TreePrompt: Distilling Boosted Tree Ensembles for In-Context Learning in LLMs

Brian Liu
Massachusetts Institute of Technology
briliu@mit.edu

Rahul Mazumder
Massachusetts Institute of Technology
rahulmaz@mit.edu

Abstract

Large Language Models (LLMs) are widely used in natural language processing and other real-world applications due to their ability to generalize across a broad range of tasks. However, they often underperform on tabular prediction problems, where traditional machine learning methods such as gradient boosting remain the state-of-the-art [17]. In this paper we introduce TreePrompt, a framework that aims to bridge this gap. TreePrompt distills a tree ensemble into a concise textual representation and uses the representation for in-context learning in LLMs. This allows an LLM to effectively incorporate structured tabular information, from the tree ensemble, without any expensive model re-fitting or fine-tuning. Across several benchmark datasets, we show that TreePrompt consistently improves LLM performance on tabular prediction tasks and outperforms other in-context learning strategies under a fixed token budget.

1 Introduction

Large Language Models (LLMs) such as GPT-4, PaLM, and Llama have demonstrated exceptional capabilities across a wide range of applications, from question answering and summarization to code generation [1, 2, 19], and have become increasingly central to modern machine learning pipelines across many domains and industries [3]. Despite their versatility and remarkable generalization across natural language tasks, LLMs–and deep learning approaches more broadly–remain limited when reasoning over structured tabular data [17]. For tabular prediction tasks, traditional machine learning methods such as gradient boosted decision trees (e.g. XGBoost, LightGBM, and CatBoost) [6, 10, 16] remain the most competitive and achieve state-of-the-art predictive performance [4].

However, using LLMs for structured tabular data is not without potential. For example, the TabLLM framework demonstrates that when tabular inputs (rows) are carefully serialized into natural-language prompts, LLMs achieve non-trivial zero-shot performance and, with fine-tuning, can match the performance of gradient boosted decision trees [9]. Moreover, one promising direction to improve LLM performance in this domain is through in-context learning (ICL), where the model is provided with a small number of labeled examples directly in the input prompt at inference time—without any retraining or parameter updates [5, 7]. This has the potential to improve predictive performance while avoiding the substantial computational costs associated with fine-tuning.

In this paper, we propose TreePrompt, a novel framework for improving the performance of LLMs on tabular prediction tasks. TreePrompt works by distilling trained gradient boosted tree ensembles, known for their state-of-the-art predictive accuracy on structured data, into sets of compact textual representations that capture the decision logic of the model. These distilled rules are then formatted as in-context examples and provided to an LLM at inference time. This approach allows the LLM to perform accurate predictions on tabular data *without any additional training or fine-tuning*, leveraging the structured decision logic of tree-based models and the flexible reasoning capabilities of language models.

We summarize the contributions of our paper below:

- We propose TreePrompt, a novel framework to distill tree ensembles into textual representations that can be used for in-context learning in language models.
- We demonstrate through our experiments that LLMs augmented with TreePrompt can significantly outperform both zero-shot and more traditional in-context learning (ICL) approaches on tabular classification tasks.

The remainder of the paper is organized as follows. We begin with preliminaries discussing the use of LLMs for tabular prediction tasks along with a review of model distillation techniques for tree ensembles (§2). We then introduce our proposed TreePrompt framework in detail (§3), followed by experimental results comparing the performance of our framework against competing approaches (§4). We conclude by exploring the use of TreePrompt on a real world application (§5).

2 Preliminaries

In this section, we discuss preliminaries on the use of LLMs for tabular prediction as well as methods for distilling tree ensembles into rule-based representations.

2.1 LLMs for Tabular Prediction

TabLLM introduces a framework for applying large language models (LLMs) to tabular classification tasks in both zero-shot (without re-training) and few-shot settings [9]. The framework explores multiple strategies for serializing tabular data, ranging from manually designed list-based and text-based templates to more sophisticated methods that leverage LLMs themselves to generate serialized representations. Each observation (i.e., row) is serialized into a natural language string and combined with a task-specific prompt—e.g., "Answer this question Yes or No"—to frame the classification task. The model's output probabilities over specific verbalizer tokens (e.g., "Yes", "No") are then mapped to class probabilities to make a final prediction. One surprising finding of TabLLM is that manually crafted text-template serializations, which enumerate all features in the form "The feature name is value," consistently outperform more complex alternatives. Using this serialization scheme, TabLLM exhibits non-trivial predictive accuracy in the zero-shot setting, which indicates that the framework utilizes knowledge encoded into the LLM. With fine-tuning in the few-shot setting, TabLLM can match the performance of boosted tree ensembles, however, the fine-tuning process is computationally expensive and requires GPUs [13].

In-context learning (ICL), where prompts are augmented with task instructions or labeled examples, has also been shown to enhance the predictive accuracy of LLMs without fine-tuning [18, 20]. However, the effectiveness of ICL is often constrained by the limited context windows of LLMs, which restrict the number of examples that can be provided during inference [20]. In TreePrompt, we address this limitation by distilling a compact set of rules from a trained tree ensemble. These rules serve as informative in-context demonstrations that significantly improve the predictive performance of the LLM on tabular tasks, compared to providing traditional in-context labeled examples. We discuss distilling tree ensembles in the section below.

2.2 Distilling Tree Ensembles

Boosted tree ensembles consist of many decision trees that are fit sequentially on the residuals of the previous model to improve predictive performance. Each individual decision tree within the ensemble can be succinctly summarized as a collection of logical rules, obtained by traversing the paths from the root to leaf nodes, that map feature conditions to predictions.

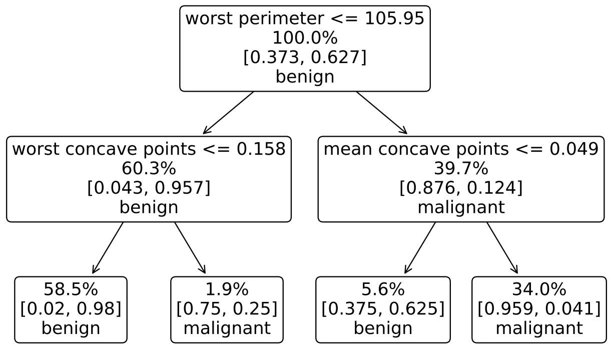


Figure 1: Single decision tree of depth 2 fit on the Wisconsin Breast Cancer Dataset. This decision tree yields four decision rules, obtained by traversing the tree from the root to leaves.

We demonstrate this in Figure 1, where we examine a single decision tree of depth 2 fit on the Wisconsin Breast Cancer Dataset from the UCI Machine Learning Repository [8]. This tree yields 4 decision rules, each corresponding to a unique path from the root to a leaf node. These rules can be expressed by the following textual representations:

- **Rule 1:** If the worst perimeter of the mass is less than 105.95 and the worst concave points is less than 0.158, predict that the mass is benign.
- **Rule 2:** If the worst perimeter is less than 105.95 and the worst concave points is greater than 0.158, predict that the mass is malignant.
- **Rule 3:** If the worst perimeter is greater than 105.95 and the mean concave points is less than 0.049, predict that the mass is benign.

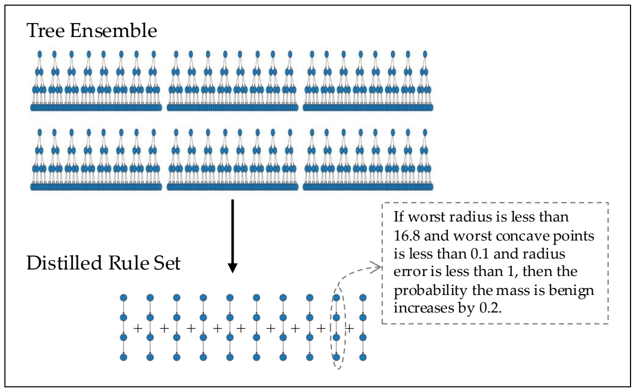

Tree Ensemble
Distilled Rule Set
If worst radius is less than 16.8 and worst concave points is less than 0.1 and radius error is less than 1, then the probability the mass is benign increases by 0.2.

Figure 2: Distilling sparse rule sets from a tree ensemble yields a compact textual representation of the model.

- **Rule 4:** If the worst perimeter is greater than 105.95 and the mean concave points is greater than 0.049, predict that the mass is malignant.

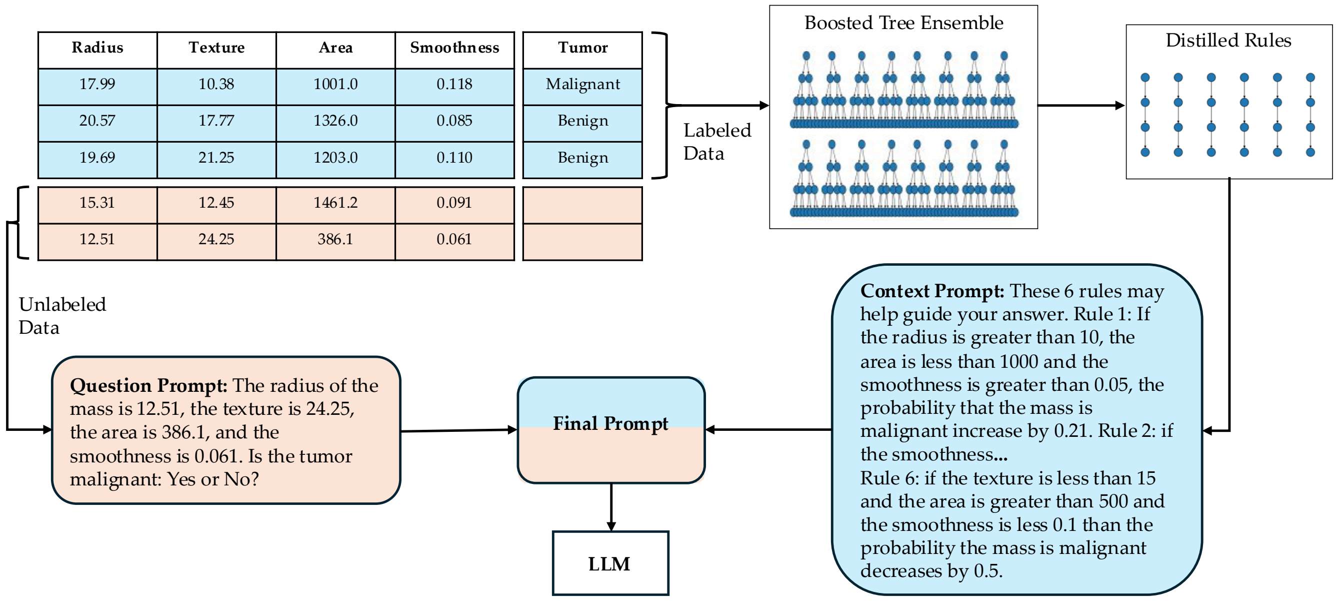
Tree ensembles can be represented by similar sets of rules, however, their textual representations can grow extremely large. Consider a medium size boosting ensemble of 1000 depth 2 decision trees, which contains 4000 decision rules. The textual representation of this ensemble contains 200000 tokens, which far exceeds the token limit of even large models such as GPT-4o [1].

To address this, we can distill tree ensembles by extracting sets of rules that 1) perform well in terms of predictive accuracy and 2) are compact enough to have concise textual representations. We do so using this general optimization-based framework, where $w \in \mathbb{R}^m$ is a vector of decision variables assigned to each potential rule (node) in the original tree ensemble and $r_j(\mathbf{x})$, $\forall\ j \in [m]$, represent the predictions of the rules:

$$\min_{w}\ L\Big(\mathbf{y},\ \sum_{j=1}^{m} w_j r_j(\mathbf{x})\Big) \quad \text{s.t. } ||w||_0 \le k. \tag{1}$$

The constraint $||w||_0 \le k$ controls the number of rules extracted and the objective captures data fidelity, i.e., how well the predictions of the extracted rule set fits response vector $\mathbf{y}$. This general optimization framework has been used by many methods such as FIRE [11] and ForestPrune [12] to distill tree ensembles into compact representations that account for rule sparsity and interaction depth.

In Figure 2, we illustrate our distillation approach using the Wisconsin Breast Cancer Dataset. We first fit a boosting ensemble of 500 depth 3 decision trees on the dataset. A subset of these trees is shown in the top of Figure 2. We use the optimization algorithm presented in FIRE [11] to find a good solution for Problem (1) in order to extract a compact set of 10 decision rules. These 10 rules achieve predictive accuracy comparable to the full ensemble. We can express each rule in a textual format, Figure 2 shows an example of this, and the resulting rule set is concise enough to be used as contextual examples for prompting LLMs. We formalize our TreePrompt framework in the section below.



Radius
Texture
Area
Smoothness
Tumor
17.99
10.38
1001.0
0.118
Malignant
20.57
17.77
1326.0
0.085
Benign
19.69
21.25
1203.0
0.110
Benign
15.31
12.45
1461.2
0.091
12.51
24.25
386.1
0.061
Labeled Data
Boosted Tree Ensemble
Distilled Rules
Unlabeled Data
Question Prompt: The radius of the mass is 12.51, the texture is 24.25, the area is 386.1, and the smoothness is 0.061. Is the tumor malignant: Yes or No?
Final Prompt
LLM
Context Prompt: These 6 rules may help guide your answer. Rule 1: If the radius is greater than 10, the area is less than 1000 and the smoothness is greater than 0.05, the probability that the mass is malignant increase by 0.21. Rule 2: if the smoothness...
Rule 6: if the texture is less than 15 and the area is greater than 500 and the smoothness is less 0.1 than the probability the mass is malignant decreases by 0.5.

Figure 3: A visualization of our TreePrompt framework.

3 TreePrompt Framework

We present our TreePrompt framework for binary classification tasks on tabular datasets. Figure 3 shows a visualization of our framework.

3.1 Formulation

We start with a tabular dataset of n observations and p features which, following the notation from TabLLM [9], we represent as $\mathcal{D} = \{(\mathbf{x_i}, y_i)\}_{i=1}^n$. Vector $\mathbf{x_i} \in \mathbb{R}^p$ is the feature vector and y_i gives the class label (binary) for observation i. We assume access to a labeled subset $\mathcal{D}_l \subset \mathcal{D}$ and an unlabeled subset $\mathcal{D}_u \subset \mathcal{D}$. The goal of TreePrompt is to use a large language model with in-context learning to generate accurate predictions for the unlabeled instances in $\mathcal{D}_u$.

3.2 Labeled Data: Generating Contextual Rules

On the labeled subset of the data, $\mathcal{D}_l$, we first fit a boosted tree ensemble such as XGBoost, LightGBM, or a gradient boosting machine [14], to predict class labels y_i using feature vectors $\mathbf{x_i}$, for all $(\mathbf{x_i}, y_i) \in \mathcal{D}_l$. Let $r_1(\mathbf{x}), \dots, r_m(\mathbf{x})$ represent the set of possible rules to extract from the ensemble and let vector $\boldsymbol{\rho} \in [0,1]^{|\mathcal{D}_l|}$ represent the predicted probabilities of the tree ensemble. We follow the general framework presented in display (1) to extract a sparse subset of k rules. Consider the optimization problem:

$$\min_{w} \frac{1}{2}\left\|\boldsymbol{\rho} - \sum_{j=1}^{m} w_j r_j(\mathbf{x})\right\|_2^2 + \lambda\|w\|_0, \tag{2}$$

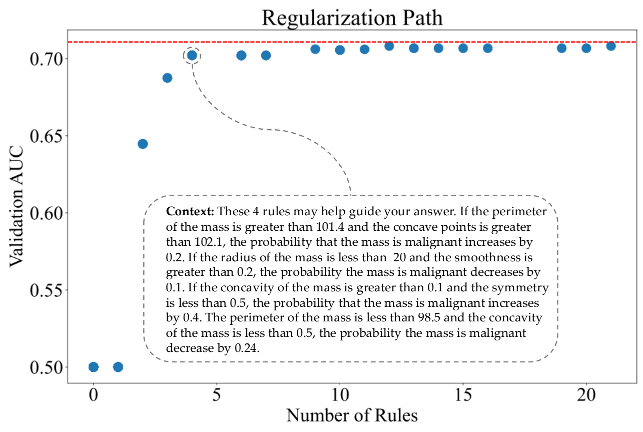
which is a penalized formulation of (1). We use parameter λ to control the sparsity of the extracted rules to be less than k. In this approach, we also directly fit the extracted rules to approximate ρ, the predicted probabilities of the original tree ensemble. As such, the output of the extracted rules, $w_j r_j(\mathbf{x})$ where $w_j \neq 0$, can be interpreted as an increase or decrease in the probability that the predicted class is positive. To obtain high-quality solutions for Problem (2) we apply an iterative coordinate descent-based algorithm similar to the one used in [11].

After we use Problem (2) to extract k rules from our ensemble, we serialize the rules into natural language text, using the following template-based procedure. For each rule, textually enumerate each condition, "*If the value of feature 1 is greater than threshold 1 and if the value of feature 2 is less than threshold 2, then...*" and if the rule is satisfied state that the "*probability that the class is positive increases*" (or decreases) by output $w_j r_j(\mathbf{x})$.

After serializing all k rules, we combine the serializations and append the sentence "These k rules may help guide your answer." to the prompt. The resulting **context prompt** is used for ICL, allowing the language model to incorporate the extracted rules into its prediction. We show an example of this procedure for the labeled observations, highlighted in blue, in Figure 3.

3.3 Unlabeled Data

On the unlabeled subset of the data, $\mathcal{D}_u$, we serialize each feature vector $\mathbf{x_i} \in \mathcal{D}_u$ into natural language text. We use the text template procedure discussed in [9] to serialize each unlabeled observation by textually enumerating its feature values. We append the sentence "*Is the class positive: Yes or No? Answer:*" to the serialization of each observation to generate our **question prompt**. Finally, for each observation in the unlabeled subset of the data we combine the context prompt and the question prompt to into the final prompt to input into the LLM.



Context: These 4 rules may help guide your answer. If the perimeter of the mass is greater than 101.4 and the concave points is greater than 102.1, the probability that the mass is malignant increases by 0.2. If the radius of the mass is less than 20 and the smoothness is greater than 0.2, the probability the mass is malignant decreases by 0.1. If the concavity of the mass is greater than 0.1 and the symmetry is less than 0.5, the probability that the mass is malignant increases by 0.4. The perimeter of the mass is less than 98.5 and the concavity of the mass is less than 0.5, the probability the mass is malignant decrease by 0.24.

Figure 4: Example regularization path to evaluate the trade-off between size and performance of the distilled model. The performance of the original boosted tree ensemble is indicated in red.

3.4 How many contextual rules to include?

An important parameter in our TreePrompt framework is the number of rules k included in the context prompt. In general, increasing k improves the predictive accuracy of the distilled model by better approximating the full ensemble. However, including too many rules can increase inference costs and may cause the prompt to exceed the context window of the LLM.

We use the following procedure to determine good values for k. We further split the labeled subset of the data $\mathcal{D}_l$ into a training set and validation set. We extract rule sets of varying sizes, by sweeping through values of λ in (2), and evaluate the validation performance of the models. This yields a regularization path, or a sequence of rule sets of varying sizes, like the one shown in Figure 4. In this figure, the horizontal axis shows the number of rules and the vertical axis shows the validation AUC[1] of the extracted rule sets. The performance of the full tree ensemble is indicated by the red line, and we also show a sample context prompt for $k = 4$ rules. This regularization path evaluates the trade-off between model size and predictive performance for the distilled rule set. In practice, we can select k to be in the "elbow" of the path, for example $k = 4$ in Figure 4 to balance model size and performance. Empirically, we observe that values of k between 10 to 15 work well for ICL.

4 Experiments

In this section, we present our experimental evaluation of TreePrompt against several competing methods.

4.1 Setup

We evaluate TreePrompt on 7 datasets from the UCI Machine Learning Repository [8]: BREAST_W, ILPD, BLOOD, PHONEME, DIABETES, CLIMATE, KC2. On each dataset, we randomly split the observations evenly into a training set and a test set, which correspond to $\mathcal{D}_l$ and $\mathcal{D}_u$ respectively, and apply our TreePrompt framework. On the training dataset, we fit a scikit-learn gradient boosting ensemble [14] of 500 depth 3 decision trees and use Problem (2) to distill the model into a compact set of decision rules. We use the elbow method discussed in §4 to select k. We serialize the distilled rules to construct the context prompt, and, on the test dataset, we serialize each observation to construct the question prompts. We combine the context prompt with the question prompts to obtain the final prompt to input into the LLM; for experiment, we use GPT-4o Mini [1]. We use the log-probability of the output verbalizer token as the predicted probability of the positive class and compute the test AUC of TreePrompt based on these predicted probabilities. For each dataset, we repeat this procedure of 5 random train-test splits.

We compare our approach against the following competing algorithms. Neither these baselines nor our TreePrompt framework require retraining or fine-tuning of the LLM.

- **LLM-baseline:** For this baseline algorithm, we input the question prompts directly into the LLM, without context prompts. This evaluates the zero-shot performance of the LLM [9].
- **DT-Context:** We fit a single depth 3 decision tree on the training data and use the textual representation of the tree as our context prompt. This a new competing algorithm that we propose for this experiment.
- **TabularICL:** We randomly select observations in the test set and serialize them, with their corresponding class labels. We use these examples for ICL [5] in the context prompt.

Importantly, for a fair comparison, we constrain TreePrompt, DT-Context, and TabularICL to construct context prompts with the same token budget.

4.2 Results

Data	Tree Prompt	LLM baseline	DT Context	Tabular ICL	GBDT
BREAST_W	0.99 (0.003)	0.96 (0.007)	0.88 (0.006)	0.95 (0.002)	0.98 (0.005)
ILPD	0.74 (0.017)	0.70 (0.019)	0.72 (0.021)	0.71 (0.018)	0.72 (0.01)
BLOOD	0.74 (0.004)	0.55 (0.011)	0.68 (0.005)	0.61 (0.012)	0.75 (0.001)
PHONEME	0.74 (0.012)	0.56 (0.009)	0.63 (0.02)	0.62 (0.012)	0.82 (0.01)
DIABETES	0.82 (0.0005)	0.76 (0.0001)	0.78 (0.007)	0.79 (0.0007)	0.84 (0.003)
CLIMATE	0.79 (0.0005)	0.53 (0.019)	0.59 (0.041)	0.55 (0.02)	0.81 (0.004)
KC2	0.85 (0.009)	0.82 (0.002)	0.74 (0.03)	0.83 (0.005)	0.82 (0.001)
Avg. Rank	**1.00**	**3.71**	**2.71**	**2.71**	—

Table 1: Experiment results in terms of test AUC. TreePrompt outperforms all competing algorithms and can outperform boosting on several of the datasets considered.

We present the results of our experiment in Table 4, which reports the average test AUC achieved by each method across all

[1]Recall that we are fitting (2) on the predicted probabilities of the original tree ensemble

datasets. Values in parentheses indicate standard errors. The rightmost column shows the average test AUC of the boosted tree ensemble, which serves as a strong point of reference given that boosted trees are specifically designed for predictive tasks on tabular data.

Across all datasets, we observe that TreePrompt outperforms our LLM-baseline, DT-Context, and Tabular-ICL competing methods. The average rank of each method, obtained over all datasets, is shown in the bottom row of the table. We also note that the performance of TreePrompt matches or exceeds to performance of our GBDT boosting ensemble in several cases. Among our competing algorithms, we observe that, consistent with findings from Hegselmann et al. [9], the zero-shot LLM-baseline achieves non-trivial and, on some datasets, surprisingly strong predictive performance. Our competing in-context learning approaches such as TabularICL, which includes labeled examples in the prompt, and DT-Context, which uses a decision tree trained on the labeled data, further improve performance over the baseline.

Our approach, however, achieves the best overall results. We emphasize that TreePrompt, does not require fine-tuning or retraining of the LLM, which is often computationally prohibitive. Instead, TreePrompt leverages structured label data to construct a context prompt by distilling a tree ensemble trained on the dataset. Our experimental results suggest that, under a fixed token budget, the context prompts generated by TreePrompt contain more relevant information than those produced by competing ICL methods, leading to improved LLM performance on tabular prediction tasks.

5 Conclusion

We conclude with a case study demonstrating the use of TreePrompt on a real-world multimodal dataset: the Liar benchmark for fake news detection. This dataset comprises thousands of short political statements, each labeled for truthfulness by PolitiFact [15]. The available labels include *true*, *mostly-true*, *half-true*, *barely-true*, *false*, and *pants-fire*. For our case study, we focus on the borderline categories, *barely-true* and *mostly-true*, and train a classifier to distinguish between them.

Each observation in the Liar dataset includes the text of the statement and the name of the speaker, along with structured tabular features such as the speaker's political affiliation, geographic location, and credibility history, i.e., counts of previous statements rated as truthful or untruthful. We split the dataset evenly into training and test sets. On the test set, we apply a baseline LLM approach that combines the statement text with a serialized representation of the tabular features to create question prompts. Using these question prompts, we ask the model to classify the statement as either *mostly-true* (positive class) or *barely-true* (negative class). This approach achieves a test AUC of 0.72.

We then apply TreePrompt, fitting a gradient boosting tree ensemble on the structured tabular features from the training set. Following the procedure outlined in §4.1, we distill the ensemble into 11 interpretable rules and serialize them to construct the context prompt. When this context prompt is combined with the question prompts used above, the LLM's test AUC increases to 0.81. Notably, the distilled rules capture aspects of the speaker's credibility history, information not explicitly present in the statement text. By integrating structured features with natural language input, TreePrompt can significantly improve the performance of LLMs on tabular prediction tasks.

References

[1] Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774* (2023).

[2] Rohan et al. Anil. 2023. PaLM 2 Technical Report. *arXiv preprint arXiv:2305.10403* (2023).

[3] Rishi Bommasani, Drew A Hudson, Ehsan Adeli, Russ Altman, Simran Arora, Sydney von Arx, Michael S Bernstein, Jeannette Bohg, Antoine Bosselut, Emma Brunskill, et al. 2021. On the opportunities and risks of foundation models. *arXiv preprint arXiv:2108.07258* (2021).

[4] Vyacheslav Borisov, Thomas Leemann, Patrick Seegerer, Yao Wang, Frank Hutter, and Iryna Gurevych. 2022. Deep neural networks and tabular data: A survey. *arXiv preprint arXiv:2110.01889* (2022).

[5] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language Models are Few-Shot Learners. *Advances in Neural Information Processing Systems* 33 (2020), 1877–1901.

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

[7] Qingxiu Dong, Lei Li, Damai Dai, Ce Zheng, Jingyuan Ma, Rui Li, Heming Xia, Jingjing Xu, Zhiyong Wu, Tianyu Liu, et al. 2022. A survey on in-context learning. *arXiv preprint arXiv:2301.00234* (2022).

[8] Dheeru Dua and Casey Graff. 2019. UCI Machine Learning Repository. (2019). http://archive.ics.uci.edu/ml

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

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

[11] Brian Liu and Rahul Mazumder. 2023. Fire: An optimization approach for fast interpretable rule extraction. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 1396–1405.

[12] Brian Liu and Rahul Mazumder. 2023. Forestprune: Compact depth-pruned tree ensembles. In *International Conference on Artificial Intelligence and Statistics*. PMLR, 9417–9428.

[13] Haokun Liu, Derek Tam, Mohammed Muqeeth, Jay Mohta, Tenghao Huang, Mohit Bansal, and Colin A Raffel. 2022. Few-shot parameter-efficient fine-tuning is better and cheaper than in-context learning. *Advances in Neural Information Processing Systems* 35 (2022), 1950–1965.

[14] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Édouard Duchesnay. 2011. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research* 12 (2011), 2825–2830.

[15] PolitiFact. 2007. PolitiFact. https://www.politifact.com/ Accessed: 2025-05-27.

[16] Liudmila Prokhorenkova, Gleb Gusev, Aleksandr Vorobev, Anna Dorogush, and Andrey Gulin. 2018. CatBoost: unbiased boosting with categorical features. In *Advances in neural information processing systems*, Vol. 31.

[17] Ravid Shwartz-Ziv and Aviram Armon. 2022. Tabular data: Deep learning is not all you need. *Information Fusion* 81 (2022), 84–90.

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

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

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