Zero-Shot Decision Tree Construction via Large Language Models

Anonymous ACL submission

Abstract

We present a zero-shot algorithm for building decision trees with large language models (LLMs) based on CART principles. Unlike traditional methods, which require labeled data, our approach uses the pretrained knowledge of LLMs to perform key operations such as feature discretization, probability estimation, and Gini-based split selection without training data. We also introduce a few-shot calibration step that refines the zero-shot tree with a small set of labeled examples. The resulting trees are interpretable, achieve competitive performance on tabular datasets, and outperform existing zero-shot baselines while approaching supervised models in low-data regimes. Our method provides a transparent, knowledge-driven alternative for decision tree induction in settings with limited data.

1 Introduction

Tabular data is central to decision-making in fields like healthcare, finance, and climate science. While traditional models—such as logistic regression, gradient-boosted trees, and neural networks—perform well in these settings (Grinsztajn et al., 2022), they require large labeled datasets, which are often scarce or costly. In high-stakes domains, interpretability is also critical—a strength of decision trees due to their transparent, rule-based structure (Rudin, 2019; Lipton, 2018). This paper introduces a zero-shot framework for inducing decision trees using LLMs, relying solely on feature and target descriptions. Our method produces interpretable trees that generalize well in low-data settings by leveraging the pretrained knowledge of the LLM.

Decision trees are the most widely used machine learning models, valued for their interpretability, simplicity, and effectiveness in classification tasks (Grinsztajn et al., 2022). Traditional algorithms for constructing decision trees, such as CART and C4.5, require access to labeled datasets to recursively partition the feature space using criteria like information gain or the Gini index (Quinlan, 2014). However, these approaches face limitations when data is scarce or entirely unavailable, prompting the need for alternative methodologies.

LLMs have demonstrated a remarkable ability to perform complex tasks in diverse domains by leveraging knowledge encoded during pretraining (Brown et al., 2020). LLMs excel in zero-shot and few-shot learning paradigms, where minimal or no labeled examples are provided (Raffel et al., 2020; Sanh et al., 2022), making them ideal candidates for addressing data-scarce challenges. This paper introduces a zero-shot method for building decision trees with an LLM.

Our method leverages the contextual reasoning capabilities of LLMs to carry out the core operations required for decision tree construction. Specifically, the LLM is prompted to discretize continuous attributes and estimate conditional class probabilities—key steps in evaluating potential splits. Using this information, we recursively build the tree by selecting splits that minimize the Gini index. Our approach shows that LLMs can emulate the logic of traditional decision tree algorithms without access to labeled training data.

Our contributions are summarized as follows:

- **Zero-Shot Tree Induction.** We introduce an algorithm for constructing decision trees in a zero-shot and few-shot setting.
- **Effectiveness in Data-Scarce Environments.** We show that our LLM-based approach performs competitively in low-data regimes.
- **Few-Shot Calibration.** We introduce a few-shot refinement procedure that adjusts the tree using a small number of labeled examples.

2 Related Work

2.1 Classification with LLMs

Transformer models (Vaswani, 2017), which form the backbone of most large language models, are pre-trained on vast amounts of text data and demonstrate remarkable versatility in generalizing to new tasks with minimal or no labeled examples. The architecture's ability to effectively leverage prior knowledge encoded within its parameters makes LLMs particularly attractive for few-shot learning scenarios (Brown et al., 2020; Sanh et al., 2022).

Zero-Shot Classification Zero-shot learning has emerged as a promising approach for scenarios where labeled data is scarce or unavailable. This method leverages pre-trained models to predict new, unseen tasks without task-specific training data. Recent studies have shown the potential of zero-shot learning in various domains. Hegselmann et al. (Hegselmann et al., 2023) demonstrated that prompting LLMs with zero-shot or few-shot examples yields performance comparable to fine-tuned models in specific classification tasks. Similarly, pre-training on large tabular datasets further enhances an LLM's ability to generalize to unseen data, as shown by work in supervised tabular learning (Wen et al., 2024), unified table representation (Yang et al., 2024), and tabular Transformers (Wang and Sun, 2022). Recent work has demonstrated that LLMs can generate decision trees without data (Knauer et al., 2024). Our approach is closest to this line but differs in that we construct the tree *iteratively*, inspired by the CART algorithm, prompting the model step by step for splits and conditional probabilities.

Supervised Learning Deep learning for tabular data has gained attention recently, resulting in new transformer-based architectures. (Yin et al., 2020) introduced *TabERT*, a self-supervised model that improves learning from structured datasets through goals such as masked cell prediction and contrastive losses. Building on this, (Arik and Pfister, 2021) proposed *TabNet*, an attention-based framework specifically designed to capture sparse and meaningful feature interactions in tabular data. Similarly, (Somepalli et al., 2021; Chen et al., 2023) developed *SAINT* and *ExcelFormer*, respectively, both of which propose novel self-attention mechanisms to model interactions in tabular datasets. *MetaTree* (Zhuang et al., 2024) uses transformers and meta-learning to construct decision trees, mirroring classical greedy algorithms recursively. Other approaches, such as (Hollmann et al., 2022), introduced *TabPFN*, a Bayesian neural network pre-trained on synthetic tabular datasets, demonstrating strong generalization capabilities across diverse tabular tasks.

Fine-Tuning Another approach for classification is fine-tuning an LLM with a serialized representation of the tabular data and a description of the classification problem (Dinh et al., 2022; Hegselmann et al., 2023). TapTap employs pre-training on a large corpus of tabular data and can generate high-quality synthetic tables to improve the performance of prediction models (Zhang et al., 2023a). To achieve this, TableGPT2 (Su et al., 2024) creates a dataset that includes various tabular datasets and tasks, such as classifications, for fine-tuning LLMs. TableLlama (Zhang et al., 2023b) shows the potential of fine-tuning with multimodal models to address the challenges of table-based tasks.

Interpretable Classification Despite advances in classification with machine learning methods, gradient-boosted tree ensembles remain the dominant choice for many practical applications due to their robustness, interpretability, and consistent performance across various tabular datasets. Classic methods such as XGBoost (Chen and Guestrin, 2016) and LightGBM (Ke et al., 2017) continue to set the standard, often outperforming deep learning models in real-world scenarios (Shwartz-Ziv and Armon, 2022; Grinsztajn et al., 2022; Borisov et al., 2022; McElfresh et al., 2024). Studies have highlighted the unique challenges deep learning models face with tabular data, including feature representation, small sample sizes, and overfitting, further reinforcing the advantages of traditional ensemble methods (Borisov et al., 2022). This persistent superiority emphasizes the need for hybrid approaches that combine the strengths of deep learning with the efficiency and interpretability of tree-based models (McElfresh et al., 2024).

3 Methodology

We propose a zero-shot, recursive algorithm for constructing binary decision trees using a LLM. The process is composed of four core steps: (1) proposing feature splits, (2) estimating class probabilities, (3) selecting optimal splits using the Gini index, and (4) few-shot calibration. The tree is built recursively without supervised training data,

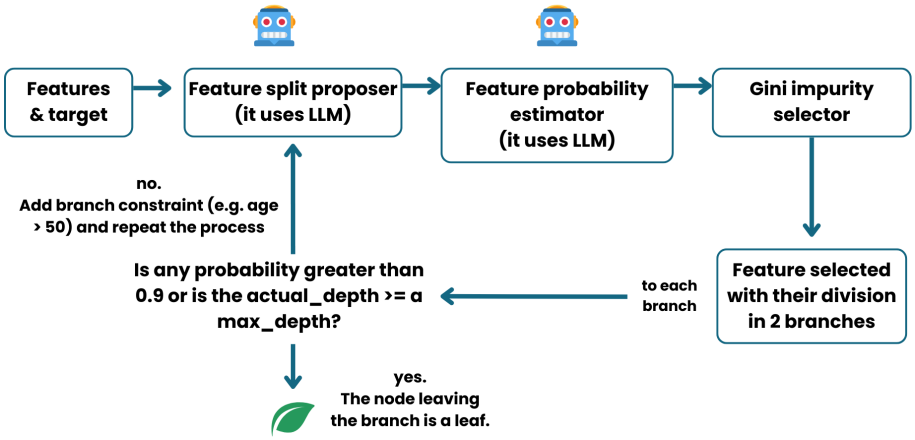

Figure 1: Overview of Zero-Shot Decision Tree Construction with LLMs. Given only feature descriptions, the model selects split nodes and estimates class probabilities. The constraint imposed by the branch is added to a list of constraints, and it iterates again, thus going down one level in the tree.

relying solely on the names and descriptions of the input features and target variable. An optional few-shot calibration step allows the model to refine node probabilities using limited supervision. Figure 1 illustrates the full process.

This recursive framework produces a hierarchical structure with decision nodes representing attribute splits and leaf nodes corresponding to high-confidence prediction probabilities.

3.1 Overview of the Process

The algorithm takes as input the names, types, and descriptions of the features and target variable. For example, when predicting whether a patient has diabetes based on attributes like age, BMI, and glucose, the LLM leverages its pretrained knowledge to drive the construction process.

First, the **Feature Split Proposer** suggests candidate splits (e.g., "age > 50") based on the semantics of the features. Next, the **Feature Probability Estimator** evaluates each split by estimating class probabilities for the resulting branches (e.g., 85% probability of diabetes for "glucose > 140").

The **Gini Impurity Selector** scores these splits using the Gini index and selects the one that maximizes label homogeneity. The selected split partitions the input space, and the algorithm recursively proceeds on each resulting branch. This continues until a stopping condition is met: a confidence threshold $\tau = 0.9$ or a maximum tree depth.

Although the core tree is constructed without labeled data, we optionally apply a **few-shot calibration** step to refine class probabilities using a small labeled set. This improves predictive performance without altering the structure of the tree.

3.2 Attribute Splitting

The first step to build the decision tree is to generate splits for each feature based on the target variable (e.g., probability of diabetes). To do this, the LLM is provided with the feature names, their types (numerical or categorical, along with possible categories), and any contextual constraints or prior probabilities from the current subtree. Using this information, the LLM proposes semantically valid and context-aware partitions, such as "glucose > 140" or "age > 50", that aim to divide the input space in a way that reflects the problem context.

All this information is passed to the language model via prompting, asking it to deliver a value or groupings that allow the data to be divided into 2 groups. The detailed prompts can be found in Appendix Section D.1.1.

3.2.1 Numerical Attributes

For numerical features, the LLM proposes a threshold to split the data into two groups: `values` $\leq$ `threshold` and `values` $>$ `threshold`. These thresholds are guided by the model's contextual knowledge and aim to improve label separation. For instance, for the feature "age," the LLM might suggest a split at 35. The feature's data type (e.g., float or integer) is also provided for validation. A typical prompt is:

> *Your role is to propose a single numeric value to divide the data into two groups based on the attribute* `{feature}`...

3.2.2 Categorical Attributes

For categorical features, the LLM proposes splits by dividing the categories into two disjoint, semantically meaningful groups that aim to separate target labels effectively. For example, given the feature "color" with categories $\{red, blue, green, yellow\}$, it might return $\{red, blue\}$ vs. $\{green, yellow\}$. A typical prompt is:

> *Your task is to propose two disjoint groups of categories based on the categorical attribute* `{feature}` *with possible categories:* `{possible values}`...

3.3 Probability Estimation

For each proposed split, the LLM estimates the conditional probabilities of the target labels for the resulting branches (e.g., "`values` $\leq$ `threshold`" vs. "$>$ `threshold`"). These estimates incorporate the split condition, feature descriptions, and prior probabilities from the parent node. By considering the context of the current subtree, the LLM provides class distributions that guide the selection of the most informative split. A typical prompt is:

> *Estimate the probabilities for each target class based on the previous node and the new split. Avoid overconfident predictions.*

The LLM uses its pretrained knowledge to infer label distributions without access to data, relying on patterns and contextual cues (Brown et al., 2020). To prevent overestimation, which is common in early splits, we refined the prompts to instruct the LLM to avoid overconfidence and ensure more balanced probability estimates. These estimated probabilities are then used to evaluate split quality via Gini impurity.

3.4 Gini Impurity Minimization

To guide tree construction, each candidate split is evaluated using Gini impurity, which quantifies label homogeneity within the resulting branches. The split with the lowest impurity, indicating the best class separation, is selected. Gini impurity is defined as:

$$\text{Gini} = 1 - \sum_{i=1}^{C} p_i^2,$$

where p_i is the predicted probability of the i-th class, and C is the number of target labels.

Since our approach lacks access to instance-level data, we aggregate the impurities of the two branches using the harmonic mean:

$$\text{Score} = \frac{2 \cdot \text{Gini}_1 \cdot \text{Gini}_2}{\text{Gini}_1 + \text{Gini}_2}.$$

Unlike CART (Breiman, 2017), which uses a weighted average based on instance counts, the harmonic mean favors balanced splits by penalizing uneven impurity distributions. This encourages the selection of splits that are effective for both branches, improving overall tree quality and promoting label purity across the structure.

Branch Creation. Once the optimal split is selected, the input context is partitioned into two branches corresponding to the left and right child nodes. Each branch inherits the current constraints and incorporates the new condition imposed by the split (e.g., "glucose > 140"). These updated contexts are treated as independent subproblems, and the decision tree construction procedure is recursively applied to each.

Constraint Propagation. As the tree expands, feature constraints are dynamically maintained to ensure valid and non-redundant splits. For numerical features, the upper or lower bounds are adjusted based on previous split conditions, effectively narrowing the feature's valid range. For categorical features, selected categories are removed from consideration; if only one category remains, the feature is excluded from future splits. This mechanism ensures that subsequent decisions are consistent with earlier ones and helps prevent logically invalid or redundant partitions.

Stopping Criteria. Recursion continues until a stopping condition is met: either the maximum tree depth is reached, or the predicted probability of a single class exceeds a confidence threshold (e.g., ≥ 0.9). At that point, a leaf node is created, storing the most probable class label and the final class probabilities.

3.4.1 Few-Shot Tree Calibration

Once the zero-shot tree has been constructed, we apply a calibration step to refine the class probabilities at the leaf nodes using a small labeled set. This post hoc adjustment improves predictive accuracy without altering the tree structure. For each labeled example, the tree is traversed to the corresponding leaf, and the probability assigned to the true label is

incrementally updated. Let LP denote the current probability of the correct class; it is updated using the following rule:

$$LP \leftarrow LP + p(1 - LP), \tag{1}$$

where p is a learning rate that controls the strength of the adjustment. This formula increases the true label's probability by a fraction p of the remaining mass, gradually reinforcing correct predictions as more examples are processed.

3.5 Prompt Design

The prompts used for feature splitting and probability estimation required careful refinement to ensure valid and balanced outputs. Early challenges included overlapping groupings, biased probabilities, and invalid splits. To address these issues, prompts were iteratively adjusted with explicit constraints and examples. See Appendix D for prompt details.

4 Experimental Design and Evaluation

To evaluate the proposed methodology for zero-shot decision trees, we conducted a comparative analysis with two baseline approaches: (1) **TabLLM** (Hegselmann et al., 2023), a state-of-the-art method for tabular data classification using LLMs, and (2) traditional **supervised Decision Trees**. These methods were selected as benchmarks to assess the strengths and limitations of our approach, focusing on ROC-AUC and accuracy scores across diverse classification tasks.

For the traditional Decision Tree models, we varied the `max_depth` parameter (set to 7, 5, and 3) to analyze its impact on predictive accuracy and interpretability. For the zero-shot methods, we test two LLMs: `GPT-4o-mini` (closed-weight) and `Llama 4-70B` (open-source).

To assess performance under different data regimes, we trained the supervised models using subsets of $0, 4, 8, 16, 32, 64, 128, 256$ examples, and the full training set. For each shot size k, we extracted multiple independent support sets to obtain statistically robust metrics: 50 random samples for those of the classical decision tree, 10 for the LLM tree (due to the higher inference cost), and 4 for TabLLM (the more expensive variant). Each model was trained/evaluated at each repetition, and the ROC-AUC and accuracy scores shown correspond to the mean of those repetitions. This design enables a direct comparison of our method's generalization ability and data efficiency in both low- and high-resource scenarios. Additionally, for our method, the same labeled subsets were used to apply the **few-shot calibration** step.

Given that our approach is designed for low-data settings, we selected datasets from diverse domains—health, finance, environment, and politics to evaluate its adaptability across real-world scenarios. Notably, we included two recent political datasets:*Presidential Approval* and *Judgment on Abortion* from September 2024, which fall beyond the training cutoffs of GPT-4 (October 2023) (OpenAI, 2023) and Llama 4 Maverick (August 2024). This offered a unique opportunity to test the robustness and generalization of our zero-shot method on truly out-of-sample data.

Each dataset was split into training and testing sets to compare across methods. Training data was used to train supervised models and apply few-shot calibration, while test sets were used to evaluate all methods, including zero-shot.

4.1 Baseline Methods

TabLLM zero-shot This baseline leverages the ability of LLMs to perform zero-shot classification directly on tabular data. TabLLM is designed to encode tabular data into text prompts, referred to as serialization, and combine it with a concise description of the classification problem. This enables the model to perform classification without training, relying solely on its prior knowledge.

Traditional Decision Trees Decision trees were trained using `Scikit-learn` (Pedregosa et al., 2012) to serve as a supervised learning baseline. These models rely on access to labeled training data and were evaluated using varying amounts of data to understand the impact of data availability on model performance. In our experiments, we used the `max_depth` parameter of the decision tree model to control the maximum depth of the tree.

4.2 Data

We selected datasets from health, finance, environment, and politics to evaluate the adaptability of our method across diverse real-world tasks. The datasets used are:

Diabetes (National Institute of Diabetes and Digestive and Kidney Diseases, 2016) A health-related dataset focused on predicting the onset of diabetes based on individual measurements.

Credit (UCI Machine Learning Repository, 1994) A financial dataset containing information on past loans. The target variable predicts whether an individual is likely to repay the loan.

Weather (Kumar, 2023) A binary weather prediction dataset (sunny vs. rainy) based on environmental features like temperature and humidity, derived from a multiclass dataset by selecting two contrasting climates.

Hepatic Damage (Soriano, 2022) This dataset was binarized to answer: *Is there liver damage?* Patients with Hepatitis C, Fibrosis, or Cirrhosis were labeled *True*, healthy individuals *False*, and suspected cases were excluded.

Presidential Approval (Centro de Estudios Públicos, 2024) A binary classification task predicting whether individuals approve or disapprove of the Chilean government, based on demographic features such as age, religion, and education. The dataset is drawn from a 2024 public survey, beyond the training cutoffs of the LLMs used (OpenAI, 2023), providing a robust test of generalization to unseen contexts.

Abortion Opinion (Centro de Estudios Públicos, 2024) Also sourced from the same 2024 public survey, this task predicts views on abortion using the same input features. The original five-point question was binarized by retaining only the most polarized responses: full support vs. total prohibition.

5 Results

Table 1 summarizes the ROC-AUC scores for each method. For tree-based methods, we selected the tree max depth that yielded the strongest performance on average across datasets and shot configurations; the distribution of ROC-AUC scores is shown in Figure 2. The compared methods include TabLLM, LLM-generated decision trees (LLM Trees), and traditional decision trees trained with the CART algorithm (DT). Full results, including accuracy scores and performance across all depths and shot counts, are provided in Tables 3 and 4 in the Appendix.

Performance of TabLLM Zero-Shot TabLLM demonstrated moderate performance, excelling in simpler datasets like *Weather*, where best score achieved is 91% with only 8 shots. However, it struggled with datasets such as *Presidential Approval* and *Credit*, achieving lower scores, around 55%.

Performance of Zero-Shot LLM Tree Our LLM-Tree outperformed both TabLLM and traditional DT on three of the six datasets. In *Abortion*, the Llama-4 variant (max depth = 5) achieved a ROC–AUC of 77%, outperforming all other competitors. In *Diabetes*, the GPT-4o-mini model (max depth = 3) obtained 76 % ROC–AUC—surpassed only by the traditional DT trained with all training data. Performance on *Presidential Approval* was similarly competitive: the Llama-4 tree delivered around 58% ROC–AUC for each of shots, outperforming TabLLM, but still behind the traditional DT trained on the full data. In contrast, performance is weaker on the remaining datasets: in *Weather*, the best configuration (256 shots with Llama 4) reached 88% ROC–AUC but remained below TabLLM, while in *Credit* and *Hepatic damage*, scores are around 52%. The results indicate that there is no best LLM in all cases to build the tree via LLMs. Also, in some cases, incorporating more shots into the LLM Trees can be negative.

Traditional Decision Trees with Full Data Traditional DTs trained on full datasets outperformed almost all cases the zero- or few-shot methods. The performance of these trees in the *Weather* and *Hepatic damage* sets stands out, where for 128 and 256 shots it already possesses over 80% roc-auc. However, it's surprising that in the *Abortion* dataset, where the traditional trees have their best roc-auc of 73% using all the data, the method of Trees constructed via LLMs outperforms it, even when using very few shots.

Low-Data Scenarios. In low-data scenarios (e.g., $4-32$ shots), traditional DTs exhibited limitations, highlighting the competitive performance of LLM-based zero- and few-shot approaches in resource-constrained environments. For example, with less than 32 shots, traditional DTs are outperformed by DT constructed via LLM method in datasets such as *Abortion*, *Presidential Approval*, and *Diabetes*, demonstrating the potential of LLM-based methods in handling limited data scenarios.

5.1 Interpretability Analysis

A key advantage of our approach is its ability to combine the interpretability of decision trees with zero-shot learning, making it well-suited for low-

Dataset	Method	Number of Shots								
		0	4	8	16	32	64	128	256	all
Abortion	DT (max depth=3)	—	.52	.53	.58	.59	.66	.69	.73	.73
	LLM Tree Llama 4 (max depth=5)	.77	.76	.75	.75	.73	.73	.72	.73	—
	LLM Tree gpt-4o-mini (max depth=3)	.63	.63	.62	.62	.62	.64	.63	.64	—
	TabLLM Llama 4	.66	.62	.64	.65	.64	.64	.69	.69	—
	TabLLM gpt-4o-mini	.63	.62	.65	.61	.66	.67	.63	.66	—
Presidential Approval	DT (max depth=3)	—	.52	.52	.52	.52	.54	.57	.58	.63
	LLM Tree Llama-4 (max depth=5)	.58	.58	.56	.55	.56	.58	.60	.59	—
	LLM Tree gpt-4o-mini (max depth=3)	.56	.56	.56	.56	.57	.57	.57	.56	—
	TabLLM Llama-4	.58	.52	.53	.51	.51	.52	.50	.51	—
	TabLLM gpt-4o-mini	.51	.56	.55	.55	.52	.54	.53	.54	—
Credit	DT (max depth=3)	—	.50	.53	.55	.56	.60	.63	.68	.70
	LLM Tree Llama-4 (max depth=5)	.50	.50	.50	.50	.50	.50	.51	.52	—
	LLM Tree gpt-4o-mini (max depth=3)	.50	.50	.50	.50	.50	.50	.50	.50	—
	TabLLM Llama-4	.51	.54	.57	.61	.58	.59	.63	.65	—
	TabLLM gpt-4o-mini	.55	.55	.54	.55	.54	.51	.53	.53	—
Diabetes	DT (max depth=3)	—	.55	.60	.62	.64	.66	.70	.72	.79
	LLM Tree Llama-4 (max depth=5)	.59	.61	.62	.64	.66	.68	.67	.69	—
	LLM Tree gpt-4o-mini (max depth=3)	.70	.68	.69	.68	.71	.74	.74	.76	—
	TabLLM Llama-4	.67	.68	.68	.69	.71	.71	.72	.73	—
	TabLLM gpt-4o-mini	.67	.68	.66	.65	.66	.64	.65	.66	—
Hepatic damage	DT (max depth=3)	—	.52	.59	.67	.70	.75	.81	.84	.95
	LLM Tree Llama-4 (max depth=5)	.56	.55	.56	.55	.55	.55	.54	.54	—
	LLM Tree gpt-4o-mini (max depth=3)	.52	.52	.52	.53	.53	.54	.53	.53	—
	TabLLM Llama-4	.76	.81	.79	.78	.82	.86	.86	.86	—
	TabLLM gpt-4o-mini	.69	.73	.71	.79	.82	.83	.83	.83	—
Weather	DT (max depth=3)	—	.66	.84	.84	.89	.92	.94	.95	.98
	LLM Tree Llama-4 (max depth=5)	.62	.62	.64	.70	.76	.82	.86	.88	—
	LLM Tree gpt-4o-mini (max depth=3)	.68	.77	.79	.77	.79	.80	.82	.83	—
	TabLLM Llama-4	.87	.91	.91	.93	.93	.94	.95	.95	—
	TabLLM gpt-4o-mini	.83	.89	.91	.91	.91	.91	.91	.90	—

Table 1: ROC-AUC results for traditional decision trees, TabLLM, and the method presented in this paper, trees created with LLMs. For the tree-based models, the scores shown correspond to the configuration with a fixed maximum depth that produced the most robust performance.

resource scenarios. Figure 3 in the Appendix illustrates a zero-shot tree generated for the *Diabetes* dataset and compares it with a traditional tree trained on the full dataset. Despite using no labeled examples, the zero-shot tree achieves comparable predictive performance.

The zero-shot decision tree model, with a maximum depth of 5, offers a straightforward and interpretable structure, making it accessible to non-technical users. Its feature selection emphasizes domain-relevant variables like `Age`, `Glucose`, and `Insulin`, which aligns with user expectations and enhances trust in the model's decisions. The use of simple, intuitive decision thresholds, such as `Age` ≤ 25 or `Glucose` ≤ 140, further supports the interpretability of the model.

However, this approach sacrifices fine-grained distinctions in favor of simplicity, which may reduce accuracy and limit the model's ability to capture subtler patterns that traditional decision trees can identify. The zero-shot tree, with less nodes compared to the fully trained tree, reflects this trade-off. Additionally, the exclusion of features such as `Pregnancies`, `Blood Pressure`, and `Diabetes Pedigree Function` highlights a balance between interpretability and comprehensiveness in feature selection.

Overall, the zero-shot decision tree excels in interpretability but may lack robustness due to its simplified decision-making process, making it better suited for applications where ease of understanding takes precedence over detailed accuracy.

6 Discussion and Conclusion

This method leverages the LLMs' pre-trained knowledge to generate interpretable models without requiring labeled data. Our approach mimics the CART algorithm via iterative prompting to produce decision trees that rival traditional supervised models in low-data settings. This section discusses the method's advantages and limitations and suggests directions for future work.

6.1 Advantages

Efficiency and Deployment Although constructing the tree is computationally demanding, the re-

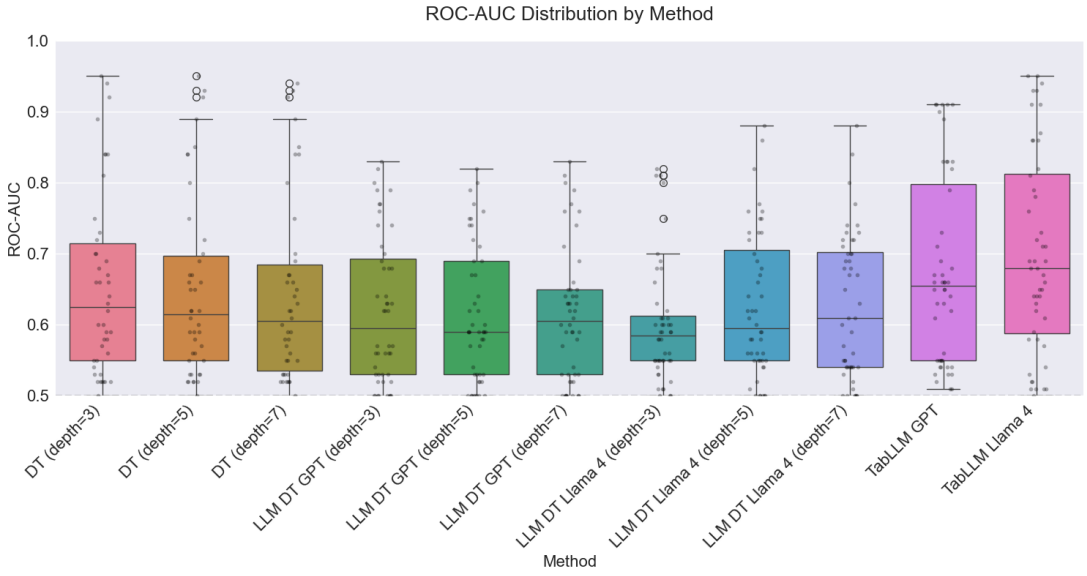

Figure 2: Boxplot of ROC–AUC scores for each method across all *dataset–shot* combinations. Each box represents the distribution of 56 scores (6 datasets $\times$ 7 shot counts) for a given method: TabLLM, LLM Trees, and traditional decision trees. This visualization guided the choice of the `max_depth` used in Table 1.

sulting model is lightweight and efficient at inference time. Unlike prompt-based zero-shot classifiers like TabLLM, our decision trees incur no additional runtime cost per prediction, making them well-suited for deployment in real-time or resource-constrained environments.

Interpretability The use of decision trees ensures model transparency, a key requirement in domains such as healthcare, law, or finance. Our method preserves this advantage while replacing data-driven splits with LLM-inferred structure, enabling users to inspect the reasoning behind each decision path.

Performance in Low-Data Settings By relying on the semantic and statistical knowledge encoded in LLMs, our method performs competitively where traditional methods fail due to lack of data. This makes it a promising tool for early-stage modeling, hypothesis generation, or deployment in domains with expensive or sensitive data collection.

6.2 Future Work

Interactive Refinement Introducing a human-in-the-loop refinement process could improve contextual relevance and mitigate LLMs biases. Experts might refine or validate the tree structure iteratively during its construction, combining automation with expert oversight.

Domain-Specific Fine-Tuning Fine-tuning the model on domain-specific data could enhance the accuracy and relevance of the constructed trees. Future work could explore techniques for targeted adaptation of LLMs to specific fields, ensuring alignment with the nuances of specialized datasets.

Bias Mitigation Biases embedded in the LLM can influence the construction of decision trees. Future research could integrate fairness-aware algorithms into the tree-building process, ensuring that attribute selection, discretization, and splitting decisions account for fairness considerations and regulatory requirements.

6.3 Final Remarks

Our findings suggest that zero-shot decision trees built via LLMs are a viable alternative to supervised models in data-scarce environments. The method combines interpretability, simplicity, and generalization from pretraining, defining a new baseline for interpretable classification without labeled data. Future work should address the limitations of model bias, opacity in reasoning, and domain misalignment to make this framework suitable for broader adoption.

Acknowledgements Some phrasing and edits in this paper were assisted by ChatGPT.

Limitations

The use of LLMs in the zero-shot model introduces inherent risks related to bias and discrimination, as these models are trained on vast amounts of data that may contain historical and systemic biases. This may result in models that not only reflect but also reinforce these biases.

LLM Bias LLMs are trained on vast amounts of text data from diverse sources, inevitably including societal biases. These biases can be encoded into the knowledge base of the LLM and may inadvertently affect its outputs. When used for decision tree construction, these biases could manifest in selecting attributes, discretizing attribute values, or the computation of probabilities, potentially leading to models that reflect and perpetuate these biases. For example, the LLM might prioritize certain features based on its training data rather than their relevance to the problem domain, inadvertently embedding stereotypical or unfair assumptions into the model. Careful monitoring of the LLM's outputs and evaluation of resulting models are needed to detect and correct this bias. Techniques such as auditing the generated trees for fairness, analyzing the distributions of decisions across sensitive attributes, and leveraging fairness-aware metrics can help identify and address bias. Additionally, incorporating domain expertise to cross-validate the model's decisions can provide a safeguard against unintentional perpetuation of harmful biases (Mehrabi et al., 2021).

Dependency on Pre-trained Knowledge The proposed method relies heavily on the pre-trained knowledge embedded in the LLM. While this allows for zero-shot decision tree construction, it also means that the quality and accuracy of the trees are inherently limited by the breadth and depth of the LLM's training data (Bommasani et al., 2021). If the LLM lacks sufficient knowledge about a particular domain or exhibits inaccuracies in its responses, this could compromise the performance of the resulting decision tree. One potential mitigation strategy is fine-tuning the LLM on domain-specific data by tailoring the model's knowledge to the specific problem context.

Interpretability vs. LLM Complexity One of the key advantages of decision trees is their interpretability. However, the involvement of an LLM introduces a layer of complexity that may not be fully transparent. For instance, the rationale behind the LLM's decisions during attribute selection or probability assignment might not always be explainable, potentially reducing the model's overall transparency. Achieving a balance between the clarity of decision trees and the inherent complexity of LLM operations is essential and raises important research questions.

References

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

Rishi Bommasani, Drew A Hudson, Ehsan Adeli, Russ Altman, Simran Arora, Sydney von Arx, Michael S Bernstein, Jeannette Bohg, Antoine Bosselut, Emma Brunskill, and 1 others. 2021. On the opportunities and risks of foundation models. *arXiv preprint arXiv:2108.07258*.

Vadim Borisov, Tobias Leemann, Kathrin Seßler, Johannes Haug, Martin Pawelczyk, and Gjergji Kasneci. 2022. Deep neural networks and tabular data: A survey. *IEEE transactions on neural networks and learning systems*.

Leo Breiman. 2017. *Classification and regression trees*. Routledge.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, and 1 others. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

Centro de Estudios Públicos. 2024. Encuesta cep no. 92: National public opinion survey, aug–sep 2024. https://www.cepchile.cl/encuesta/encuesta-cep-n-92/. Micro-data and user manual. Accessed 19 May 2025.

Jintai Chen, Jiahuan Yan, Qiyuan Chen, Danny Ziyi Chen, Jian Wu, and Jimeng Sun. 2023. Excelformer: A neural network surpassing gbdts on tabular data. *arXiv preprint arXiv:2301.02819*.

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*, pages 785–794.

Tuan Dinh, Yuchen Zeng, Ruisu Zhang, Ziqian Lin, Michael Gira, Shashank Rajput, Jy-yong Sohn, Dimitris Papailiopoulos, and Kangwook Lee. 2022. Lift: Language-interfaced fine-tuning for non-language machine learning tasks. *Advances in Neural Information Processing Systems*, 35:11763–11784.

Léo Grinsztajn, Edouard Oyallon, and Gaël Varoquaux. 2022. Why do tree-based models still outperform deep learning on typical tabular data? *Advances in neural information processing systems*, 35:507–520.

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*, pages 5549–5581. PMLR.

Noah Hollmann, Samuel Müller, Katharina Eggensperger, and Frank Hutter. 2022. Tabpfn: A transformer that solves small tabular classification problems in a second. *arXiv preprint arXiv:2207.01848*.

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. *Advances in neural information processing systems*, 30.

Ricardo Knauer, Mario Koddenbrock, Raphael Wallsberger, Nicholas M Brisson, Georg N Duda, Deborah Falla, David W Evans, and Erik Rodner. 2024. "oh llm, i'm asking thee, please give me a decision tree": Zero-shot decision tree induction and embedding with large language models. *arXiv preprint arXiv:2409.18594*.

Nikhil Kumar. 2023. Weather type classification dataset. https://www.kaggle.com/datasets/nikhil7280/weather-type-classification. Accessed 19 May 2025.

Zachary C Lipton. 2018. The mythos of model interpretability: In machine learning, the concept of interpretability is both important and slippery. *Queue*, 16(3):31–57.

Duncan McElfresh, Sujay Khandagale, Jonathan Valverde, Vishak Prasad C, Ganesh Ramakrishnan, Micah Goldblum, and Colin White. 2024. When do neural nets outperform boosted trees on tabular data? *Advances in Neural Information Processing Systems*, 36.

Ninareh Mehrabi, Fred Morstatter, Nripsuta Saxena, Kristina Lerman, and Aram Galstyan. 2021. A survey on bias and fairness in machine learning. *ACM computing surveys (CSUR)*, 54(6):1–35.

National Institute of Diabetes and Digestive and Kidney Diseases. 2016. Pima indians diabetes database. https://www.kaggle.com/datasets/uciml/pima-indians-diabetes-database. Kaggle mirror of the original UCI dataset. Accessed 19 May 2025.

OpenAI. 2023. Models. https://platform.openai.com/docs/models#gpt-4o. Accessed: January 20, 2024.

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. 2012. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

J Ross Quinlan. 2014. *C4. 5: programs for machine learning*. Elsevier.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of machine learning research*, 21(140):1–67.

Cynthia Rudin. 2019. Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. *Nature machine intelligence*, 1(5):206–215.

Victor Sanh, Albert Webson, Colin Raffel, Stephen Bach, Lintang Sutawika, Zaid Alyafeai, Antoine Chaffin, Arnaud Stiegler, Arun Raja, Manan Dey, and 1 others. 2022. Multitask prompted training enables zero-shot task generalization. In *International Conference on Learning Representations*.

Ravid Shwartz-Ziv and Amitai Armon. 2022. Tabular data: Deep learning is not all you need. *Information Fusion*, 81:84–90.

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

Federico Soriano. 2022. Hepatitis c prediction dataset. https://www.kaggle.com/datasets/fedesoriano/hepatitis-c-dataset. Accessed 19 May 2025.

Aofeng Su, Aowen Wang, Chao Ye, Chen Zhou, Ga Zhang, Guangcheng Zhu, Haobo Wang, Haokai Xu, Hao Chen, Haoze Li, and 1 others. 2024. Tablegpt2: A large multimodal model with tabular data integration. *arXiv preprint arXiv:2411.02059*.

UCI Machine Learning Repository. 1994. German credit (statlog) dataset. https://www.openml.org/d/31. OpenML ID 31. Accessed 19 May 2025.

A Vaswani. 2017. Attention is all you need. *Advances in Neural Information Processing Systems*.

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

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*, pages 3323–3333.

Yazheng Yang, Yuqi Wang, Guang Liu, Ledell Wu, and Qi Liu. 2024. Unitabe: A universal pretraining protocol for tabular foundation model in data science. In *The Twelfth International Conference on Learning Representations*.

Pengcheng Yin, Graham Neubig, Wen-tau Yih, and Sebastian Riedel. 2020. Tabert: Pretraining for joint understanding of textual and tabular data. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8413–8426.

Tianping Zhang, Shaowen Wang, Shuicheng Yan, Jian Li, and Qian Liu. 2023a. Generative table pretraining empowers models for tabular prediction. *arXiv preprint arXiv:2305.09696*.

Tianshu Zhang, Xiang Yue, Yifei Li, and Huan Sun. 2023b. Tablellama: Towards open large generalist models for tables. *arXiv preprint arXiv:2311.09206*.

Yufan Zhuang, Liyuan Liu, Chandan Singh, Jingbo Shang, and Jianfeng Gao. 2024. Learning a decision tree algorithm with transformers. *arXiv preprint arXiv:2402.03774*.

A Statistics for Data

Table 2 shows that class imbalance varies considerably across corpora. WEATHER is perfectly balanced, whereas HEPATIC DAMAGE is highly skewed ($\approx$9:1), and PRESIDENTIAL APPROVAL presents a moderate 2:1 ratio.

Pre-processing. All column names were normalised to short, descriptive English tokens to improve LLM comprehension. For categorical variables, categories representing $< 5\%$ of the data were merged into an `OTHER` label. Multiclass corpora were binarised by keeping the two most extreme categories (e.g. *rainy* vs. *sunny* in WEATHER). No additional filtering or resampling was performed.

B Model Size and Budget

This section details the models used, their parameters, and the computational and financial resources required for running the experiments. We focus on inference-only usage of LLMs via cloud APIs, alongside local experiments for traditional decision trees. Cost estimates reflect real-world deployment conditions and highlight the accessibility of our zero-shot approach in terms of both model size and compute budget.

Models

- **GPT-4o-mini** (2024-07-18, OpenAI API). Parameter count not publicly disclosed[1].
- **Llama 4 Maverick 17B Instruct (128E)** via OpenRouter. 17 B parameters.
- **CART baselines.** Depths 3/5/7; each tree $< 2^d$ leaves, trained with scikit-learn.

Compute.

- *LLM inference only* (no fine-tuning).
- **GPT-4o-mini**: 41.735M input tokens, 2.812M output tokens $\rightarrow$ USD \$ 63 (TabLLM evaluation) + \$ 12 (our LLM Tree generation) = \$ 75.
- **Llama 4 17B**: 314 k tokens total $\rightarrow$ USD \$ 60.06 (evaluation + tree construction).
- **Hardware.** All LLM calls executed via cloud APIs from Santiago, Chile. Local experiments (data prep, CART training, evaluation scripts) ran on an HP Victus notebook with an NVIDIA RTX 3060 6 GB GPU and 16 GB RAM (Python 3.12.1, scikit-learn 1.4.2).

Budget summary. Total cloud spend: \$ 135.06. No additional GPU hours beyond API usage were incurred; local GPU utilization was solely for feature engineering, plotting, and traditional decision tree training.

C Construted Trees

The left panel in Figure 3 shows a tree produced by the LLM-Tree method (GPT-4o-mini, zero-shot), while the right panel displays a traditional CART decision tree constrained to a maximum depth of 5. These visualizations highlight structural differences between language-model–generated splits and those obtained through conventional impurity minimization.

[1] Industry estimates place the model in the 15–25B range; we report no official figure.

Dataset	Rows	Cols	Train	Test	Class balance	
					Class 0	Class 1
Diabetes (PIMA)	768	9	614	154	500	268
German Credit	1,000	21	800	200	700 (good)	300 (bad)
Weather (Rain/Sun)	6,600	11	5,280	1,320	3,300 (rainy)	3,300 (sunny)
Hepatic Damage	582	13	466	116	526 (no)	56 (yes)
Presidential Approval	628	13	502	126	428 (disapprove)	200 (approve)
Abortion Opinion	334	14	267	67	219 (pro-choice)	115 (anti)

Table 2: Final dataset sizes after cleaning. Splits are 80/20 stratified train–test. No separate dev set; hyperparameters were fixed *a priori*.

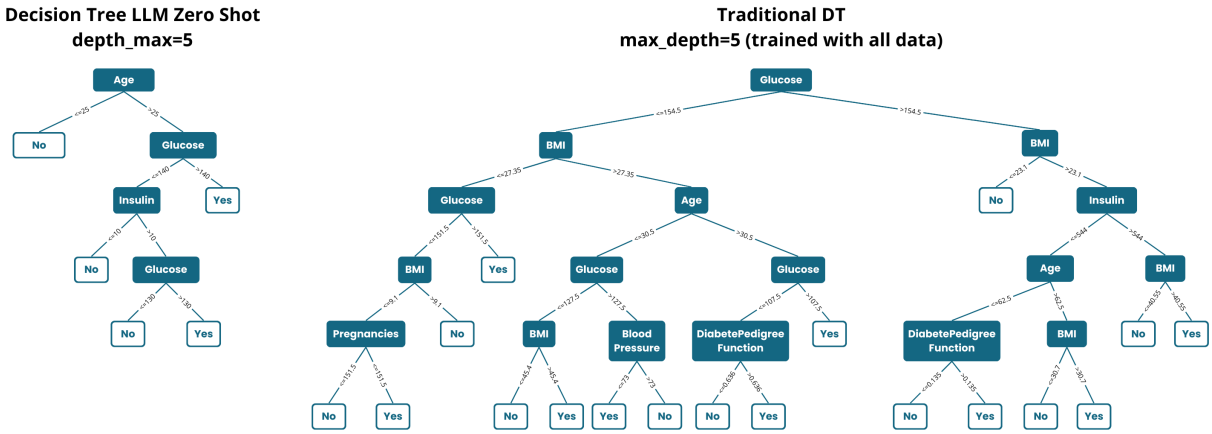

Figure 3: Comparison of decision trees: (Left) Zero-shot method with LLMs and (Right) CART algorithm for the *Diabetes* dataset. These trees are simplified versions where branches leading to unanimous outcomes (*Yes* or *No*) are condensed to single nodes.

D Prompts

This section presents the prompts in each step requiring an LLM call. Notably, two prompts are used for the three steps involving LLMs: one instructs the LLM to perform a task, and the other extracts its response, the parser. The approach was refined to request direct, concise answers instead of detailed analysis to improve efficiency and reduce computational costs.

Each prompt consists of three parts: the system message, the assistant's response, and the user's request.

D.1 Attribute Splitting Proposals

D.1.1 Get Numerical Features Splits

System message:

> *You are an expert in classifying* `{problem}` *without data. Your role is to propose a single numeric value to divide the data into two groups based on the attribute* `{feature}`*, don't analyze or reason too much, delivers the numerical value directly. You must choose the value imagining that you have data to answer it. You must respect the restrictions given to you. If there is more than one restriction for the same attribute you must respect all restrictions.*

Assistant message:

> *Understood. I will reason and provide a numerical value based solely on my knowledge, ensuring the value of the attribute* `{feature}` *meets ALL the constraints:* `{branch context}`*. I will provide a single numeric value to divide the data into two groups in a short response. If there is more than one restriction for the same attribute, all restrictions must be respected.*

User message:

> *What value of the attribute* `{feature}` *(that meets the constraints:* `{branch context}`*) should I use to divide the data into two groups? Choose a single numeric value, and answer only with the value, without any other text.*

D.1.2 Parse Numerical Features Splits

System message:

> *You are an expert in extracting information from responses. Your task is to extract the numeric value suggested for dividing data into two groups based on a specific feature. Don't analyze or reason too much, delivers the numerical value directly.*

Assistant message:

> *Understood. I will receive your response in the following format: Input:<response>. Then, I will extract and provide the numeric value after the prefix 'Output:'<numeric value>. For example: Input: 'For the numeric feature age, the best division is 15.5', I will respond 'Output: 15.5' without adding anything else. If no numeric value or grouping can be extracted, I will respond with the string 'Nothing'.*

User message:

> *For the numeric feature '*`{feature}`*', extract the numeric value indicating the cutoff point to divide the data into two groups. Input:* `{response text}`*. I will answer only with the numeric value and the text 'Output:', and nothing else.*

D.1.3 Get Categoric Features Splits

System message:

> *You are an expert in* `{problem}`*. Your task is to use your knowledge to propose two disjoint groups of categories based on the categorical attribute* `{feature}` *with possible categories:* `{possible values}`*, don't analyze or reason too much, delivers the groupings directly.. You have to try to make the groups formed as homogeneous as possible according to the problem variable that has as labels* `{target labels}`*.*

Assistant message:

> *Understood. I will reason and provide two disjoint grpups of categories based solely on my knowledge, ensuring the groupings meet the constraints:* `{branch context}`*. I will provide two disjoint groupings of categories to divide the data in a short response. I will try to to make the groups as homogeneous as possible.*

User message:

> *For the categorical attribute* `{feature}` *(that meets the constraints:* `{branch context}`*) with possible values:* `{possible values}`*, what two groups of categories should I use to divide the data into two groups as homogeneously as possible according to the* `{problem}` *variable with labels* `{target labels}`*? Choose two disjoint groupings, and answer only with the groupings, without any other text.*

D.1.4 Parse Categoric Features Splits

System message:

> *You are an expert in extracting information from responses. Your task is to extract the two disjoint groupings of categories suggested for dividing data into two groups based on a specific feature.*

Assistant message:

> *Understood. I will receive your response in the following format: Input:<response>. Then, I will extract and provide the two disjoint groupings after the prefix 'Output:'<grouping_1>;;<grouping_2>. For example: Input: 'For the categorical feature color, the best division is to create a group of colors [red, green] and another group of colors [blue]', I will respond 'Output: red, green;;blue' without adding anything else. If no groupings can be extracted, I will respond with the string 'Nothing'.*

User message:

> *For the categorical feature '*`{feature}`*' with possible values:* `{possible values}`*, extract the two disjoint groupings of categories suggested for dividing the data into two groups. Input:* `{response text}`

D.1.5 Get Probabilities Estimation

System message:

> *You are an expert in estimating the labels probabilities at the nodes of a decision tree. Your task is to provide a rough estimate of the probabilities that a given instance belongs to the possible target classes based on: the characteristics, probabilities of the previous node and the information from the new node split. Use only your general knowledge, since you do not have specific data. Remember that splits usually improve node purity. Remember also when there is little information do not give high probabilities (equal or higher than 0.9), unless you are very sure of them, because you may be overestimating. Don't analyze or reason too much, delivers the numerical value directly.*

Assistant message:

> *Understood. Although it is difficult to provide exact probabilities for a particular individual, I will reason based on the information provided and my general knowledge to provide rough estimates of specific probabilities. I will provide probability values between 0 and 1 for each label, for each possible new division. In case there is little information, I will give low probabilities (< 0.9) unless I am very sure. On the other hand, I will not give high probabilities (>= 0.9) unless I am very sure, as I could be overestimating.*

User message:

> *Considering a classification problem of* `{problem}`*, estimate the probability that a* `{instance type}`*instance type has the target variable* `{target feature}` *for each of the possible values:* `{classes}`*, based on the characteristics of the previous node:* `{previous context}` *where the probability of the target variable is* `{previous probabilities}`*, and the possible new divisions are:* `{division 1}` *or* `{division 2}`*.*

D.1.6 Parse Probabilities Estimation

System message:

> *You are an expert in extracting information from responses. Your task is to extract the label probability suggested for a classification problem to a specific label and new information. Do not provide anything other than the Output. Adhere to the format. Don't analyze or reason too much, delivers the numerical value directly.*

Assistant message:

> *Understood. I will extract the label probability to a new information from the given response and provide it after the prefix 'Output:'<probability>. If no probability can be extracted, I will respond with 'Nothing'. For example: Input: 'The probability that a* `{instance type}` *has the target variable with the value* `{class 1}` *is 0.8' Output: 0.8. Other example: Input: 'The probability that a* `{instance type}` *has the target variable with the value* `{class 2}` *is 0.3' Output: 0.3.*

User message:

> *For the classification problem with the label* `{label}` *and the following new information* `{new information}`*, extract the probability suggested for the problem. The given response is:* `{response text}`*.*

E Detailed Results Tables

Tables 3 and 4 provide the full evaluation results for all methods across datasets and shot configurations. Table 3 reports ROC-AUC scores, while Table 4 presents accuracy scores. Results are shown for all model variants, including traditional decision trees (with varying depths), LLM-generated trees, and TabLLM baselines, enabling a comprehensive comparison across methods and settings.

Dataset	Method	Number of Shots								
		0	4	8	16	32	64	128	256	all
Abortion	DT (max depth=3)	—	.52	.53	.58	.59	.66	.69	.73	.73
	DT (max depth=5)	—	.52	.53	.58	.59	.62	.67	.66	.63
	DT (max depth=7)	—	.52	.53	.57	.58	.61	.65	.67	.65
	LLM Tree Llama 4 (max depth=3)	.59	.59	.60	.60	.56	.58	.55	.56	—
	LLM Tree Llama 4 (max depth=5)	.77	.76	.75	.75	.73	.73	.72	.73	—
	LLM Tree Llama 4 (max depth=7)	.72	.73	.72	.71	.70	.73	.74	.77	—
	LLM Tree gpt-4o-mini (max depth=3)	.63	.63	.62	.62	.62	.64	.63	.64	—
	LLM Tree gpt-4o-mini (max depth=5)	.57	.58	.59	.59	.60	.62	.62	.63	—
	LLM Tree gpt-4o-mini (max depth=7)	.61	.62	.63	.62	.63	.63	.63	.65	—
	TabLLM Llama 4	.66	.62	.64	.65	.64	.64	.69	.69	—
	TabLLM gpt-4o-mini	.63	.62	.65	.61	.66	.67	.63	.66	—
Presidential Approval	DT (max depth=3)	—	.52	.52	.52	.52	.54	.57	.58	.63
	DT (max depth=5)	—	.52	.52	.53	.52	.53	.56	.56	.57
	DT (max depth=7)	—	.52	.52	.53	.52	.52	.55	.53	.53
	LLM Tree Llama-4 (max depth=3)	.66	.62	.61	.60	.60	.60	.61	.59	—
	LLM Tree Llama-4 (max depth=5)	.58	.58	.56	.55	.56	.58	.60	.59	—
	LLM Tree Llama-4 (max depth=7)	.55	.57	.55	.54	.56	.59	.61	.60	—
	LLM Tree gpt-4o-mini (max depth=3)	.56	.56	.56	.56	.57	.57	.57	.56	—
	LLM Tree gpt-4o-mini (max depth=5)	.58	.59	.59	.59	.59	.60	.59	.57	—
	LLM Tree gpt-4o-mini (max depth=7)	.58	.59	.59	.59	.60	.60	.59	.57	—
	TabLLM Llama-4	.58	.52	.53	.51	.51	.52	.50	.51	—
	TabLLM gpt-4o-mini	.51	.56	.55	.55	.52	.54	.53	.54	—
Credit	DT (max depth=3)	—	.50	.53	.55	.56	.60	.63	.68	.70
	DT (max depth=5)	—	.50	.53	.55	.57	.59	.61	.65	.67
	DT (max depth=7)	—	.50	.53	.55	.56	.58	.59	.63	.68
	LLM Tree Llama-4 (max depth=3)	.53	.50	.51	.51	.50	.51	.52	.53	—
	LLM Tree Llama-4 (max depth=5)	.50	.50	.50	.50	.50	.50	.51	.52	—
	LLM Tree Llama-4 (max depth=7)	.50	.50	.50	.50	.50	.50	.51	.52	—
	LLM Tree gpt-4o-mini (max depth=3)	.50	.50	.50	.50	.50	.50	.50	.50	—
	LLM Tree gpt-4o-mini (max depth=5)	.50	.50	.50	.50	.50	.50	.50	.50	—
	LLM Tree gpt-4o-mini (max depth=7)	.50	.50	.50	.50	.50	.50	.50	.50	—
	TabLLM Llama-4	.51	.54	.57	.61	.58	.59	.63	.65	—
	TabLLM gpt-4o-mini	.55	.55	.54	.55	.54	.51	.53	.53	—
Diabetes	DT (max depth=3)	—	.55	.60	.62	.64	.66	.70	.72	.79
	DT (max depth=5)	—	.55	.60	.62	.62	.65	.69	.72	.79
	DT (max depth=7)	—	.55	.60	.62	.62	.64	.66	.69	.77
	LLM Tree Llama-4 (max depth=3)	.50	.54	.58	.63	.68	.69	.68	.70	—
	LLM Tree Llama-4 (max depth=5)	.59	.61	.62	.64	.66	.68	.67	.69	—
	LLM Tree Llama-4 (max depth=7)	.61	.63	.65	.67	.68	.69	.69	.70	—
	LLM Tree gpt-4o-mini (max depth=3)	.70	.68	.69	.68	.71	.74	.74	.76	—
	LLM Tree gpt-4o-mini (max depth=5)	.67	.67	.69	.69	.71	.72	.74	.76	—
	LLM Tree gpt-4o-mini (max depth=7)	.64	.64	.65	.64	.66	.69	.71	.74	—
	TabLLM Llama-4	.67	.68	.68	.69	.71	.71	.72	.73	—
	TabLLM gpt-4o-mini	.67	.68	.66	.65	.66	.64	.65	.66	—
Hepatic damage	DT (max depth=3)	—	.52	.59	.67	.70	.75	.81	.84	.95
	DT (max depth=5)	—	.52	.59	.67	.70	.75	.80	.85	.95
	DT (max depth=7)	—	.52	.59	.67	.70	.75	.80	.85	.95
	LLM Tree Llama-4 (max depth=3)	.55	.55	.55	.55	.55	.55	.55	.55	—
	LLM Tree Llama-4 (max depth=5)	.56	.55	.56	.55	.55	.55	.54	.54	—
	LLM Tree Llama-4 (max depth=7)	.53	.54	.54	.54	.55	.54	.54	.54	—
	LLM Tree gpt-4o-mini (max depth=3)	.52	.52	.52	.53	.53	.54	.53	.53	—
	LLM Tree gpt-4o-mini (max depth=5)	.52	.52	.52	.53	.53	.54	.53	.53	—
	LLM Tree gpt-4o-mini (max depth=7)	.52	.52	.52	.53	.53	.54	.53	.53	—
	TabLLM Llama-4	.76	.81	.79	.78	.82	.86	.86	.86	—
	TabLLM gpt-4o-mini	.69	.73	.71	.79	.82	.83	.83	.83	—
Weather	DT (max depth=3)	—	.66	.84	.84	.89	.92	.94	.95	.98
	DT (max depth=5)	—	.66	.84	.84	.89	.92	.93	.95	.99
	DT (max depth=7)	—	.66	.84	.84	.89	.92	.93	.94	.98
	LLM Tree Llama-4 (max depth=3)	.59	.58	.61	.75	.80	.81	.81	.82	—
	LLM Tree Llama-4 (max depth=5)	.62	.62	.64	.70	.76	.82	.86	.88	—
	LLM Tree Llama-4 (max depth=7)	.67	.68	.70	.72	.74	.80	.84	.88	—
	LLM Tree gpt-4o-mini (max depth=3)	.68	.77	.79	.77	.79	.80	.82	.83	—
	LLM Tree gpt-4o-mini (max depth=5)	.64	.74	.75	.75	.77	.79	.80	.82	—
	LLM Tree gpt-4o-mini (max depth=7)	.65	.76	.76	.77	.79	.80	.81	.83	—
	TabLLM Llama-4	.87	.91	.91	.93	.93	.94	.95	.95	—
	TabLLM gpt-4o-mini	.83	.89	.91	.91	.91	.91	.91	.90	—

Table 3: ROC-AUC results obtained for each method on the six datasets, for different numbers of shots.

Dataset	Method	Number of Shots								
		0	4	8	16	32	64	128	256	all
Abortion	DT (max depth=3)	—	.58	.59	.63	.63	.69	.72	.76	.76
	DT (max depth=5)	—	.58	.59	.63	.63	.67	.71	.73	.72
	DT (max depth=7)	—	.58	.59	.63	.63	.66	.69	.70	.66
	LLM Tree Llama-4 (max depth=3)	.36	.56	.64	.64	.64	.63	.64	.65	—
	LLM Tree Llama-4 (max depth=5)	.66	.69	.69	.71	.69	.70	.71	.71	—
	LLM Tree Llama-4 (max depth=7)	.66	.69	.70	.71	.69	.72	.72	.74	—
	LLM Tree gpt-4o-mini (max depth=3)	.49	.60	.61	.63	.65	.64	.65	.65	—
	LLM Tree gpt-4o-mini (max depth=5)	.51	.52	.53	.54	.58	.61	.64	.65	—
	LLM Tree gpt-4o-mini (max depth=7)	.51	.52	.52	.53	.55	.59	.62	.66	—
	TabLLM Llama-4	.61	.64	.66	.68	.68	.70	.75	.75	—
	TabLLM gpt-4o-mini	.52	.57	.66	.62	.65	.68	.63	.65	—
Presidential Approval	DT (max depth=3)	—	.57	.60	.59	.60	.63	.64	.67	.69
	DT (max depth=5)	—	.57	.60	.59	.59	.60	.63	.65	.70
	DT (max depth=7)	—	.57	.60	.59	.58	.58	.62	.62	.63
	LLM Tree Llama-4 (max depth=3)	.73	.73	.70	.70	.67	.65	.68	.66	—
	LLM Tree Llama-4 (max depth=5)	.68	.68	.67	.68	.66	.66	.68	.67	—
	LLM Tree Llama-4 (max depth=7)	.70	.70	.68	.69	.67	.67	.70	.69	—
	LLM Tree gpt-4o-mini (max depth=3)	.64	.64	.63	.64	.66	.64	.65	.63	—
	LLM Tree gpt-4o-mini (max depth=5)	.62	.62	.63	.62	.62	.63	.64	.63	—
	LLM Tree gpt-4o-mini (max depth=7)	.61	.61	.61	.60	.61	.60	.61	.61	—
	TabLLM Llama-4	.54	.51	.51	.56	.55	.52	.56	.57	—
	TabLLM gpt-4o-mini	.69	.62	.64	.62	.62	.63	.65	.65	—
Credit	DT (max depth=3)	—	.58	.61	.62	.62	.66	.66	.68	.70
	DT (max depth=5)	—	.58	.61	.62	.62	.65	.65	.68	.68
	DT (max depth=7)	—	.58	.61	.62	.62	.64	.64	.67	.69
	LLM Tree Llama-4 (max depth=3)	.70	.69	.66	.65	.66	.66	.64	.66	—
	LLM Tree Llama-4 (max depth=5)	.70	.69	.66	.65	.66	.66	.64	.66	—
	LLM Tree Llama-4 (max depth=7)	.70	.69	.66	.65	.66	.66	.64	.66	—
	LLM Tree gpt-4o-mini (max depth=3)	.70	.67	.65	.64	.65	.64	.64	.66	—
	LLM Tree gpt-4o-mini (max depth=5)	.70	.67	.65	.64	.65	.64	.64	.66	—
	LLM Tree gpt-4o-mini (max depth=7)	.70	.67	.65	.64	.65	.64	.64	.66	—
	TabLLM Llama-4	.60	.52	.50	.60	.60	.62	.65	.70	—
	TabLLM gpt-4o-mini	.41	.41	.40	.42	.42	.40	.41	.46	—
Diabetes	DT (max depth=3)	—	.59	.64	.65	.66	.67	.70	.70	.69
	DT (max depth=5)	—	.59	.64	.66	.65	.67	.69	.71	.79
	DT (max depth=7)	—	.59	.64	.66	.65	.67	.68	.70	.77
	LLM Tree Llama-4 (max depth=3)	.34	.50	.56	.64	.68	.69	.70	.70	—
	LLM Tree Llama-4 (max depth=5)	.52	.54	.57	.62	.65	.67	.68	.69	—
	LLM Tree Llama-4 (max depth=7)	.58	.58	.60	.63	.66	.67	.68	.70	—
	LLM Tree gpt-4o-mini (max depth=3)	.56	.58	.62	.66	.70	.69	.70	.69	—
	LLM Tree gpt-4o-mini (max depth=5)	.61	.61	.62	.63	.65	.67	.70	.70	—
	LLM Tree gpt-4o-mini (max depth=7)	.62	.61	.62	.62	.62	.65	.66	.68	—
	TabLLM Llama-4	.65	.64	.65	.70	.69	.70	.71	.73	—
	TabLLM gpt-4o-mini	.58	.65	.62	.64	.65	.63	.66	.66	—
Hepatic damage	DT (max depth=3)	—	.87	.89	.92	.92	.94	.95	.96	.97
	DT (max depth=5)	—	.87	.89	.92	.92	.94	.95	.97	.98
	DT (max depth=7)	—	.87	.89	.92	.92	.94	.95	.97	.98
	LLM Tree Llama-4 (max depth=3)	.91	.90	.91	.91	.91	.91	.91	.91	—
	LLM Tree Llama-4 (max depth=5)	.91	.91	.91	.91	.91	.91	.91	.91	—
	LLM Tree Llama-4 (max depth=7)	.91	.91	.91	.91	.91	.91	.91	.91	—
	LLM Tree gpt-4o-mini (max depth=3)	.42	.86	.86	.91	.91	.91	.91	.91	—
	LLM Tree gpt-4o-mini (max depth=5)	.42	.55	.82	.90	.91	.91	.91	.91	—
	LLM Tree gpt-4o-mini (max depth=7)	.42	.55	.82	.90	.91	.91	.91	.91	—
	TabLLM Llama-4	.86	.85	.91	.93	.94	.96	.97	.97	—
	TabLLM gpt-4o-mini	.67	.58	.56	.70	.75	.77	.77	.76	—
Weather	DT (max depth=3)	—	.66	.84	.84	.89	.92	.93	.94	.95
	DT (max depth=5)	—	.66	.84	.84	.89	.92	.93	.94	.95
	DT (max depth=7)	—	.66	.84	.84	.89	.92	.93	.94	.94
	LLM Tree Llama-4 (max depth=3)	.42	.50	.56	.71	.77	.77	.77	.78	—
	LLM Tree Llama-4 (max depth=5)	.42	.48	.54	.66	.71	.74	.77	.80	—
	LLM Tree Llama-4 (max depth=7)	.61	.62	.63	.67	.70	.74	.76	.79	—
	LLM Tree gpt-4o-mini (max depth=3)	.59	.59	.61	.64	.69	.75	.77	.78	—
	LLM Tree gpt-4o-mini (max depth=5)	.60	.61	.63	.65	.69	.73	.76	.77	—
	LLM Tree gpt-4o-mini (max depth=7)	.65	.65	.67	.69	.72	.75	.77	.78	—
	TabLLM Llama-4	.87	.91	.91	.93	.93	.94	.95	.95	—
	TabLLM gpt-4o-mini	.83	.89	.91	.91	.91	.91	.91	.90	—

Table 4: Accuracy obtained by each method on the six datasets, for different numbers of*shots*.