al., 2020). However, the heterogeneous nature of tabular data makes it challenging to find universally applicable augmentations, and these strategies have not been notably effective in few-shot settings (Nam et al., 2023b).

Recent work have proposed training models on a broad range of tabular datasets beyond those related to the target task and then applying transfer learning to the target task (Zhang et al., 2023; Levin et al., 2022; Zhu et al., 2023). For instance, TransTab (Wang & Sun, 2022) trains transformer-based models learning of semantic relationships between columns through column names, not just table values. The knowledge extracted from various tables and columns has facilitated zero-shot or few-shot inference for new tabular tasks. Meanwhile, TabPFN (Hollmann et al., 2023) mixes different types of distributions to create synthetic data for pre-training a Bayesian neural network. After pre-training, inference is performed by inputting both training and test samples of the target task without further training. The prior knowledge gained from various datasets helped surpass the performance of traditional tree-based approaches and is proved to be beneficial in few-shot settings.

2.2. Language-Interfaced Tabular Learning

Another promising approach to leverage prior knowledge is the incorporation of large language models (LLMs) (Anil et al., 2023; OpenAI, 2023). LLMs, trained on extensive and diverse text corpora, have demonstrated the ability to generalize to unseen tasks, proving their utility across various domains (Brown et al., 2020). Recent studies have attempted to harness the rich prior knowledge encapsulated by LLMs for tabular learning. They typically propose serializing tabular data into text and feeding them into the input prompts of LLMs (Dinh et al., 2022; Hegselmann et al., 2023; Wang et al., 2023). By including tabular data, fea-
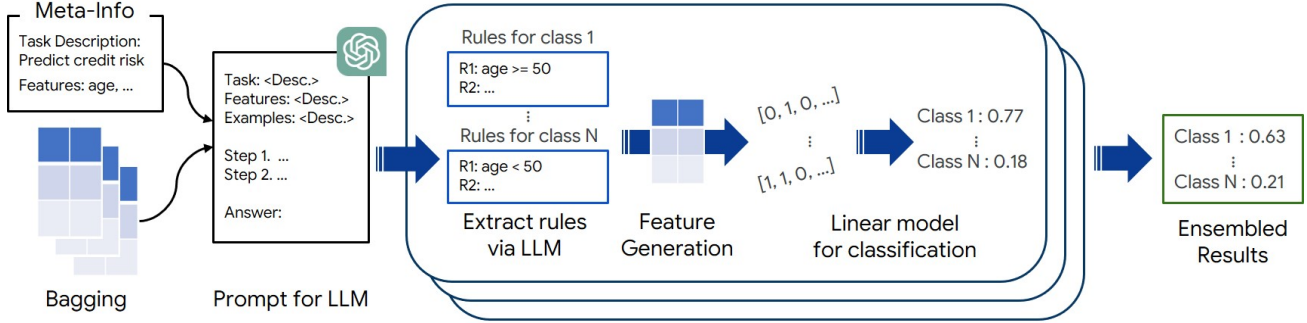
Figure 1: Illustration of FeatLLM. FeatLLM extracts rules for each class, utilizing prior knowledge and few-shot examples. These rules are then parsed and applied to create binary features for data samples. A linear layer is trained on these binary features to estimate class likelihoods. This procedure is repeated multiple times for ensembling.

ture descriptions, and task descriptions in the input prompts, these can infer the relationships between tasks and features to perform inference. Research has evolved to either specifically train models using parameter-efficient tuning techniques like LoRA (Hu et al., 2021) or IA3 (Liu et al., 2022), or employ in-context learning by adding few-shot example demonstrations to the prompts without training (Dinh et al., 2022; Hegselmann et al., 2023; Nam et al., 2023a; Slack & Singh, 2023).

Despite the effectiveness of the aforementioned methods in improving few-shot tabular learning, the requirement to always pass inference through the LLM poses challenges for application in industrial domains. This study aims to depart from the conventional black-box approach of processing each sample's prediction in an end-to-end fashion via an LLM. Instead, it targets the LLM to directly infer the conditions or rules associated with each class. FeatLLM parses the generated rules into the program code to create new features, enabling predictions without the need to pass through the LLM and leveraging in-context learning to extract rules utilizing prior knowledge and information from few-shot samples.

3. Methods

FeatLLM is designed to extract "rules", the conditions associated with each answer class, from the few samples it inputs, as depicted in Figure 1. The extracted rules are parsed through an LLM, and then used to create new binary features of the given sample indicating whether or not each rule is satisfied. The set of new binary features are input to a dot product operation with non-negative trainable weights, to estimate the likelihood of each class. By training this model, we learn the importance of each feature and how it affects the class likelihood (Section 3.2). This process is repeated multiple times with bagging and then combined via ensembling. Problem formulation and details of each stage are explained below.

Problem formulation. Let's consider a tabular dataset with N labeled samples $\mathcal{D} = \{(\mathbf{x}^i, \mathbf{y}^i)\}_{i=1}^N$. The dataset $\mathcal{D}$ consists of d input features (i.e., each $\mathbf{x}^i$ is a d-dimensional vector), and comes with natural language feature names, denoted by $F = \{f_j\}_{j=1}^d$, along with their definitions provided separately. Assuming the classification setup, the label $\mathbf{y}^i$ is defined as one of the predefined set of classes. For the k-shot learning experiments, only k ($< N$) labeled samples are randomly sampled to train the model.

3.1. Prompt Design for Extracting Rules

To enable the LLM to extract rules based on a more accurate reasoning path, we guide the problem-solving process to mimic how an expert human might approach a tabular learning task. Our prompt comprises three main components as follows (shown in Fig. 2 and Appendix A.2):

Basic information description. This part provides essential information for solving the problem (see orange text in Fig. 2). It includes a description of the task to be solved with label information, descriptions of features, and example demonstrations. The task description is formulated as a question (e.g., "Does this patient's myocardial infarction complications data indicate chronic heart failure? Yes or no?". More examples are provided in Appendix A.1). The feature description indicates its value type and optionally includes its definition. For categorical variables, examples of the possible categories can be provided. As example demonstrations, few training samples are serialized into text, along with their ground-truth labels. Serialization follows the format used in previous studies (Dinh et al., 2022; Hegselmann et al., 2023):

$$\text{Serialize}(\mathbf{x}^i, \mathbf{y}^i, F) =$$
$$\text{`` } f_1 \text{ is } \mathbf{x}_1^i. \ ... \ f_d \text{ is } \mathbf{x}_d^i. \ \text{ Answer: } \mathbf{y}^i \text{ ''} \qquad (1)$$

Reasoning instruction. We aim to enhance the LLM's reasoning by providing guidance through reasoning instruc-

You are an expert. Given the task description and the list of features and data examples, you are extracting conditions for each answer class to solve the task.

Task: <Task description>
Features: <Feature descriptions>
Examples: <Serialized training examples>

Let's first understand the problem and solve the problem step by step.

Step 1. Analyze the causal relationship or tendency between each feature and task description based on general knowledge and common sense within a short sentence.

Step 2. Based on the above examples and Step 1's results, infer 10 different conditions per answer, following the format below. The condition should make sense, well match examples, and must match the format for [condition] according to value type.

Format for Response:
10 different conditions for class [Class name]:
- [Condition]
...

Format for [Condition]:
For the categorical variable only,
- [Feature] is in [List of categories]
For the numerical variable only,
- [Feature] ($>$ or $>=$ or $<$ or $<=$) [Value]
- [Feature] is within range of [Value_start, Value_end]

Answer:
Step 1.

Figure 2: Prompt for rule extraction. Text in orange provides basic information description; blue text outlines reasoning instruction; and yellow text details response instruction.

tions (see blue text in Fig. 2). The reasoning instruction begins with an introductory sentence similar to the chain-of-thought approach (Wei et al., 2022b). Then, we divide the LLM's inference process into two steps. In the first step, the LLM is encouraged to infer the causal relationship or tendency between features and the task description. This is done without referring to example demonstrations, relying instead on general knowledge or common sense, allowing the LLM to fully extract prior knowledge about the problem. In the second step, the LLM uses example demonstrations and the information from the first step to deduce rules for each class. This two-step reasoning process prevents the model from identifying spurious correlations in irrelevant columns and can assist in focusing on more significant features. To prevent extracting too few or too many rules, potentially leading to either underfitting or creating an excessively large prompt, we have set a fixed number of

Provide me a python code for function, given description below.

Function name: <Name>
Input: Dataframe df_input
Input Features: <Features>
Output: Dataframe df_output. Create a new dataframe df_output. Each column in df_output refers whether the selected column in df_input follows the condition (1) or not (0). Be sure that the function code well matches with its feature type (i.e., numerical, categorical).

Conditions: <Conditions>

Wrap only the function part with <start> and <end> tokens, and do not add any comments, descriptions, and package importing lines in the code.

Figure 3: Prompt for parsing rules. This prompt incorporates the rules generated in the previous stage, placed within the <Conditions> section.

rules (i.e., 10)[1] to be extracted for each class. Additionally, when formulating rules, we let the LLM refer to the type of each feature in the basic information description part to avoid type mismatches in rules.

Response instruction. To facilitate easier parsing and use of the inferred rules, we guide the LLM on how to structure its responses through specific instructions (see yellow text in Fig. 2). The prompt includes the guidelines on the required format for responses for each answer class (i.e., format for the response) and the structure each rule should follow (i.e., format for the rule). These instructions allow the LLM to select the appropriate format based on the feature type. Without additional guidance, the LLM can combine multiple rules using logical operators like AND or OR. The example outcome is provided in Appendix A.3.

3.2. Inferring Class Likelihood via Rules

Parsing rules for feature generation. We utilize the rules generated in the previous stage to create new binary features. These features are created for each class, indicating whether the sample satisfies the rules associated with that class (i.e., either '0' or '1'). They can be used in predicting how likely a sample belongs to each class. However, since the rules generated by the LLM are based on natural language, designing a parsing strategy is required for automatic feature generation. Even though we constrain formatting of responses and rules during rule generation for easier parsing, the LLM's output can include noisy syntactic changes (Kovač et al., 2023). For instance, the LLM might alter expressions, like substituting ">" or "<" with phrases

---

[1]Analysis on the number of rules can be found in Section 4.2.

like "greater than" or "less than." This variability makes parsing more challenging.

To address the challenges of parsing noisy text, instead of building complex program code, we leverage the LLM itself. The LLM is good at understanding the semantic meaning despite syntactic changes in text, and it has the ability to generate a simple code with instructions (given it's trained on code data corpus). To leverage these, we include the function name, input and output descriptions, and inferred rules in the prompt, then input them into the LLM. Figures 3 and A.4-A.5 in Appendix show the prompt examples and the corresponding outcomes. The generated code is executed using Python's `exec()` function to perform data conversion.

Inferring class likelihood. Given new binary features from rules for each class, a simple method to measure the class likelihood of the sample is to count how many rules of each class it satisfies (i.e., the sum of the binary feature per class). However, not all rules and features carry the same importance, necessitating learning their significance from training samples. We aim to learn this importance using a conventional machine learning (ML) model, applied to each class's binary feature. For example, we can consider a linear model without bias. Let us denote the generated binary feature from sample $\mathbf{x}^i$ for class k as $\mathbf{z}_k^i \in \{0, 1\}^R$, where R is the number of rules per each class and c is the number of classes. The probability for each class $\mathbf{p}^i$ is computed as follows:

$$\begin{aligned} \text{logit}_k^i &= \max(\mathbf{w}_k, 0) \cdot \mathbf{z}_k^i, \\ \mathbf{p}^i &= \text{Softmax}([\text{logit}_1^i, ..., \text{logit}_c^i]). \end{aligned} \qquad (2)$$

Here, $\mathbf{w}_k$ represent the trainable weight in the linear model and they signify the importance of each feature. By projecting these weights into a positive subspace, we can ensure that the features generated by the LLM do not reduce the likelihood of a class during learning. The logit vector is then converted into class probabilities through the softmax function. We optimize the cross-entropy loss to learn the weight $\mathbf{w}_k$ using a few-shot training samples.

Ensembling with bagging. Finally, we repeatedly execute the entire process to create multiple inference models to make the final prediction via ensemble. Given the collection of trained weights from T trials, denoted as $\{\mathbf{w}[t]\}_{t=1}^T$, the prediction is made by averaging the probability for each class $\mathbf{p}^i$ computed with each weight $\mathbf{w}[t]$ as in Eq. 2.

To introduce randomness in rule generation for ensemble, a sufficiently high temperature is set for LLM inference. We also randomize the order of few-shot demonstrations based on the observation that the order can impact the query outcome of LLM (Lu et al., 2022)[2]. To utilize the diversity in prediction for more robust predictions, we adopt bagging to select a subset of features or instances for each trial,

which gets particularly useful for larger number of training samples or features. The proposed ensemble approach offers two main advantages. First, even if the LLM generates incorrect rules in some trials, the correct rules from other trials can compensate, enhancing the overall robustness of the framework. This ties into the LLM's self-consistency (Wang et al., 2022), as rules commonly inferred across multiple trials are more likely to be accurate. Second, the ensemble method addresses the limitation of LLM's prompt size. When the tabular data exceed the allowable input prompt size, bagging helps reduce it, ensuring the model's robust performance. While a single rule set might not capture the relationships of all features, the ensemble of various weak learners created across multiple trials can nullify the impact of incomplete feature/instance coverage in individual trials.

4. Experiments

We evaluate FeatLLM over multiple tabular datasets, and conduct various analyses to investigate how our framework works. Due to space constraints, additional experiments, such as varying LLMs, qualitative analysis, and others, are described in Appendix.

4.1. Performance Evaluation

Datasets. Our experiment utilizes 13 datasets for binary or multi-class classification tasks: (1) Adult (Asuncion & Newman, 2007) for predicting whether an individual earns over $50,000 annually; (2) Bank (Moro et al., 2014) for predicting whether a customer will subscribe to a term deposit; (3) Blood (Yeh et al., 2009) for predicting whether donors will return for subsequent donations; (4) Car (Kadra et al., 2021) for predicting the quality of a car; (5) Communities (Redmond, 2009) for predicting the crime rate in a specified region; (6) Credit-g (Kadra et al., 2021) for predicting whether an individual poses a good or bad credit risk; (7) Diabetes[3] for predicting the presence of diabetes in an individual; (8) Heart[4] for predicting the occurrence of coronary artery disease in an individual; and (9) Myocardial (Golovenkin & Voino-Yasenetsky, 2020) for predicting whether an individual suffers from chronic heart failure.

We also include more recent datasets and synthetic datasets which have been less frequently discussed and were not included in the pre-training process of LLMs: (10) Cultivars for predicting how high the grain yield of this soybean cultivar will be[5]; (11) NHANES for predicting the given person's age group from the record[6]; (12) Sequence-type for

---

[2]Analysis on the rule diversity can be found in Appendix D

[3]`kaggle.com/datasets/uciml/pima-indians-diabetes-database`
[4]`kaggle.com/datasets/fedesoriano/heart-failure-prediction`
[5]`archive.ics.uci.edu/dataset/913`
[6]`archive.ics.uci.edu/dataset/887`

Table 1: Evaluation results, showing AUC across 11 datasets with attributes fewer than 100, are presented. Best performances are bolded, and our framework's performances, when second-best, are underlined. Results with more training shots (e.g., 32, 64) and an additional baseline (e.g., RandomForest) are provided in Appendix K.

Data	Shot	LogReg	XGBoost	SCARF	TabPFN	STUNT	In-context	TABLET	TabLLM	Ours
Adult	4	72.10±12.30	50.00±0.00	58.34±15.42	60.89±23.28	67.43±29.61	77.51±5.24	75.29±12.24	83.57±2.69	**86.68±0.86**
	8	76.02±3.37	59.19±6.92	72.42±8.95	70.42±9.96	82.16±6.93	79.30±2.89	77.56±7.56	83.52±4.30	**87.89±0.06**
	16	75.20±5.10	60.68±13.92	75.63±9.56	70.34±9.96	80.57±10.93	79.50±4.57	79.74±5.64	83.23±2.45	**87.54±0.50**
Bank	4	63.70±3.87	50.00±0.00	58.53±5.49	63.19±11.60	56.34±12.82	61.38±1.30	58.11±6.29	62.51±8.95	**70.45±3.69**
	8	72.52±3.21	58.78±10.54	55.28±11.88	62.81±7.84	63.01±8.78	69.57±13.35	69.08±6.00	63.19±5.79	**75.85±6.66**
	16	77.51±3.09	70.34±5.86	65.81±1.79	73.79±2.21	69.85±0.95	69.76±8.55	69.40±11.28	63.73±6.43	**78.41±1.08**
Blood	4	56.79±26.02	50.00±0.00	56.22±21.00	58.72±19.16	48.57±6.04	56.30±12.43	56.45±15.45	55.87±13.49	**68.34±7.48**
	8	68.51±5.16	59.97±1.36	65.77±5.00	66.30±10.01	60.00±4.84	58.99±10.12	56.37±11.56	66.01±9.25	**70.37±3.23**
	16	68.30±6.16	63.28±7.62	66.27±5.04	64.14±6.80	54.76±4.53	56.59±5.21	60.62±4.13	65.14±7.55	**70.07±5.19**
Car	4	62.38±4.13	50.00±0.00	62.52±3.80	58.14±4.15	61.32±3.83	62.47±2.47	60.21±4.81	**85.82±3.65**	72.69±1.52
	8	72.05±1.20	64.00±3.57	72.23±2.59	63.95±4.35	67.86±0.49	67.57±3.44	65.53±8.00	**87.43±2.56**	73.26±1.46
	16	82.42±4.13	72.26±4.43	75.77±2.71	71.35±5.33	75.56±2.88	76.94±3.04	74.02±1.01	**88.65±2.63**	79.43±1.24
Credit-g	4	52.68±4.46	50.00±0.00	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**
	8	55.52±8.88	52.22±4.90	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**
	16	58.26±5.17	56.23±4.37	59.22±11.38	58.91±8.04	57.63±7.58	55.29±4.80	51.65±4.02	**60.38±14.03**	56.60±2.22
Diabetes	4	57.09±18.84	50.00±0.00	62.35±7.48	56.28±13.01	64.22±6.78	71.71±5.31	63.96±3.32	70.42±3.69	**80.28±0.75**
	8	65.52±13.18	50.86±22.03	64.69±13.33	69.08±9.68	67.39±12.92	72.21±2.07	65.47±3.95	64.30±5.88	**79.38±1.66**
	16	73.44±0.52	65.69±6.54	71.86±3.16	73.69±3.21	73.79±6.48	71.64±5.05	66.71±0.76	67.34±2.79	**80.15±1.35**
Heart	4	70.54±3.83	50.00±0.00	59.38±3.42	67.33±15.29	**88.27±3.32**	60.76±4.00	68.19±11.17	59.74±4.49	75.66±4.59
	8	78.12±10.59	55.88±3.98	74.35±6.93	77.89±2.34	**88.78±2.38**	65.46±3.77	69.85±10.82	70.14±7.91	79.46±2.16
	16	83.02±3.70	78.62±7.14	83.66±5.91	81.45±5.05	**89.13±2.10**	67.00±7.83	68.39±11.73	81.72±3.92	83.71±1.88
Cultivars	4	53.45±10.79	50.00±0.00	46.99±6.33	49.80±15.90	**57.10±8.66**	51.38±2.48	54.28±3.73	54.39±5.61	55.63±5.24
	8	56.22±11.87	52.60±6.31	51.76±9.99	54.72±9.35	**57.26±9.52**	51.68±4.43	51.48±3.85	52.86±6.13	56.97±5.08
	16	**60.35±4.23**	56.87±2.50	57.06±9.27	54.92±8.32	60.09±7.64	54.31±6.12	57.44±3.53	56.97±2.22	57.19±5.30
NHANES	4	91.96±7.02	50.00±0.00	51.58±4.66	80.74±3.89	69.32±19.59	91.84±3.79	93.54±4.20	**99.49±0.23**	92.20±1.71
	8	92.38±8.21	92.92±4.56	55.07±4.79	85.10±5.88	68.56±18.35	86.67±5.49	94.25±3.35	**100.00±0.00**	93.29±7.01
	16	94.12±3.88	94.52±6.28	89.78±6.77	95.54±0.82	68.62±19.81	93.33±4.47	95.02±1.57	**100.00±0.00**	95.64±4.67
Sequence -type	4	66.80±4.98	50.00±0.00	67.08±2.75	71.93±4.17	60.12±6.51	93.01±1.84	91.00±4.25	75.38±5.21	**98.60±0.66**
	8	72.55±4.37	61.01±10.96	60.53±6.91	79.55±6.71	60.58±9.58	94.29±1.64	90.18±3.73	79.91±2.13	**98.70±1.44**
	16	83.10±6.86	80.31±5.79	73.14±5.29	89.59±2.33	65.12±5.28	93.46±0.72	92.09±2.90	86.97±1.97	**99.29±0.78**
Solution -mix	4	72.71±1.96	50.00±0.00	68.48±14.97	71.19±3.90	64.52±8.41	73.53±1.94	80.26±5.11	73.75±3.90	**100.00±0.00**
	8	82.91±7.74	58.31±19.17	56.67±7.64	82.42±3.72	72.04±10.34	76.34±4.26	83.19±3.70	76.57±4.80	**99.81±0.08**
	16	83.87±1.90	68.09±1.76	70.37±13.96	84.39±0.83	71.82±8.79	82.15±5.01	88.10±4.72	84.28±5.15	**99.67±0.13**
Average	4	65.47	50.00	58.22	62.93	62.36	68.44	68.69	70.26	**77.86**
	8	72.03	60.52	62.18	69.53	67.47	70.41	70.53	72.76	**79.31**
	16	76.33	69.72	71.69	74.37	69.72	72.72	73.02	76.22	**80.70**

classifying a given sequence as one of four types: arithmetic, geometric, Fibonacci, or Collatz; (13) Solution-mix for predicting whether the percentage concentration of a mixture from four solutions exceeds 0.5. Former two datasets were recently donated to the UCI Machine Learning Repository after September 2023, ensuring they were not part of the pre-training for the LLM API (gpt-3.5-turbo-0613) used by our model. The latter two are synthetic datasets which ensures they could not be memorized by the LLM API and proceeded with the evaluation.

These datasets vary in size and complexity, as detailed in Table 2. Each dataset provides a clear name and description for each attribute. Datasets like 'Communities' and

'Myocardial' include over 100 attributes, posing challenges due to limited context window size in LLM.

Baselines. We conduct comparisons with nine baselines. The first three baselines are conventional tabular learning approaches, including (1) Logistic regression (LogReg), (2) XGBoost (Chen & Guestrin, 2016), and (3) RandomForest (Ho, 1995). The next two baselines assume the use of unlabeled datasets, including (4) SCARF (Bahri et al., 2022) and (5) STUNT (Nam et al., 2023b). Note that obtaining a large quantity of unlabeled dataset may be infeasible in real-world applications. Another recent baseline is (6) TabPFN (Hollmann et al., 2023), which generates a synthetic dataset with diverse distribution to pretrain the model. The final three baselines utilize LLM, including

Table 2: Basic information of each dataset used in our experiments. The numbers in parentheses indicate the count of categorical and numerical attributes, respectively.

Data	# of samples	# of features	Label ratio (%)
Adult	48842	14 (7/7)	76:24
Bank	45211	16 (8/8)	88:12
Blood	748	4 (0/4)	76:24
Car	1728	6 (5/1)	70:22:4:4
Communities	1994	103 (1/102)	34:33:33
Credit-g	1000	20 (12/8)	70:30
Diabetes	768	8 (0/8)	65:35
Heart	918	11 (4/7)	45:55
Myocardial	1700	111 (94/17)	22:78
Cultivars	320	10 (3/7)	50:50
NHANES	6287	8 (1/7)	84:16
Sequence-type	250	5 (0/5)	40:40:10:10
Solution-mix	300	8 (0/8)	52:48

(7) In-context learning (In-context) (Wei et al., 2022a), (8) TABLET (Slack & Singh, 2023), and (9) TabLLM (Hegselmann et al., 2023). In-context learning directly embeds few-shot training samples into the input prompt without tuning. TABLET further incorporates additional hints into the prompt through an external classifier via rule-sets and prototypes to enhance the quality of inference. TabLLM employs parameter-efficient tuning technique like IA3 (Liu et al., 2022) to train the LLM with few-shot samples.

Implementation details. FeatLLM uses GPT-3.5 as its LLM backbone, yet is designed to be agnostic to the choice of LLMs (see Appendix H for results with different backbones). The temperature for the LLM inference is set to 0.5 and the top-p value is set to the default value of 1 in API. We set the number of ensembles and the number of rules for extracting to 20 and 10 respectively. Details on hyper-parameter impacts are in Figure 6 and Appendix B. We use the Adam optimizer of a learning rate 0.01 for the linear model, training for 200 epochs. We employ k-fold cross-validation for optimal epoch selection. In replicating baselines, GPT-3.5 is utilized for in-context learning, whereas T0 (Sanh et al., 2022) is employed in the TabLLM, following the original setting. Note that TabLLM restricts the use of LLM to only those with publicly available checkpoints; therefore, GPT-3.5 is not accessible in this context. Regarding traditional machine learning methods, such as LogReg, XGBoost, and RandomForest, we determined their parameters using Grid-search combined with k-fold cross validation. The value of k is set either as 2 or 4, guaranteeing that the training set includes at least one example of each class. For more detailed information on implementations, please refer to Appendix C.

Results. In Tables 1 and 3, performance comparisons across a range of datasets are presented. We repeat experiments three times with different randomness on the choice

Table 3: Evaluation results with AUC across two datasets with over 100 attributes. TabPFN and two LLM-based baselines are excluded in the table as they failed to make inference over the dataset with a large number of features.

Communities	Shots		
	4	8	16
LogReg	67.45±13.26	73.73±5.45	72.55±4.83
XGBoost	53.94±4.19	66.65±4.50	68.01±1.97
RandomForest	66.09±10.52	71.16±4.61	71.66±4.81
SCARF	66.18±9.13	72.69±3.79	73.09±2.84
STUNT	66.87±14.10	76.36±4.55	**77.29±2.56**
FeatLLM	**75.39±5.05**	**76.59±1.25**	76.25±0.64

Myocardial	Shots		
	4	8	16
LogReg	51.25±3.85	55.34±1.11	60.00±5.16
XGBoost	50.00±0.00	55.63±2.92	56.55±12.22
RandomForest	51.91±4.49	52.77±5.83	54.16±4.53
SCARF	47.70±4.10	49.37±3.41	54.31±1.42
STUNT	52.77±2.01	55.40±4.41	**61.22±3.45**
FeatLLM	**52.87±3.44**	**56.22±1.64**	55.32±9.15

of training/test sets, and present the average and standard deviation of the AUC values. FeatLLM consistently ranks as the top performer or secures the second place when compared with other baseline models. Our framework, when utilizing just 4 shots, achieves comparable performance to conventional tabular baselines that require 32 shots (see Fig. 4). Notably, while STUNT and TabPFN demonstrated significant performance improvements on certain datasets, their overall enhancement over conventional machine learning models was not substantial. Moreover, models utilizing LLM, such as in-context learning, TABLET, or TabLLM, failed to infer on tabular datasets with a large number of features. Our framework, employing both bagging and ensemble techniques, proved to be effective even on datasets with numerous features, exhibiting satisfactory performance. It is important to note that our bagging and ensemble concept could be adapted to existing LLM-based approaches. However, such an adaptation would double the number of per-sample inference, leading to a substantial increase in computational complexity and thus becoming impractical.

4.2. Analysis & Discussion

Ablation study. We analyze the contributions of major components to the performance by performing ablation studies, focusing on: (1) -Tuning: omitting the weight tuning process in the linear model, summing up the binary feature to compute the logit instead; (2) -Ensemble: omitting the ensemble process; (3) -Description: omitting the feature description, and (4) -Reasoning: omitting the Step-1 process in reasoning instruction part when generating rules.
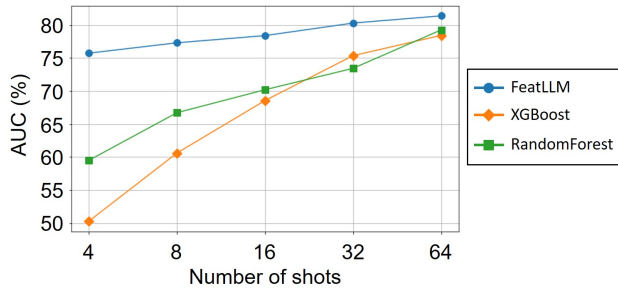
Figure 4: Performance comparison summaries among conventional tabular baselines and FeatLLM. Averaged AUC over all datasets across the number of shots are reported.

Table 4 summarizes the change in the AUC from ablations averaged across all datasets. In all cases, modifying the ablated component results in performance degradation. Interestingly, the benefit of weight tuning becomes higher when the number of shots increases. This indicates that when there is a large amount of data, accurate estimation of the importance of rules becomes feasible, making the effect of tuning significant. On the other hand, the effect of feature descriptions and reasoning instructions are high when the number of shot is small. This suggests that the efficient utilization of the prior knowledge of LLM becomes crucial for performance improvements. Lastly, the proposed ensemble approach consistently improves the performance in all settings.

Table 4: Ablation study summaries. Averaged changes of AUC with standard error after altering or omitting modules over 13 datasets with three different trials are reported.

Shot	FeatLLM	-Tuning	-Ensemble	-Description	-Reasoning
4	75.7	-1.41±1.00	-5.39±0.81	-1.76±1.06	-5.03±1.96
8	77.3	-2.72±0.93	-6.96±1.40	-1.20±0.33	-3.55±0.81
16	78.4	-2.57±0.73	-6.65±1.18	-0.26±0.31	-1.50±0.87
32	80.3	-5.75±1.19	-7.38±1.34	-0.29±0.58	-2.42±1.15
64	81.4	-4.88±1.40	-6.09±0.96	-0.70±0.54	-1.71±0.47
Avg	78.6	-3.47±0.51	-6.49±0.51	-0.84±0.28	-2.84±0.53

Training & inference time comparison. We compare the training and inference time between different methods. The experiments are performed on the Adult dataset, including a training set of 16 shots. One A100 GPU is used as the default, except for TabLLM which uses four A100 GPUs for model parallelism. Table 5 provides the runtime comparisons. Notably, FeatLLM shows a relatively low inference time, comparable to that of conventional methods. In contrast, other LLM-based approaches, such as In-context learning, TABLET, and TabLLM, exhibit much higher inference time. In terms of training time, FeatLLM is comparable with other LLM-based methods – it requires only 30 API queries, which does not impose a significant burden on the budget and even can be parallelized.

Table 5: Comparison of training (with 16 samples) and inference (over one sample) time on Adult.

Model	Training (in seconds)	Inference (in milliseconds)
LogReg	0.721	0.001
XGBoost	28.512	0.006
RandomForest	1.343	0.001
SCARF	426.859	0.002
TabPFN	0.440	1.149
STUNT	642.796	0.006
In-context†	N/A	463.000
TABLET†	0.813	523.254
TabLLM	251.242	335.127
FeatLLM†	860.094	0.006

† These models employ API queries, where the runtime is subject to the API's status at the time of use.

Quality of features generated by FeatLLM. We conduct a simple experiment to investigate the informativeness of rule-based features generated by FeatLLM in solving practical tasks. Specifically, we measure the average AUROC between the newly generated features and the target labels, and then compute the ratio of useful features with an AUROC higher than 0.5 for each dataset. Table 6 below summarizes the results. It confirms that the majority of the created rules are relevant to the target attribute and provide informative insights for addressing the target task (refer to the column "Before tuning"). Moreover, during the process of tuning the linear model with data, rules that have little relevance to the data (i.e., rules with learned weights below the threshold) are automatically filtered out (refer to the column "After tuning"). Note that the threshold was set at 0.1, taking into consideration the overall histogram of weights.

Table 6: Evaluation of feature quality before and after linear model tuning. Ratio of newly-generated features with an AUROC higher than 0.5 is reported across datasets.

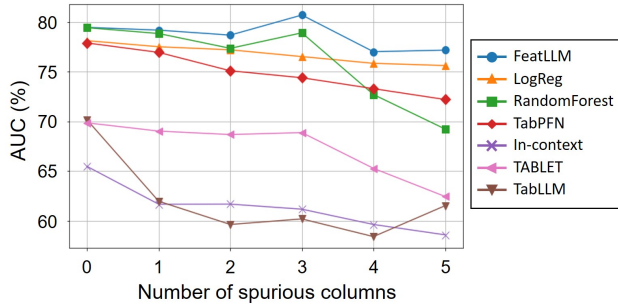| Method	Ratio of useful rules	
	Before tuning	After tuning
Adult	0.971	0.980
Bank	0.627	0.958
Blood	0.889	0.975
Car	0.712	0.849
Communities	0.767	0.889
Credit-g	0.643	0.802
Diabetes	0.974	0.972
Heart	0.800	0.924
Myocardial	0.690	0.744
Cultivars	0.775	0.927
NHANES	0.600	0.832
Sequence-type	0.896	0.985
Solution-mix	0.917	1.000
Average	0.789	0.911

Figure 5: Visualization of performance impact from spurious correlations. The results exhibit the models' performance (AUC) each time a noisy column from the Adult dataset is added to the original Heart dataset. XGBoost is excluded here due to its lower performance.

Figure 6: Effect of the number of rules per each rule extraction. Averaged AUC over all datasets is reported.

Effect of adding spurious correlations. To ascertain how effectively various methods filter out noisy spurious correlations in a low-shot regime, we conducted an analysis. The experiment utilized the Heart dataset, to which we randomly attach columns from the Adult dataset that are irrelevant to the task of the Heart dataset. We then measure the performance changes of the models while progressively increasing the number of unrelated columns. For a fair comparison, we assume the same 8 shots of labeled samples without considering an unlabeled dataset, thereby excluding methods like SCARF or STUNT from the comparison.

Figure 5 visualizes the performance changes due to spurious correlations. Compared to baselines, FeatLLM showed the least performance degradation from spurious correlations. This is likely because our framework forces the consideration of the relationship between the task and features based on prior knowledge when extracting rules, preventing the generation of rules based on irrelevant columns and thereby yielding robust results. Indeed, we observe that rules based on noisy columns accounted for only 1-2% of the total. Meanwhile, we also observe that FeatLLM can also learn spurious correlations and misunderstand causal relationships between task and features if columns from closely related data are randomly attached (see Appendix I).

Hyper-parameter analysis FeatLLM has two main hyper-parameters. One is the number of ensembles, and the other is how many rules per class to extract during each LLM query. We conducted experiments to investigate the impact of these hyper-parameters on performance. The results showed that performance improves as the number of ensembles increases, but beyond a certain point (i.e., around 20), the performance gains begin to converge (see Fig. 11 in Appendix). As for the number of rules, while there wasn't a statistically significant difference, we determined that 10 is the most appropriate number considering the trade-off
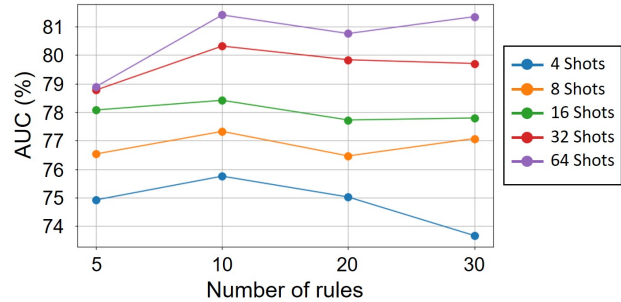
between prompt length and performance (see Fig. 6). We speculate this result is due to more rules enlarging input dimensionality, leading to an increase in information, yet also resulting in overfitting within the low-shot regime.

5. Conclusion

We present FeatLLM which advances the state-of-the-art in LLM-based few-shot tabular learning. FeatLLM goes beyond using the LLMs for predictions for each sample, and focuses on generating criteria for making predictions as feature engineers. Through rule generation and an ensemble process with bagging, FeatLLM requires minimal inference cost, needs only API access without training, and overcomes the constraints to data feature size. FeatLLM achieves high prediction performance and is a superior alternative for applying LLM on real-world tabular data in a data-efficient way.

FeatLLM is designed to focus on low-shot learing regime only. Our future goal is to expand its capabilities of crafting new features for datasets with a larger number of samples, exploring various feature types beyond rules, and bringing interpretability, which enables its application across a wider array of tasks.

Impact Statement

The proposed framework, FeatLLM, aims to utilize the prior knowledge of LLM in performing tabular learning tasks to go beyond the achievable performance with standard supervised tabular learning. To push the data efficiency in learning, our research makes important contributions to mitigate the issue of overfitting due to limited training samples, by supplementing with prior knowledge. FeatLLM is particularly beneficial for practitioners in real-world scenarios (such as from finance and healthcare industries) where labeling is challenging due to the high human cost or label collection difficulties, and prior knowledge in LLMs can be potentially very valuable (as the LLMs are pretrained

with relevant finance or healthcare related text). As one potential important future work direction towards widespread adoption, analyses of the impact of the societal biases and misinformation embedded in the prior knowledge of LLMs would be important, as the prior knowledge would be directly affecting the predictions even dominating over the observed or collected labeled data samples.

Acknowledgement

References

100,000 genomes pilot on rare-disease diagnosis in health care — preliminary report. *New England Journal of Medicine*, 385(20):1868–1880, 2021.

Anil, R., Dai, A. M., Firat, O., Johnson, M., Lepikhin, D., Passos, A., Shakeri, S., Taropa, E., Bailey, P., Chen, Z., et al. Palm 2 technical report. *arXiv preprint arXiv:2305.10403*, 2023.

Arik, S. Ö. and Pfister, T. Tabnet: Attentive interpretable tabular learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, pp. 6679–6687, 2021.

Asuncion, A. and Newman, D. Uci machine learning repository, 2007.

Bahri, D., Jiang, H., Tay, Y., and Metzler, D. Scarf: Self-supervised contrastive learning using random feature corruption. In *International Conference on Learning Representations*, 2022.

Bartlett, P. L., Montanari, A., and Rakhlin, A. Deep learning: a statistical viewpoint. *Acta numerica*, 30:87–201, 2021.

Breiman, L. Random forests. *Machine learning*, 45:5–32, 2001.

Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33: 1877–1901, 2020.

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

Chen, T., Kornblith, S., Norouzi, M., and Hinton, G. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pp. 1597–1607. PMLR, 2020.

Chen, W.-Y., Liu, Y.-C., Kira, Z., Wang, Y.-C. F., and Huang, J.-B. A closer look at few-shot classification. In *International Conference on Learning Representations*, 2018.

Creswell, A., Shanahan, M., and Higgins, I. Selection-inference: Exploiting large language models for interpretable logical reasoning. In *The Eleventh International Conference on Learning Representations*, 2022.

Dinh, T., Zeng, Y., Zhang, R., Lin, Z., Gira, M., Rajput, S., Sohn, J.-y., Papailiopoulos, D., and Lee, K. Lift: Language-interfaced fine-tuning for non-language machine learning tasks. *Advances in Neural Information Processing Systems*, 35:11763–11784, 2022.

Fan, Z., Gao, X., Mirchev, M., Roychoudhury, A., and Tan, S. H. Automated repair of programs from large language models. In *2023 IEEE/ACM 45th International Conference on Software Engineering (ICSE)*, pp. 1469–1481. IEEE, 2023.

Finch, S. E., Paek, E. S., and Choi, J. D. Leveraging large language models for automated dialogue analysis. In *Proceedings of the 24th Meeting of the Special Interest Group on Discourse and Dialogue*, pp. 202–215, 2023.

Golovenkin, S.E., S. V. R. D. S. P. N. S. O. Y. and Voino-Yasenetsky, V. Myocardial infarction complications. UCI Machine Learning Repository, 2020. DOI: https://doi.org/10.24432/C53P5M.

Hegselmann, S., Buendia, A., Lang, H., Agrawal, M., Jiang, X., and Sontag, D. Tabllm: Few-shot classification of tabular data with large language models. In *International Conference on Artificial Intelligence and Statistics*, pp. 5549–5581. PMLR, 2023.

Ho, T. K. Random decision forests. In *Proceedings of 3rd international conference on document analysis and recognition*, volume 1, pp. 278–282. IEEE, 1995.

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

Hu, E. J., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L., Chen, W., et al. Lora: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2021.

Imani, S., Du, L., and Shrivastava, H. Mathprompter: Mathematical reasoning using large language models. In *ICLR 2023 Workshop on Trustworthy and Reliable Large-Scale Machine Learning Models*, 2023.

Jakobsen, J. C., Gluud, C., Wetterslev, J., and Winkel, P. When and how should multiple imputation be used for

handling missing data in randomised clinical trials–a practical guide with flowcharts. *BMC medical research methodology*, 17(1):1–10, 2017.

Kadra, A., Lindauer, M., Hutter, F., and Grabocka, J. Well-tuned simple nets excel on tabular datasets. *Advances in neural information processing systems*, 34:23928–23941, 2021.

Kang, H. The prevention and handling of the missing data. *Korean journal of anesthesiology*, 64(5):402–406, 2013.

Kovač, G., Sawayama, M., Portelas, R., Colas, C., Dominey, P. F., and Oudeyer, P.-Y. Large language models as superpositions of cultural perspectives. *arXiv preprint arXiv:2307.07870*, 2023.

Kuhn, H. W. The hungarian method for the assignment problem. *Naval research logistics quarterly*, 2(1-2):83–97, 1955.

Levin, R., Cherepanova, V., Schwarzschild, A., Bansal, A., Bruss, C. B., Goldstein, T., Wilson, A. G., and Goldblum, M. Transfer learning with deep tabular models. In *The Eleventh International Conference on Learning Representations*, 2022.

Liu, H., Tam, D., Muqeeth, M., Mohta, J., Huang, T., Bansal, M., and Raffel, C. A. Few-shot parameter-efficient fine-tuning is better and cheaper than in-context learning. *Advances in Neural Information Processing Systems*, 35: 1950–1965, 2022.

Liu, N. F., Lin, K., Hewitt, J., Paranjape, A., Bevilacqua, M., Petroni, F., and Liang, P. Lost in the middle: How language models use long contexts. *arXiv preprint arXiv:2307.03172*, 2023.

Lu, Y., Bartolo, M., Moore, A., Riedel, S., and Stenetorp, P. Fantastically ordered prompts and where to find them: Overcoming few-shot prompt order sensitivity. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 8086–8098, 2022.

Majmundar, K. A., Goyal, S., Netrapalli, P., and Jain, P. Met: Masked encoding for tabular data. In *NeurIPS 2022 First Table Representation Workshop*, 2022.

Majumder, S., Chen, C., Al-Halah, Z., and Grauman, K. Few-shot audio-visual learning of environment acoustics. *Advances in Neural Information Processing Systems*, 35: 2522–2536, 2022.

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

Nam, J., Song, W., Park, S. H., Tack, J., Yun, S., Kim, J., and Shin, J. Semi-supervised tabular classification via in-context learning of large language models. In *Workshop on Efficient Systems for Foundation Models@ ICML2023*, 2023a.

Nam, J., Tack, J., Lee, K., Lee, H., and Shin, J. Stunt: Few-shot tabular learning with self-generated tasks from unlabeled tables. In *The Eleventh International Conference on Learning Representations*, 2023b.

OpenAI. Gpt-4 technical report, 2023.

Redmond, M. Communities and Crime. UCI Machine Learning Repository, 2009. DOI: https://doi.org/10.24432/C53W3X.

Sanh, V., Webson, A., Raffel, C., Bach, S., Sutawika, L., Alyafeai, Z., Chaffin, A., Stiegler, A., Raja, A., Dey, M., et al. Multitask prompted training enables zero-shot task generalization. In *International Conference on Learning Representations*, 2022.

Schick, T. and Schütze, H. Exploiting cloze-questions for few-shot text classification and natural language inference. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pp. 255–269, 2021.

Singhal, K., Azizi, S., Tu, T., Mahdavi, S. S., Wei, J., Chung, H. W., Scales, N., Tanwani, A., Cole-Lewis, H., Pfohl, S., et al. Large language models encode clinical knowledge. *Nature*, 620(7972):172–180, 2023.

Slack, D. and Singh, S. Tablet: Learning from instructions for tabular data. *arXiv preprint arXiv:2304.13188*, 2023.

Somepalli, G., Goldblum, M., Schwarzschild, A., Bruss, C. B., and Goldstein, T. Saint: Improved neural networks for tabular data via row attention and contrastive pre-training. *arXiv preprint arXiv:2106.01342*, 2021.

Taylor, R., Kardas, M., Cucurull, G., Scialom, T., Hartshorn, A., Saravia, E., Poulton, A., Kerkez, V., and Stojnic, R. Galactica: A large language model for science. *arXiv preprint arXiv:2211.09085*, 2022.

Ucar, T., Hajiramezanali, E., and Edwards, L. Subtab: Subsetting features of tabular data for self-supervised representation learning. *Advances in Neural Information Processing Systems*, 34:18853–18865, 2021.

Van Buuren, S. and Groothuis-Oudshoorn, K. mice: Multivariate imputation by chained equations in r. *Journal of statistical software*, 45:1–67, 2011.

Wang, X., Wei, J., Schuurmans, D., Le, Q. V., Chi, E. H., Narang, S., Chowdhery, A., and Zhou, D. Self-consistency improves chain of thought reasoning in language models. In *The Eleventh International Conference on Learning Representations*, 2022.

Wang, Z. and Sun, J. Transtab: Learning transferable tabular transformers across tables. *Advances in Neural Information Processing Systems*, 35:2902–2915, 2022.

Wang, Z., Gao, C., Xiao, C., and Sun, J. Anypredict: Foundation model for tabular prediction. *arXiv preprint arXiv:2305.12081*, 2023.

Wei, J., Tay, Y., Bommasani, R., Raffel, C., Zoph, B., Borgeaud, S., Yogatama, D., Bosma, M., Zhou, D., Metzler, D., et al. Emergent abilities of large language models. *arXiv preprint arXiv:2206.07682*, 2022a.

Wei, J., Wang, X., Schuurmans, D., Bosma, M., Xia, F., Chi, E., Le, Q. V., Zhou, D., et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in Neural Information Processing Systems*, 35: 24824–24837, 2022b.

Wilcox, A. B. and Hripcsak, G. The role of domain knowledge in automating medical text report classification. *Journal of the American Medical Informatics Association*, 10(4):330–338, 2003.

Yeh, I.-C., Yang, K.-J., and Ting, T.-M. Knowledge discovery on rfm model using bernoulli sequence. *Expert Systems with applications*, 36(3):5866–5871, 2009.

Yoon, J., Zhang, Y., Jordon, J., and van der Schaar, M. Vime: Extending the success of self-and semi-supervised learning to tabular domain. *Advances in Neural Information Processing Systems*, 33:11033–11043, 2020.

Zhang, T., Wang, S., Yan, S., Li, J., and Liu, Q. Generative table pre-training empowers models for tabular prediction. *arXiv preprint arXiv:2305.09696*, 2023.

Zhao, W. X., Zhou, K., Li, J., Tang, T., Wang, X., Hou, Y., Min, Y., Zhang, B., Zhang, J., Dong, Z., et al. A survey of large language models. *arXiv preprint arXiv:2303.18223*, 2023.

Zhu, B., Shi, X., Erickson, N., Li, M., Karypis, G., and Shoaran, M. Xtab: Cross-table pretraining for tabular transformers. *arXiv preprint arXiv:2305.06090*, 2023.

Appendix

A. Examples of Prompts and Their Corresponding Outputs

A.1. Task Description for Each Dataset

This section demonstrates creation of prompts for rule extraction for each task, along with the task descriptions we use (see Section 3.1 - Basic Information Description). The descriptions of the tasks are developed by referring to the descriptions in the original data links and previous works (Hegselmann et al., 2023), as outlined in the table below.

Table 7: Description of the tasks associated with each dataset utilized in the prompt. The text in blue describes the label information, providing the list of possible answer classes.

Data	Task description
Adult	Does this person earn more than 50000 dollars per year? Yes or no?
Bank	Does this client subscribe to a term deposit? Yes or no?
Blood	Did the person donate blood? Yes or no?
Car	How would you rate the decision to buy this car? Unacceptable, acceptable, good or very good?
Communities	How high will the rate of violent crimes per 100K population be in this area. Low, medium, or high?
Credit-g	Does this person receive a credit? Yes or no?
Diabetes	Does this patient have diabetes? Yes or no?
Heart	Does the coronary angiography of this patient show a heart disease? Yes or no?
Myocardial	Does the myocardial infarction complications data of this patient show chronic heart failure? Yes or no?
Cultivars	How high will the grain yield of this soybean cultivar. Low or high?
NHANES	Predict this person's age group from the given record. Senior or non-senior?
Sequence-type	What is the type of following sequence? Arithmetic, geometric, fibonacci, or collatz?
Solution-mix	Given the volumes and concentrations of four solutions, does the percent concentration of the mixed solution over 0.5? Yes or no?

A.2. Example Prompt for Extracting Rules

We introduce an example prompt designed for rule extraction in the 'Heart' dataset. This prompt utilizes the template illustrated in Figure 2, with relevant information filled in. We assume the 4-shot setting.

You are an expert. Given the task description and the list of features and data examples, you are extracting conditions for each answer class to solve the task.

Task: Does the coronary angiography of this patient show a heart disease? Yes or no?

Features:
- Age: age of the patient (numerical variable)
- Sex: sex of the patient (categorical variable with categories [M, F])
- ChestPainType: chest pain type (categorical variable with categories [ATA, NAP, ASY, TA])
- RestingBP: resting blood pressure [mm Hg] (numerical variable)
- Cholesterol: serum cholesterol [mm/dl] (numerical variable)
- FastingBS: fasting blood sugar [1: if FastingBS > 120 mg/dl, 0: otherwise] (numerical variable)
- RestingECG: resting electrocardiogram results (categorical variable with categories [Normal, ST, LVH])
- MaxHR: maximum heart rate achieved (numerical variable)
- ExerciseAngina: exercise-induced angina (categorical variable with categories [N, Y])
- Oldpeak: oldpeak = ST [Numeric value measured in depression] (numerical variable)
- ST_Slope: the slope of the peak exercise ST segment (categorical variable with categories [Up, Flat, Down])

Examples:
Age is 63. Sex is M. ChestPainType is NAP. RestingBP is 130. Cholesterol is 0. FastingBS is 1. RestingECG is ST. MaxHR is 160. ExerciseAngina is N. Oldpeak is 3.0. ST_Slope is Flat.
Answer: no
Age is 39. Sex is M. ChestPainType is ATA. RestingBP is 120. Cholesterol is 204. FastingBS is 0. RestingECG is Normal. MaxHR is 145. ExerciseAngina is N. Oldpeak is 0.0. ST_Slope is Up.
Answer: no
Age is 58. Sex is M. ChestPainType is NAP. RestingBP is 160. Cholesterol is 211. FastingBS is 1. RestingECG is ST. MaxHR is 92. ExerciseAngina is N. Oldpeak is 0.0. ST_Slope is Flat.
Answer: yes
Age is 55. Sex is M. ChestPainType is ASY. RestingBP is 160. Cholesterol is 289. FastingBS is 0. RestingECG is LVH. MaxHR is 145. ExerciseAngina is Y. Oldpeak is 0.8. ST_Slope is Flat.
Answer: yes

Let's first understand the problem and solve the problem step by step.
Step 1. Analyze the causal relationship or tendency between each feature and task description based on general knowledge and common sense within a short sentence.
Step 2. Based on the above examples and Step 1's results, infer 10 different conditions per answer, following the format below. The condition should make sense, well match examples, and must match the format for [condition] according to value type.

Format for Response:
10 different conditions for class "no":
- [Condition]
...

10 different conditions for class "yes":
- [Condition]
...

Format for [Condition]:
For the categorical variable only,
- [Feature] is in [List of categories]
For the numerical variable only,
- [Feature] (> or >= or < or <=) [Value]
- [Feature] is within range of [Value_start, Value_end]

Answer: Step 1.

Figure 7: Example prompt for rule extraction in the 'Heart' dataset

A.3. Example Outcome from Rule Extraction

We demonstrate how rules are derived through prompts by showing an example of an outcome. If we input the prompt from the previous subsection (see Fig. 7), the following result is obtained.

Step 1. The relationship between each feature and the task description:

- Age: There is a tendency for older age to be associated with a higher risk of heart disease.
- Sex: There may be a difference in the prevalence of heart disease between males and females, with males potentially having a higher risk.
- ChestPainType: Certain types of chest pain, such as NAP and ASY, may be indicative of heart disease.
- RestingBP: Higher resting blood pressure levels may be associated with a higher risk of heart disease.
- Cholesterol: Higher levels of serum cholesterol may be associated with a higher risk of heart disease.
- FastingBS: Fasting blood sugar levels above 120 mg/dl may be indicative of heart disease.
- RestingECG: Abnormal resting electrocardiogram results, such as ST and LVH, may be indicative of heart disease.
- MaxHR: Lower maximum heart rate achieved may be associated with a higher risk of heart disease.
- ExerciseAngina: The presence of exercise-induced angina may be indicative of heart disease.
- Oldpeak: Higher values of oldpeak (ST depression) may be associated with a higher risk of heart disease.
- ST_Slope: The slope of the peak exercise ST segment may provide information about the presence of heart disease.

Step 2. Inferred conditions for each answer class:

10 different conditions for class "no":
- Age is within range of [40, 60]
- Sex is in [F]
- ChestPainType is in [ATA, NAP]
- RestingBP (< 140)
- Cholesterol (< 200)
- FastingBS is in [0]
- RestingECG is in [Normal]
- MaxHR (> 140)
- ExerciseAngina is in [N]
- Oldpeak (< 1.0)

10 different conditions for class "yes":
- Age is within range of [50, 70]
- Sex is in [M]
- ChestPainType is in [ASY, TA]
- RestingBP (> 140)
- Cholesterol (> 200)
- FastingBS is in [1]
- RestingECG is in [ST, LVH]
- MaxHR (< 150)
- ExerciseAngina is in [Y]
- Oldpeak (> 1.0)

Figure 8: Example outcome from rule extraction in the 'Heart' dataset

A.4. Example Prompt for Parsing Rules

Utilizing the given rules, we convert them into code capable of transforming each data point into a binary feature. This binary feature indicates whether the sample adheres to the rules associated with its class. Below is an example prompt for parsing rules, employing the same example as in the previous subsection.

Provide me a python code for function, given description below.

Function name: extracting_features_no

Input: Dataframe df_input

Input Features:
- Age: age of the patient (numerical variable)
- Sex: sex of the patient (categorical variable with categories [M, F])
- ChestPainType: chest pain type (categorical variable with categories [ATA, NAP, ASY, TA])
- RestingBP: resting blood pressure [mm Hg] (numerical variable)
- Cholesterol: serum cholesterol [mm/dl] (numerical variable)
- FastingBS: fasting blood sugar [1: if FastingBS $>$ 120 mg/dl, 0: otherwise] (numerical variable)
- RestingECG: resting electrocardiogram results (categorical variable with categories [Normal, ST, LVH])
- MaxHR: maximum heart rate achieved (numerical variable)
- ExerciseAngina: exercise-induced angina (categorical variable with categories [N, Y])
- Oldpeak: oldpeak = ST [Numeric value measured in depression] (numerical variable)
- ST_Slope: the slope of the peak exercise ST segment (categorical variable with categories [Up, Flat, Down])

Output: Dataframe df_output. Create a new dataframe df_output. Each column in df_output refers whether the selected column in df_input follows the condition (1) or not (0). Be sure that the function code well matches with its feature type (i.e., numerical, categorical).

Conditions:
- Age is within range of [40, 60]
- Sex is in [F]
- ChestPainType is in [ATA, NAP]
- RestingBP ($<$ 140)
- Cholesterol ($<$ 200)
- FastingBS is in [0]
- RestingECG is in [Normal]
- MaxHR ($>$ 140)
- ExerciseAngina is in [N]
- Oldpeak ($<$ 1.0)

Wrap only the function part with $<$start$>$ and $<$end$>$, and do not add any comments, descriptions, and package importing lines in the code.

Figure 9: Example prompt for parsing rules in the 'Heart' dataset

A.5. Example Outcome After Parsing

The result of the parsing request is exemplified as follows.

```
<start>
def extracting_features_no(df_input):

    df_output = pd.DataFrame()
    df_output['Age'] = df_input['Age'].apply(lambda x: 1 if x >= 40 and x <= 60 else 0)
    df_output['Sex'] = df_input['Sex'].apply(lambda x: 1 if x == 'F' else 0)
    df_output['ChestPainType'] = df_input['ChestPainType'].apply(lambda x: 1 if x in ['ATA', 'NAP'] else 0)
    df_output['RestingBP'] = df_input['RestingBP'].apply(lambda x: 1 if x < 140 else 0)
    df_output['Cholesterol'] = df_input['Cholesterol'].apply(lambda x: 1 if x < 200 else 0)
    df_output['FastingBS'] = df_input['FastingBS'].apply(lambda x: 1 if x == 0 else 0)
    df_output['RestingECG'] = df_input['RestingECG'].apply(lambda x: 1 if x == 'Normal' else 0)
    df_output['MaxHR'] = df_input['MaxHR'].apply(lambda x: 1 if x > 140 else 0)
    df_output['ExerciseAngina'] = df_input['ExerciseAngina'].apply(lambda x: 1 if x = 'N' else 0)
    df_output['Oldpeak'] = df_input['Oldpeak'].apply(lambda x: 1 if x < 1.0 else 0)


    return df_output
<end>
```

Figure 10: Example outcome after parsing rules in the 'Heart' dataset

B. Hyper-parameter analysis

FeatLLM has two main hyper-parameters. One is the number of ensembles, and the other is how many rules per class to extract during each LLM query. We conducted experiments to investigate the impact of these hyper-parameters on performance. The results showed that performance improves as the number of ensembles increases, but beyond a certain point (i.e., around 20), the performance gains begin to converge (see Fig. 11). For an analysis of the impact of the number of rules, please refer to Section 4.2.
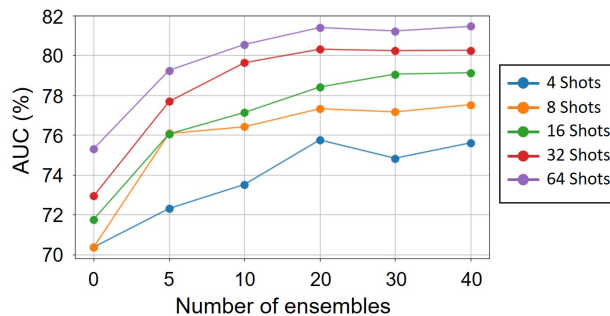

Figure 11: Effect of the number of ensembles. Averaged AUC over all datasets is reported.

C. Baseline Implementation Details

In this section, we provide a detailed information on the implementation of baselines. In our study, we adhere to the settings and parameters outlined in the original papers for various baselines. During model training, the dataset was divided, allocating 20% as the test set. This division was carried out using stratified sampling to ensure a representative distribution of classes. From the remaining data, k-shot samples were selected, constructing a balanced training set. We repeated experiments multiple times with different randomness on the choice of training/test sets.

For conventional machine learning methods, including logistic regression (LogReg), RandomForest, and XGBoost, we determined their parameters using Grid-search combined with k-shot cross validation. Following the previous work (Hegselmann et al., 2023), the search space of each parameter is described in Tables below.

Parameter	Search space
penalty	l1, l2
C	100, 10, 1, 1e-1, 1e-2, 1e-3, 1e-4, 1e-5

Table 8: Hyperparameter search space for LogReg

Parameter	Search space
bootstrap	True, False
max_depth	2, 4, 6, 8, 10, 12
n_estimators	2, 4, 8, 16, 32, 64, 128, 256

Table 9: Hyperparameter search space for RandomForest

Parameter	Search space
max_depth	2, 4, 6, 8, 10, 12
alpha	1e-8, 1e-7, 1e-6, 1e-5, 1e-4, 1e-3, 1e-2, 1e-1, 1
lambda	1e-8, 1e-7, 1e-6, 1e-5, 1e-4, 1e-3, 1e-2, 1e-1, 1
eta	0.01, 0.03, 0.1, 0.3

Table 10: Hyperparameter search space for XGBoost

For baseline models that incorporate unlabeled data in training, such as SCARF and STUNT, any data not used in the test set or as k-shot training samples is treated as the unlabeled dataset. To ensure that there are no missing values in the training and test data, columns representing more than 20% of the entire dataset were removed, similarly following the literature (Kang, 2013; Jakobsen et al., 2017). Subsequently, any rows containing missing values were also eliminated. For TabPFN, we utilized the official GitHub repository[7] to reproduce results, maintaining the default parameter settings as specified.

Regarding LLM-based baselines, we utilized GPT-3.5 for in-context learning, while T0 was chosen for the TabLLM, in line with their original usage settings. To calculate the AUC for the in-context learning and TABLET with the GPT-3.5 API, we performed the inference process three times, introducing variability with a nonzero temperature setting (i.e., 1). The results were then averaged to estimate the probabilities for each class. In contrast, for the TabLLM, we employed T0, utilizing its publicly available checkpoint. We directly calculated the probability of the class name token to estimate the class probabilities.

---

[7]https://github.com/automl/TabPFN

D. Diversity of Features Generated by FeatLLM

In our model, to guarantee diversity of features (i.e., rules) generated from the LLM in each iteration, we set the temperature of LLMs to a value greater than 0 (i.e., 0.5) to introduce randomness into the generation, applied bagging strategies (i.e., feature and sample bagging), and shuffled the order of training samples in the prompt for each iteration. To empirically verify the diversity of features, we conducted the following analysis:

1. Given two sets of rules from different iterations, we applied each set's rules across the dataset to generate binary features.
2. We used the Hungarian Maximum Matching Algorithm (Kuhn, 1955) to create a bipartite graph connecting pairs of binary features between the two sets with the highest correlation.
3. The average correlation of the selected pairs was calculated.
4. This process (steps 1-3) was repeated for all pairs of iterations to determine the average and variance of the correlations.

We conducted an ablation study to assess the effects of three components (i.e., temperature, bagging, instance shuffling) on enhancing rule diversity. Table 11 below presents the results of average correlation across iterations, where a lower correlation signifies a more diverse set of extracted rules. We observed that the average correlation was 0.368 with a variance of 0.341, indicating a generally low correlation and confirming the generation of diverse rules. Additionally, we noted an increase in correlation when each of these components was removed, demonstrating that each component contributes to diversity.

Table 11: Ablation study summaries on rule diversity.

Ablations	Correlation between rule sets across iterations
All strategies	0.368 ± 0.341
without bagging	0.384 ± 0.327
without instance shuffling	0.531 ± 0.290
with zero temperature	0.414 ± 0.373

E. Prior Knowledge vs. Training Samples.

FeatLLM simultaneously utilizes prior knowledge from LLMs and information extracted from training samples to derive rules. This raises the question: which source of knowledge does the framework prioritize when generating rules? To investigate this, we conducted a simple experiment. Keeping the training data unchanged, we reversed its labels and evaluated the resulting rule set. The experiment used Adult, and to focus solely on the rule set, we removed ensemble and weight tuning processes from the evaluation. Table 12 shows the results for rule sets evaluated with both the original and flipped data. Despite the label reversal, our framework tends to more focused on prior knowledge. However, we also observed a significant increase in the standard deviation of AUC, indicating variability in each trial, with some prioritizing training samples and others prior knowledge, seemingly at random. These findings underscore the importance of using ensemble process to cover both scenarios effectively.

Table 12: Comparison of AUC between original data and flipped data whose labels in training samples are reversely flipped. Adult dataset is utilized.

Shot	4	8	16	32	64
Original	83.66±6.76	83.48±7.21	84.51±2.64	84.16±5.32	84.07±3.38
Flipped	63.91±22.54	71.35±18.53	66.81±18.57	65.94±18.57	56.14±17.39

F. Error Rates of Automated Parsing via LLM

FeatLLM utilizes rules extracted through an LLM for parsing and converting data. However, since the conversion code is derived from the LLM, there are some cases where the generated code does not function properly and fails. We have conducted experiments and recorded how frequently such failures occur across different datasets. According to Table 13, the error rate varies per dataset, with higher failure rates often seen in data containing many categorical features. This is because GPT-3.5 sometimes makes syntactic errors in handling categorical columns in the pandas library, and the probability of errors increases with more rules pertaining to categorical variables. We believe that using a model specialized in code generation or a higher-performance model (e.g., GPT-4) could significantly reduce this error rate.

Table 13: Error rate (%) of automated parsing for each dataset.

Data	Error rate
Adult	14.67
Bank	35.33
Blood	1.33
Car	1.67
Communities	1.00
Credit-g	40.33
Diabetes	1.67
Heart	2.67
Myocardial	13.00
Cultivars	12.85
NHANES	0.00
Sequence-type	35.36
Solution-mix	18.43

G. Handling missing data.

Given missing values in data, FeatLLM requires an additional processing step to handle them. Specifically, when creating binary features through rules, we need to decide how to handle these missing values. We have compared three strategies to address this:

- Fill zero: Assuming that the rules related to the missing value are not satisfied, fill these with zero.

- Fill 0.5: Since the absence of a value leaves it unclear whether the rule is satisfied or not, fill the value as 0.5.

- Imputation: Using the binary matrix of the training data, impute the missing values via MICE (Van Buuren & Groothuis-Oudshoorn, 2011) considering the relationships between the rules.

Table 14 presents a summary of performance comparisons based on different strategies in relation to the missing ratio. Compared to other strategies, imputation demonstrated the lowest performance degradation. We speculate that the effectiveness of imputation, even with a limited number of shots, is due to the reduced complexity of imputation, a result of converting data to binary form through rules.

Table 14: Missing value analysis. Averaged changes of AUC with standard error over 13 datasets across different missing ratios are reported.

Missing ratio	0	0.05	0.1	0.15	0.2
Fill zero		-0.49±0.23	-1.57±0.21	-1.75±0.28	-2.15±0.44
Fill 0.5	78.64	-0.53±0.27	-1.55±0.22	-1.72±0.34	-2.07±0.50
Imputation		-0.29±0.20	-1.21±0.27	-1.41±0.29	-1.88±0.28

H. Varying LLM backbones

To investigate the impact of using different LLMs on our framework, given their distinct prior knowledge and reasoning abilities from being trained on various text corpora, we measured performance using not only GPT-3.5 but also the PaLM 2 Text-Unicorn model as a backbone for comparison. The results in Table 15 indicate that certain tasks show greater improvement with specific LLMs.

Table 15: Effect of varying LLM backbones: GPT-3.5 vs. PaLM 2-Unicorn

Data	Shot	FeatLLM (GPT-3.5)	FeatLLM (PaLM 2-Unicorn)
Adult	4	**86.68±0.86**	82.99±2.86
	8	**87.89±0.06**	83.24±0.87
	16	**87.54±0.50**	83.19±1.89
	32	**87.09±0.58**	83.93±3.50
	64	**87.77±0.31**	86.18±1.79
Bank	4	**70.45±3.69**	60.89±6.07
	8	**75.85±6.66**	65.44±8.26
	16	**78.41±1.08**	71.86±1.33
	32	**78.37±4.50**	76.36±4.45
	64	**81.18±6.17**	79.66±4.22
Blood	4	**68.34±7.48**	60.30±13.99
	8	**70.37±3.23**	67.67±1.25
	16	**70.07±5.19**	67.54±3.39
	32	**71.13±4.38**	69.61±2.07
	64	**71.04±4.36**	69.83±5.03
Car	4	**72.69±1.52**	66.92±7.86
	8	**73.26±1.46**	71.59±1.53
	16	**79.43±1.24**	74.40±1.87
	32	**85.01±1.36**	78.10±1.67
	64	**86.78±0.90**	80.22±3.25
Communities	4	**75.39±5.05**	73.87±5.56
	8	76.59±1.25	**78.57±1.46**
	16	76.25±0.64	**77.87±2.09**
	32	76.85±1.38	**78.92±2.17**
	64	78.62±1.39	**79.38±1.66**
Credit-g	4	**55.94±1.10**	54.00±5.91
	8	**57.42±3.10**	54.20±3.50
	16	56.60±2.22	**57.35±6.20**
	32	**61.79±10.25**	58.49±5.18
	64	**66.43±2.90**	66.09±7.29
Diabetes	4	**80.28±0.75**	78.40±2.20
	8	**79.38±1.66**	78.27±1.48
	16	**80.15±1.35**	79.99±1.45
	32	**80.06±1.18**	79.92±1.91
	64	**80.91±1.62**	79.20±0.43
Heart	4	**75.66±4.59**	75.12±3.97
	8	79.46±2.16	**82.07±10.08**
	16	83.71±1.88	**86.18±3.83**
	32	87.19±3.66	**87.99±3.26**
	64	88.08±4.11	**88.28±3.34**
Myocardial	4	52.87±3.44	**59.48±7.25**
	8	56.22±1.64	**61.94±2.35**
	16	55.32±9.15	**62.17±3.58**
	32	60.02±4.02	**60.72±5.18**
	64	**61.47±3.91**	61.25±5.60

I. Additional Experiments on Effect of Adding Spurious Correlations

In the previous section 4.2, "Effect of adding spurious correlations," we experimented with randomly mixing irrelevant columns from the Adult dataset into the Heart dataset to test for the removal of spurious correlations. In this section, we explored whether similar effects occur when columns from the Myocardial dataset, which is in a similar domain to the Heart dataset, are added. According to Figure 12, unlike results in our main experiment, FeatLLM showed unstable performance. This instability arises because many columns from the Myocardial dataset are relevant to the actual task of the Heart dataset. When these columns are combined with random values, FeatLLM tends to extract more rules based on these columns, leading to overfitting on noisy information. Indeed, we observed that rules based on noisy columns represented 7-8% of the total, significantly more than the 1-2% in the original experiments with the Adult dataset. It can also be observed that other LLM-based methodologies experience a significant decrease in performance. However, as the number of training shots increases (e.g., 64), the linear model will automatically measure the importance, which can also reduce the weight given to irrelevant columns, thereby decreasing such instability.
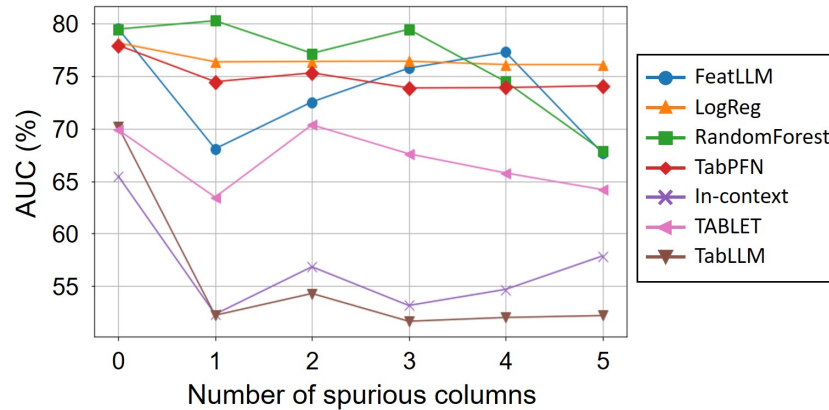
Figure 12: Visualization of performance impact from spurious correlations. The results exhibit the models' performance (AUC) each time a noisy column from the Myocardial dataset is added to the original Heart dataset. XGBoost is excluded here due to its lower performance.

J. Qualitative Analysis

Finally, to verify whether the rules deemed important by our framework are indeed significant from the perspective of the actual task definition, we conducted a qualitative analysis. We extracted the rules with the highest weights based on a linear model, selecting three for each data and answer class, and reported them in Table 16. It was observed that each rule intuitively aligns well with the task description.

Table 16: The top-3 rules for each data and answer class, ranked by their measured importance. Importance was determined based on the weights of a trained linear model, with rules extracted in descending order of importance.

Data	Label	Top-3 rules
Adult	No	educational-num <= 9 capital-gain <= 0 marital-status in [Never-married, Widowed, Divorced, Separated, Married-spouse-absent]
	Yes	educational-num >10 capital-gain >0 occupation in [Machine-op-inspct, Protective-serv, Adm-clerical, Armed-Forces]
Bank	No	contact = unknown age is within [18, 30] education in [secondary, unknown]
	Yes	balance >1000 loan = no age is within [40, 60]
Blood	No	frequency <=2 monetary <= 500 time <= 16
	Yes	recency >16 frequency >2 monetary >500
Car	Very good	doors in [4, 5more] safety = high lug_boot = big
	Good	maint = low safety = high buying in [medium, low]
	Acceptable	maint = high doors = 2 buying = very high
	Unacceptable	safety = low buying = very high doors = 2
Communities	High	NumIlleg >= 0.02 PctHousOwnOcc <= 0.4 RentLowQ >= 0.6
	Medium	MedOwnCostPctIncNoMtg is within [0.3, 0.6] PctBSorMore is within [0.3, 0.6] PctHousOwnOcc is within [0.4, 0.8]
	Low	NumIlleg <= 0.01 PctImmigRec8 <= 0.2 PctHousOwnOcc >= 0.8
Credit-g	No	existing_credits >2 property_magnitude in [life insurance, no known property] housing in [rent, for free]
	Yes	num_dependents <= 1 housing not in [rent, for free] existing_credits <= 2
Diabetes	No	Glucose <100 BMI <25 Glucose <110 & BloodPressure <70
	Yes	BMI >= 30 & Age >= 40

		SkinThickness >= 20 & Age >= 45
		Insulin >= 150
Heart	No	Cholesterol <200
		Age <50
		ExerciseAngina = N
	Yes	Age >= 50
		ExerciseAngina = Y
		FastingBS = 1
Myocardial	No	ZSN_A = 'there is no chronic heart failure'
		GEPAR_S_n = yes
		lat_im = 'QRS has no changes'
	Yes	ZSN_A = 'I stage'
		Iat_im = 'QRS is like Qr-complex'
		n_r_ecg_p_04 = no
Cultivars	High	Cultivar in ['ELISA IPRO', 'MANU IPRO', 'TMG 22X83I2X', '83IX84RSF I2X', 'MANU IPRO']
		NGL >= 2.56
		NLP >= 50
	Low	Cultivar in ['BRASMAX BÔNUS IPRO', 'GNS7700 IPRO', 'MONSOY M8606I2X']
		Cultivar in ['97Y97 IPRO', 'NEO 760 CE', 'MONSOY M8606I2X', 'GNS7700 IPRO']
		Cultivar in ['NEO 760 CE', 'FTR 3868 IPRO', 'GNS7700 IPRO', 'MONSOY M8606I2X']
NHANES	Non-senior	RIDAGEYR <60
		LBXGLT <150
		BMXBMI <30
	Senior	RIDAGEYR >= 60
		LBXGLT >150
		BMXBMI >= 30
Sequence-type	Arithmetic	(num2 + (num2 - num1)) == num3
		(num3 + (num3 - num2)) == num4
		(num4 + (num4 - num3)) == num5
		(num3 - (num2 - num1)) == num2
	Geometric	(num1 * (num3 / num2)) == num2
		(num2 * (num4 / num3)) == num3
		(num3 * (num2 / num1)) == num4
	Fibonacci	(num1 + num2) == num3
		(num2 + num3) == num4
		(num3 + num4) == num5
	Collatz	num2 % 2 == 0
		num4 % 2 == 0
		num1 > num2
Solution-mix	No	(Concentration1 * Volume1 + ... + Concentration4 * Volume4) / (Volume1 + ... + Volume4) <= 0.5
		Concentration1 <= 0.1
		Concentration3 <= 0.3 and Concentration4 <= 0.3
	Yes	(Concentration1 * Volume1 + ... + Concentration4 * Volume4) / (Volume1 + ... + Volume4) >0.5
		Volume1 <= 0.5
		Concentration1 >0.5

K. Full Results

We here present full results which were not included in the main manuscript due to space constraints.

K.1. Main Evaluation

Table 17: Evaluation results, showing AUC across 13 datasets. Best performances are bolded, and our framework's performances, when second-best, are underlined. Failed cases are represented as 'N/A'.

Data	Shot	LogReg	XGBoost	RandomForest	SCARF	TabPFN	STUNT	In-context	TABLET	TabLLM	Ours
Adult	4	72.10±12.30	50.00±0.00	53.65±24.42	58.34±15.42	60.89±23.28	67.43±29.61	77.51±5.24	75.29±12.24	83.57±2.69	**86.68±0.86**
	8	76.02±3.37	59.19±6.92	74.20±7.39	72.42±8.95	70.42±9.96	82.16±6.93	79.30±2.89	77.56±7.56	83.52±4.30	**87.89±0.06**
	16	75.20±5.10	60.68±13.92	70.96±13.46	75.63±9.56	70.34±9.96	80.57±10.93	79.50±4.57	79.74±5.64	83.23±2.45	**87.54±0.50**
	32	82.02±3.76	74.49±13.78	69.90±13.97	75.80±10.21	68.77±8.09	78.08±15.15	81.89±4.04	78.08±6.70	82.60±4.14	**87.09±0.58**
	64	81.94±5.16	79.56±2.18	82.36±2.73	80.89±2.82	78.16±4.43	86.01±0.16	N/A	N/A	84.88±0.97	**87.77±0.31**
Bank	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	**70.45±3.69**
	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	**75.85±6.66**
	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	**78.41±1.08**
	32	**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
	64	**82.27±1.61**	81.92±1.00	79.55±3.19	68.28±3.97	82.14±2.28	72.26±1.62	N/A	N/A	70.83±3.43	81.18±6.17
Blood	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	**68.34±7.48**
	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	**70.37±3.23**
	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	**70.07±5.19**
	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	**71.13±4.38**
	64	71.76±2.56	69.46±2.96	68.83±5.61	72.75±4.36	**73.88±1.97**	61.75±2.19	65.79±2.05	63.47±7.36	70.88±1.58	71.04±4.36
Car	4	62.38±4.13	50.00±0.00	58.75±3.82	62.52±3.80	58.14±4.15	61.32±3.83	62.47±2.47	60.21±4.81	**85.82±3.65**	72.69±1.52
	8	72.05±1.20	64.00±3.57	65.49±6.29	72.23±2.59	63.95±4.35	67.86±0.49	67.57±3.44	65.53±8.00	**87.43±2.56**	73.26±1.46
	16	82.42±4.13	72.26±4.43	77.12±4.45	75.77±2.71	71.35±5.33	75.56±2.88	76.94±3.04	74.02±1.01	**88.65±2.63**	79.43±1.24
	32	87.93±3.23	86.39±1.42	86.79±4.31	81.19±2.42	81.00±2.81	82.29±2.34	81.64±2.52	76.44±4.02	**89.02±1.50**	85.01±1.36
	64	91.78±2.47	91.67±1.26	**93.77±1.23**	84.22±1.93	86.73±3.10	84.45±1.69	77.65±3.74	76.13±1.17	92.18±0.47	86.78±0.90
Commun ities	4	67.45±13.26	53.94±4.19	66.09±10.52	66.18±9.13	N/A	66.87±14.10	N/A	N/A	N/A	**75.39±5.05**
	8	73.73±5.45	66.65±4.50	71.16±4.61	72.69±3.79	N/A	76.36±4.55	N/A	N/A	N/A	**76.59±1.25**
	16	72.55±4.83	68.01±1.97	71.66±4.81	73.09±2.84	N/A	**77.29±2.56**	N/A	N/A	N/A	76.25±0.64
	32	73.65±3.04	72.72±3.64	73.25±6.11	76.09±3.40	N/A	**78.60±2.59**	N/A	N/A	N/A	76.85±1.38
	64	77.78±5.16	78.33±4.52	**79.05±3.80**	77.79±3.43	N/A	79.00±1.61	N/A	N/A	N/A	78.62±1.39
Credit-g	4	52.68±4.46	50.00±0.00	**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
	8	55.52±8.88	52.22±4.90	**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
	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	**60.38±14.03**	56.60±2.22
	32	67.85±5.78	65.33±6.28	57.48±5.73	**72.60±7.18**	66.27±5.06	63.24±5.47	N/A	N/A	68.64±3.86	61.79±10.25
	64	72.77±9.29	70.79±2.34	68.53±4.99	**73.12±6.23**	68.95±6.14	70.97±4.95	N/A	N/A	70.80±4.09	66.43±2.90
Diabetes	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	**80.28±0.75**
	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	**79.38±1.66**
	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	**80.15±1.35**
	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	**80.06±1.18**
	64	74.52±1.59	72.56±3.17	76.92±2.39	74.44±4.13	77.82±3.49	78.64±3.32	70.22±4.09	69.27±6.15	71.56±4.55	**80.91±1.62**
Heart	4	70.54±3.83	50.00±0.00	70.85±2.02	59.38±3.42	67.33±15.29	**88.27±3.32**	60.76±4.00	68.19±11.17	59.74±4.49	75.66±4.59
	8	78.12±10.59	55.88±3.98	79.43±4.28	74.35±6.93	77.89±2.34	**88.78±2.38**	65.46±3.77	69.85±10.82	70.14±7.91	79.46±2.16
	16	83.02±3.70	78.62±7.14	83.45±3.95	83.66±5.91	81.45±5.05	**89.13±2.10**	67.00±7.83	68.39±11.73	81.72±3.92	83.71±1.88
	32	84.84±3.53	87.11±1.22	88.77±2.36	88.45±2.43	88.00±2.34	**89.65±3.04**	71.94±3.88	71.90±9.07	87.43±2.32	87.19±3.66
	64	89.74±3.30	89.99±3.82	89.74±2.62	**90.97±1.55**	90.20±2.57	89.62±3.16	N/A	N/A	89.78±2.59	88.08±4.11
Myocardial	4	51.25±3.85	50.00±0.00	51.91±4.49	47.70±4.10	N/A	52.77±2.01	N/A	N/A	N/A	**52.87±3.44**
	8	55.34±1.11	55.63±2.92	52.77±5.83	49.37±3.41	N/A	55.40±4.41	N/A	N/A	N/A	**56.22±1.64**
	16	60.00±5.16	56.55±12.22	54.16±4.53	54.31±1.42	N/A	**61.22±3.45**	N/A	N/A	N/A	55.32±9.15
	32	58.63±0.96	57.31±5.31	46.43±5.02	53.52±0.74	N/A	**60.76±1.58**	N/A	N/A	N/A	60.02±4.02
	64	57.04±1.94	56.18±2.85	55.00±4.73	54.41±2.00	N/A	59.79±0.56	N/A	N/A	N/A	**61.47±3.91**
Cultivars	4	53.45±10.79	50.00±0.00	48.29±10.29	46.99±6.33	49.80±15.90	**57.10±8.66**	51.38±2.48	54.28±3.73	54.39±5.61	55.63±5.24
	8	56.22±11.87	52.60±6.31	50.41±4.93	51.76±9.99	54.72±9.35	**57.26±9.52**	51.68±4.43	51.48±3.85	52.86±6.13	56.97±5.08
	16	**60.35±4.23**	56.87±2.50	55.89±3.81	57.06±9.27	54.92±8.32	60.09±7.64	54.31±6.12	57.44±3.53	56.97±2.22	57.19±5.30
	32	**60.94±12.64**	50.81±6.53	52.64±2.13	57.29±6.98	55.08±10.85	60.48±6.51	N/A	N/A	58.50±2.65	59.62±7.43
	64	**65.62±5.53**	49.82±3.43	53.87±5.37	59.02±4.58	56.25±7.36	61.07±6.77	N/A	N/A	60.32±2.60	59.14±4.79
NHANES	4	91.96±7.02	50.00±0.00	70.11±12.83	51.58±4.66	80.74±3.89	69.32±19.59	91.84±3.79	93.54±4.20	**99.49±0.23**	92.20±1.71
	8	92.38±8.21	92.92±4.56	87.81±3.42	55.07±4.79	85.10±5.88	68.56±18.35	86.67±5.49	94.25±3.35	**100.00±0.00**	93.29±7.01
	16	94.12±3.88	94.52±6.28	94.70±0.25	89.78±6.77	95.54±0.82	68.62±19.81	93.33±4.47	95.02±1.57	**100.00±0.00**	95.64±4.67
	32	97.66±0.53	98.40±2.77	97.61±2.56	94.43±2.17	98.05±0.66	75.06±3.56	88.54±5.40	95.82±3.71	**100.00±0.00**	97.29±1.28
	64	98.98±0.64	100.00±0.00	99.85±0.24	95.74±1.60	99.21±0.20	80.29±4.56	N/A	N/A	**100.00±0.00**	98.32±0.65
Sequence -type	4	66.80±4.98	50.00±0.00	67.81±1.70	67.08±2.75	71.93±4.17	60.12±6.51	93.01±1.84	91.00±4.25	75.38±5.21	**98.60±0.66**
	8	72.55±4.37	61.01±10.96	64.27±7.56	60.53±6.91	79.55±6.71	60.58±9.58	94.29±1.64	90.18±3.73	79.91±2.13	**98.70±1.44**
	16	83.10±6.86	80.31±5.79	72.58±3.51	73.14±5.29	89.59±2.33	65.12±5.28	93.46±0.72	92.09±2.90	86.97±1.97	**99.29±0.78**
	32	93.09±2.32	86.09±2.54	86.19±4.70	70.12±5.27	96.12±2.31	65.67±3.95	96.05±2.43	96.08±3.53	96.98±1.12	**99.67±0.12**
	64	94.63±1.43	90.07±2.64	90.60±2.76	67.54±3.40	97.74±0.34	68.56±0.72	98.20±2.11	98.06±2.62	99.01±0.26	**99.54±0.48**
Solution -mix	4	72.71±1.96	50.00±0.00	67.70±7.66	68.48±14.97	71.19±3.90	64.52±8.41	73.53±1.94	80.26±5.11	73.75±3.90	**100.00±0.00**
	8	82.91±7.74	58.31±19.17	71.58±10.17	56.67±7.64	82.42±3.72	72.04±10.34	76.34±4.26	83.19±3.70	76.57±4.80	**99.81±0.08**
	16	83.87±1.90	68.09±1.76	76.47±0.56	70.37±13.96	84.39±0.83	71.82±8.79	82.15±5.01	88.10±4.72	84.28±5.15	**99.67±0.13**
	32	91.29±1.43	85.52±8.05	86.74±6.63	76.33±10.57	92.44±3.47	73.08±11.62	82.44±5.43	88.15±2.99	85.39±2.09	**100.00±0.00**
	64	93.18±1.48	89.36±1.88	92.44±1.26	69.78±2.76	95.44±2.63	80.05±2.13	N/A	N/A	92.40±3.06	98.89±1.57

K.2. Ablation Study

Table 18: Ablation study results, showing AUC across 13 datasets.

Data	Shot	FeatLLM	-Tuning	-Ensemble	-Description	-Reasoning
Adult	4	86.68±0.86	87.19±0.45	84.49±1.79	86.54±1.16	85.33±0.75
	8	87.89±0.06	87.34±0.89	84.54±4.93	87.54±0.39	86.13±1.21
	16	87.54±0.50	87.54±0.40	84.75±2.88	87.66±0.56	86.55±0.39
	32	87.09±0.58	87.56±0.27	84.65±2.20	87.71±0.75	85.96±1.10
	64	87.77±0.31	87.77±0.39	84.94±2.00	87.76±0.29	86.73±0.15
Bank	4	70.45±3.69	66.19±1.88	63.88±7.37	69.46±1.54	68.77±3.64
	8	75.85±6.66	68.78±1.07	67.94±7.03	75.30±3.23	70.33±1.55
	16	78.41±1.08	72.73±0.41	68.74±11.83	77.30±0.41	79.93±1.54
	32	78.37±4.50	69.89±5.35	69.08±10.89	78.74±2.97	77.97±6.66
	64	81.18±6.17	72.27±2.09	72.03±8.81	79.88±3.05	79.03±6.70
Blood	4	68.34±7.48	69.31±5.92	61.17±9.30	57.69±6.45	65.42±2.51
	8	70.37±3.23	70.41±3.21	64.76±7.29	68.66±3.99	68.77±4.12
	16	70.07±5.19	69.99±4.66	65.56±8.29	72.01±1.92	68.66±3.18
	32	71.13±4.38	70.57±2.81	67.52±5.37	71.25±2.98	69.50±2.57
	64	71.04±4.36	71.55±4.51	67.24±4.78	70.84±1.35	69.67±2.17
Car	4	72.69±1.52	71.36±1.31	69.89±3.75	69.17±2.84	69.27±0.05
	8	73.26±1.46	72.52±1.55	70.54±4.32	71.76±5.59	72.76±4.32
	16	79.43±1.24	74.64±0.40	74.65±5.46	78.48±1.18	81.82±2.04
	32	85.01±1.36	74.89±0.97	81.20±5.50	85.72±0.18	85.58±2.11
	64	86.78±0.90	74.16±0.27	81.81±6.96	86.55±1.49	87.10±1.49
Communities	4	75.39±5.05	77.42±1.16	66.26±7.07	77.22±3.08	71.97±9.87
	8	76.59±1.25	75.92±1.69	65.89±6.33	76.38±0.93	69.63±6.74
	16	76.25±0.64	74.05±1.97	66.33±6.95	76.42±2.52	75.13±3.61
	32	76.85±1.38	71.64±4.71	66.53±5.82	77.61±1.11	77.39±1.72
	64	78.62±1.39	75.80±0.29	70.50±4.94	78.87±1.81	79.10±2.38
Credit-g	4	55.94±1.10	55.70±1.93	53.56±5.24	53.64±2.03	50.76±0.87
	8	57.42±3.10	53.51±4.26	53.08±7.09	54.13±1.76	49.02±2.68
	16	56.60±2.22	55.48±3.48	54.24±7.03	54.96±8.64	56.99±3.67
	32	61.79±10.25	53.30±3.76	55.51±8.61	62.14±4.61	62.22±2.49
	64	66.43±2.90	57.57±2.88	57.76±6.91	69.09±4.21	63.76±6.43
Diabetes	4	80.28±0.75	78.78±1.21	73.95±9.66	78.88±1.90	77.33±2.44
	8	79.38±1.66	79.81±1.80	75.95±5.59	78.41±2.63	77.09±1.61
	16	80.15±1.35	78.76±0.47	77.44±2.95	79.58±0.12	78.63±1.60
	32	80.06±1.18	78.67±1.93	78.35±2.27	79.56±1.37	78.54±0.86
	64	80.91±1.62	79.48±1.61	79.07±2.31	80.39±1.17	78.90±1.20
Heart	4	75.66±4.59	74.67±4.22	65.91±16.49	68.01±8.43	48.03±8.08
	8	79.46±2.16	69.66±0.74	64.57±19.44	79.06±5.32	78.58±9.07
	16	83.71±1.88	76.57±2.11	71.58±16.06	83.35±4.09	80.39±6.28
	32	87.19±3.66	77.37±4.39	76.61±14.45	86.20±3.27	85.37±3.33
	64	88.08±4.11	81.32±6.06	82.72±9.23	87.69±2.33	87.55±2.04
Myocardial	4	52.87±3.44	54.14±2.89	51.10±6.43	56.12±7.20	49.47±4.93
	8	56.22±1.64	55.42±2.57	52.31±6.14	52.84±4.41	51.42±8.06
	16	55.32±9.15	54.73±7.07	51.73±7.61	56.00±2.51	50.03±13.37
	32	60.02±4.02	55.20±6.38	52.29±7.07	62.20±4.95	46.40±11.98
	64	61.47±3.91	57.19±1.25	53.04±6.26	57.10±6.05	59.60±1.42
Cultivars	4	55.63±5.24	56.58±3.34	54.41±4.33	52.98±2.44	53.73±5.23
	8	56.97±5.08	56.69±4.64	55.29±5.51	54.46±3.86	52.86±3.86
	16	57.19±5.30	55.40±4.71	55.06±6.98	55.37±6.77	56.71±7.72
	32	59.62±7.43	52.96±5.95	53.33±6.10	53.74±7.43	56.15±7.55
	64	59.14±4.79	56.84±4.11	55.74±6.08	54.46±4.79	54.65±4.07
NHANES	4	92.20±1.71	80.73±3.89	84.89±7.12	93.73±2.31	89.85±2.99
	8	93.29±7.01	86.24±6.20	86.94±7.90	92.62±3.53	91.26±6.38
	16	95.64±4.67	88.66±6.17	88.70±9.11	96.63±2.76	95.29±3.48
	32	97.29±1.28	83.97±3.74	92.88±5.72	99.04±0.79	96.89±2.47
	64	98.32±0.65	83.64±3.77	95.21±4.25	97.41±0.58	96.15±1.54
Sequence-type	4	98.60±0.66	94.35±3.98	91.06±7.78	99.00±0.29	97.89±0.54
	8	98.70±1.44	94.32±5.25	91.81±7.20	98.96±0.40	99.14±0.10
	16	99.29±0.78	97.75±2.47	89.45±11.66	99.68±0.28	99.62±0.54
	32	99.67±0.12	93.92±5.13	90.11±8.90	99.33±0.83	99.36±0.57
	64	99.54±0.48	98.33±0.24	94.18±6.92	99.56±0.39	99.79±0.36
Solution-mix	4	100.00±0.00	100.00±0.00	94.08±11.97	99.41±0.51	91.58±6.68
	8	99.81±0.08	99.17±0.13	81.12±14.71	99.52±0.45	92.07±1.52
	16	99.67±0.13	99.50±0.07	84.64±22.27	98.44±1.74	90.06±6.66
	32	100.00±0.00	99.42±0.12	80.07±21.38	97.11±2.51	91.32±7.06
	64	98.89±1.57	98.83±1.65	84.73±19.43	99.48±0.34	93.85±1.18