Language Models are Weak Learners

Hariharan Manikandan [*1] **Yiding Jiang** [*1] **Zico Kolter** [1]

Abstract

A central notion in practical and theoretical machine learning is that of a *weak learner*, classifiers that achieve better-than-random performance (on any given distribution over data), even by a small margin. Such weak learners form the practical basis for canonical machine learning methods such as boosting. In this work, we illustrate that prompt-based large language models can operate effectively as said weak learners. Specifically, we illustrate the use of a large language model (LLM) as a weak learner in a boosting algorithm applied to tabular data. We show that by providing (properly sampled according to the distribution of interest) text descriptions of tabular data samples, LLMs can produce a summary of the samples that serves as a template for classification and achieves the aim of acting as a weak learner on this task. We incorporate these models into a boosting approach, which in some settings can leverage the knowledge within the LLM to outperform traditional tree-based boosting. The model outperforms both few-shot learning and occasionally even more involved fine-tuning procedures, particularly for tasks involving small numbers of data points. The results illustrate the potential for prompt-based LLMs to function not just as few-shot learners themselves, but as components of larger machine learning pipelines.

1. Introduction

Weak learners refer to classifiers that are able to attain better performance than random chance, by some given margin, on any specified distribution over training data. One of the early breakthroughs in machine learning established that this weak learning was sufficient for arbitrarily strong classification, via an ensembling procedure (Schapire, 1990). This approach in turn led to the development of boosting algorithms (Freund & Schapire, 1997), a class of approaches that continue to perform extremely well, particularly on tabular datasets that lack the input space regularity of vision or language tasks.

In a seemingly separate thread of research, large language models (LLMs) based on the transformer architecture (Vaswani et al., 2017) in recent years have come to dominate many natural language domains. These models are often finetuned on the data of new downstream tasks (Devlin et al., 2018; Liu et al., 2019), but in recent years have also been shown to exhibit strong performance as zero-shot or few-shot learning solely via prompting the model (Brown et al., 2020) with a piece of context string.

In this paper, we align these two threads of research and ask a simple question: *can LLMs also serve as weak learners in a boosting framework, specifically on tabular data (where boosting methods are most commonly applied)*? We answer this question largely in the affirmative. Specifically, we show that by appropriately converting tabular data to text form, and asking LLMs to summarize a carefully chosen set of examples from the data, we produce a summary of the examples that can serve as a template (i.e., a prompt) for a tabular data classifier, and one which typically achieves this weak learning aim. This enables us to correspondingly integrate this collection of LLM-generated weak learners into a boosting framework. In essence, the idea builds upon the fact that LLM can generate prompts for itself (Zhou et al., 2022b; Zhang et al., 2022; Yu et al., 2022), which in turn can act as weak learners for a classification problem.

We show that the resulting approach performs well in many settings, easily outperforming zero-shot and few-shot classification, as well as "single-shot" summaries generated by the LLM. This is all done without any retraining or finetuning of the LLM itself, but rather only via prompting. Furthermore, on certain domains (particularly those with very few examples, where leveraging the prior knowledge built into LLMs would be of particular importance), we show that the approach can even outperform traditional tree-based boosting and LLM-based finetuning methods and its performance would likely improve as LLMs capabilities improve. Overall, we believe this work highlights the potential of incorporating LLMs as sub-routines of a larger machine learning system.

[*]Equal contribution [1]Carnegie Mellon University, PA, USA. Correspondence to: Hariharan Manikandan <hmanikan@andrew.cmu.edu>.

Work presented at the ES-FoMo Workshop at ICML 2023.

2. LLM Summary Boosting

We refer to our main methodology as *LLM Summary Boosting*, as the core learning process is one that uses a language model to create a summary of (specifically chosen) samples from the dataset; these summaries themselves function as prompts by which we can make predictions on new examples. Finally, we construct an ensemble of these summaries that gives the overall predictions on new data points.

2.1. Data conversion

To utilize large language models (LLMs) with tabular data, it is necessary to first convert the records into natural language descriptions. We will refer to these as *data descriptions*. Template matching, commonly used in previous approaches (Dinh et al., 2022), inserts attribute values into predefined templates. However, this approach often produces unnatural descriptions that differ from how humans might describe the data. Depending on the dataset, designing the template by hand can also be challenging. To overcome this, we propose using LLMs as a more suitable solution.

We can get these data descriptions with little effort by *zero-shot prompting* the LLM with the textual representation of the tabular record (e.g., parsed JSON) and the metadata, as we will show in Appendix B. Particularly for numerical features, we encode them efficiently by binning their values and using quantifiers such as "low," "medium," and "high," to describe these bins. In Appendix E.6 and C, we will illustrate and compare several such approaches for handling continuous features. Interestingly, we find that descriptions generated by LLM this way often outperform than those from a template.

2.2. Weak learning via summarization

A typical method for performing few-shot learning with large language models (LLMs) involves providing a small number of demonstrations of the intended task (data descriptions) as a prompt and then asking the model to generate an answer. However, for tabular data, there may be a larger number of data points that do not fit within the LLM context. Furthermore, we observed that increasing the number of examples in the context naively does not always improve performance (Figure 4 - right bottom), and there was no obvious way to manage weighted distributions over examples as is required in boosting methods. These observations necessitate alternative approaches to weak learning via LLMs.

We propose instead that *producing summaries of a collection of examples* can serve as a powerful proxy for learning models based upon some number of examples. As illustrated in Figure 1, we can perform summarization by calling the LLM with a set of data descriptions appended with the directive "`tldr`". This resulting summary can be seen as a hypothesis as it provides an explanation for the data. By using the summary as a prompt, the LLM in turn uses the hypothesis to perform inference instead of the raw data description. Since the sampled summary can be noisy, we generate a fixed number of them and pick the one with the least validation error rate (refer Appendix E.2). Several methods of building such summaries are possible as we will show in Appendix C.

(Weighted) Cluster Sampling. Since the context size of existing LLMs is limited, we cannot in general fit the entire dataset into the context for summarization. Furthermore, boosting algorithms require that we provide weak learners on *weighted* samples of the training set, effectively guiding the boosting process to focus on "harder" examples as the boosting process continued. Thus, instead of attempting to summarize the entire dataset, we propose to use only a representative subset of the dataset. The size of this subset is governed by the maximum LLM input length and size of the data descriptions. To select this representative subset, we use weighted stratified sampling using subpopulations defined by clusters of language embeddings of each data description. In particular, these language embeddings are from GPT-3 and we use *hierarchical agglomerative clustering* (Nielsen, 2016) to identify clusters in the embedding. As we will show in Appendix E.7 and C, this process is able to consistently produce weak learners, and able to improve upon random guessing under the distribution of interest.

2.3. Boosting

Finally, we use the AdaBoost (Freund & Schapire, 1997) algorithm to produce an ensemble with these collections of summary-based weak learners. The central idea of AdaBoost is to fit a sequence of weak learners on repeatedly modified versions of the data. The algorithm is carried out over T rounds, where the weights of the training data points are adjusted based on the training error.

Given a new data point, the predictions from classifiers from all rounds are then combined through a weighted majority vote to produce the final prediction. We use the error on a holdout validation set to determine the number of rounds `T` and stratified cluster sampling to select mini-batches. Unlike the summary process where we resample multiple times to find the best learner, the boosting process returns immediately after finding a summary that does better than random guessing. The complete algorithm incorporating various runtime optimizations, is discussed under Appendix E.8. We analyze its time complexity in Appendix E.10.

3. Experiments

We conduct all of our experiments with OpenAI's GPT-3 API (Brown et al., 2020) and choose a collection of 18 tabu-

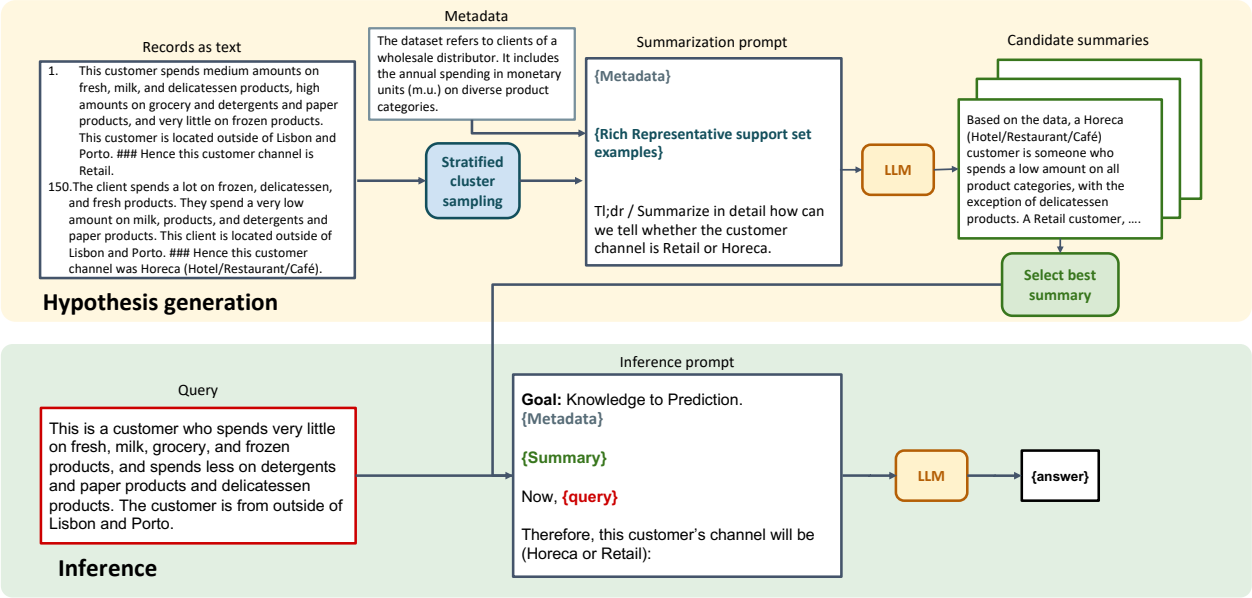

Figure 1. The process of generating summaries and using them to make predictions on new data. The top half describes how the weak learning hypothesis (summary) is generated. The bottom half illustrates how the summary is used to perform inference.

lar datasets from the UCI dataset (Dua & Graff, 2017) and OpenML (Vanschoren et al., 2014). All main experiments are done with the `Curie` variant of GPT-3 unless otherwise specified, which has 13B parameters[1]. We compare the following methods:

- `Zero-shot`: query the language model with the data description and ask the model to complete the answer (refer Appendix E.4).
- `Few-shot`: provide a few labeled data descriptions of the training data as the context and ask the model to complete the answer for a new data description. The setting is explained in Appendix E.5.
- `Summary` (ours): generate a population of summaries given a list of data descriptions with cluster sampling and pick the summary with the lowest validation error; use the best summary as the context and ask the model to complete the answer for a new data description.
- `Summary Boosting` (ours): use `Summary` as a subroutine for creating weak learner in AdaBoost.

Furthermore, we compared `Summary Boosting` against popular baselines for tabular data that do not use prompting:

- `KNN`: first embed the data descriptions with the GPT-3 embedding API [2] and then use K-nearest neighbor to classify a new data description. This simple baseline demonstrates how much information can the representation produced by LLMs provide about the tasks.
- `LIFT` (Dinh et al., 2022): Language-Interfaced Fine-Tuning (LIFT) finetunes the LM with data descriptions (without binning) and their corresponding labels in a zero-shot.
- `TabPFN` (Hollmann et al., 2022): TabPFN is a transformer-based architecture that performs Bayesian inference on the entire training and test data points at the same time.
- `XGBoost` (Chen & Guestrin, 2016): XGBoost (eXtreme Gradient Boosting) is a regularized gradient boosting algorithm that is widely used for tabular data.

For each method and dataset, we use a $50/10/40$ split for train, validation, and test sets and repeat each experiment for 3 random seeds The results are shown in Table 1. Further, the ablation experiments are discussed in Appendix C.

3.1. Analysis of Prompting-based Methods

As a general trend from Table 1, test performance improves in the order of `Zero-shot` < `Few-shot` < `Summary` < `Summary Boosting`. Firstly, unlike most works on zero-shot reasoning with LLMs, the LLMs do not have enough prior knowledge to make the correct prediction without additional information. As a result, we observe that `Zero-shot` performs poorly on all of the datasets. This observation highlights the necessity of learning from

[1] We use `Curie` because it is more cost-effective for large-scale experiments.

[2] https://beta.openai.com/docs/guides/embeddings

Table 1. Test errors for prompting-based and other popular baselines on all datasets (↓). **Data Type** indicates the number and types of attributes the dataset has (`c` is continuous and `d` is discrete). The abbreviation `S-Boost` in the column refers to `Summary Boosting`. **Size** indicates the number of data points in that dataset. In square bracket (if present) next to every dataset name, we provide its acronym referred to in our main text. In the bracket next to each dataset name is either the OpenML ID of the dataset or a reference to the dataset's associated publication. The complete set of results, including error bars, is given in Appendix tables 2 and 3.

Dataset	Data Type	Size	Zero-shot	Few-shot	Summary	S-Boost	LIFT	KNN	TabPFN	Xgboost
caesarian [cae] (42901)	1c4d	80	0.425	0.388	0.350	0.300	0.312	0.300	0.425	0.412
iris (61)	4c0d	150	0.680	0.460	0.275	0.193	0.100	0.106	0.027	0.054
tae (48)	1c4d	151	0.556	0.494	0.474	0.454	0.480	0.532	0.450	0.464
glass (41)	9c0d	214	0.486	0.473	0.466	0.370	0.218	0.294	0.158	0.254
breast-cancer [bc] (13)	7c5d	277	0.754	0.516	0.337	0.288	0.318	0.277	0.264	0.270
visualizing-environmental [ve] (678)	3c0d	111	0.522	0.366	0.304	0.268	0.430	0.308	0.370	0.279
analcatdata-chlamydia [ac] (535)	2c2d	100	0.200	0.200	0.170	0.170	0.180	0.170	0.090	0.110
wine (43571)	13c0d	178	0.820	0.674	0.475	0.320	0.065	0.214	0.040	0.040
blood-transfusion-center [btc] (1464)	4c0d	748	0.544	0.430	0.258	0.240	0.270	0.238	0.209	0.219
somerville-happiness [shs] (Koczkodaj, 2018)	0c7d	143	0.416	0.385	0.422	0.350	0.419	0.326	0.392	0.406
vehicle (54)	18c0d	846	0.765	0.560	0.510	0.410	0.111	0.636	0.178	0.260
statlog-heart (Dua & Graff, 2017)	6c7d	270	0.551	0.528	0.444	0.430	0.122	0.244	0.148	0.215
verterbra-column (1524)	6c0d	310	0.714	0.435	0.327	0.262	0.192	0.318	0.135	0.187
ecoli (1011)	7c0d	336	0.581	0.562	0.480	0.270	0.126	0.211	0.036	0.066
haberman-survival (43)	3c0d	306	0.308	0.262	0.277	0.250	0.314	0.278	0.262	0.281
diabetes (37)	8c0d	768	0.446	0.400	0.360	0.344	0.324	0.353	0.238	0.234
visualizing-hamster (708)	5c0d	73	0.464	0.481	0.360	0.207	0.334	0.528	0.328	0.411
wholesale-customers (1511)	6c1d	440	0.364	0.347	0.349	0.330	0.125	0.043	0.088	0.098

the data, and unlike other tasks, the LLMs themselves do not have enough built-in knowledge to succeed at tabular data zero-shot. Since `zero-shot` does not have enough prior knowledge to classify tabular data, we use few-shot in-context learning (`Few-shot`) to see if the added information helps make better predictions. As expected, on all the datasets other than `visualizing-hamster`, and `wholesale-customers`, `Few-shot` consistently improves the test performance compared to `Zero-shot`, suggesting that this added information is crucial for LLMs to work on tabular datasets.

Unlike naively stacking examples inside the prompt in `Few-shot`, `Summary` condenses knowledge from these examples and is the first important algorithmic component of our framework for creating weak learners using LLMs. We see that `Summary` consistently improves upon `Few-shot` on all the datasets other than `haberman-survival` and `wholesale-customers`. This observation suggests that summarization is a powerful way to improve few-shot performance and has potential for even other tasks using LLMs. Finally, for every dataset we tested, boosting with summarization consistently outperforms all other prompting-based approaches. This observation corroborates our hypothesis that *LLMs with summarization are a good candidate for creating weak learners in boosting*.

3.2. Comparison to Other Tabular Methods

In Table 1, we also observed that LLMs have a hard time reasoning about continuous attributes without finetuning, especially on the `glass`, `wine`, `iris` and `vehicle` datasets. In particular, we can see that when the datasets have many continuous features, the performance of `Summary Boosting` can be considerably worse than other methods such as `LIFT` or `Xgboost`. This may be due to the fact that LLMs are fairly bad at quantitive reasoning without finetuning (Lewkowycz et al., 2022). However, we expect this issue to be addressed as LLMs become more competent at quantitative reasoning (Lewkowycz et al., 2022).

While `KNN` is a relatively simple baseline, its performance is surprisingly good at a few tasks such as `wholesale-customers`, `statlog-heart`, `ecoli` and `wine`. This highlights that LLMs have a remarkable amount of general prior knowledge about the worlds compared to methods like `XGboost` that sometimes this knowledge alone can produce good performances.

Finally, we observe that `Summary Boosting` performs very well when the size of the dataset is very small. This makes sense since the strength of using LLMs as weak learners is that they have a large amount of generic prior about the world from pre-training. When the dataset is large, this prior knowledge might become less relevant and methods like finetuning become more competitive.

4. Conclusion

LLMs have been widely used in recent years, not just for their generative abilities but for their ability to serve as zero- or few-shot learners with proper prompting. This work aims to situate them within another context of "simple" learning algorithms – the weak learner paradigm that forms the foundation of boosting approaches. We show that leveraging the summarization capabilities of LLMs indeed leads to models that function as weak learners, and which can thus be integrated into boosting frameworks. This result leads to new potential paradigms for treating the results of LLM prompting not just as individual predictors, but as part of a larger set of meta-models as well.

References

Borisov, V., Leemann, T., Seßler, K., Haug, J., Pawelczyk, M., and Kasneci, G. Deep neural networks and tabular data: A survey. *arXiv preprint arXiv:2110.01889*, 2021.

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

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

Cui, G., Hu, S., Ding, N., Huang, L., and Liu, Z. Prototypical verbalizer for prompt-based few-shot tuning. *arXiv preprint arXiv:2203.09770*, 2022.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

Diao, S., Wang, P., Lin, Y., and Zhang, T. Active prompting with chain-of-thought for large language models. *arXiv preprint arXiv:2302.12246*, 2023.

Dinh, T., Zeng, Y., Zhang, R., Lin, Z., Rajput, S., Gira, M., Sohn, J.-y., Papailiopoulos, D., and Lee, K. Lift: Language-interfaced fine-tuning for non-language machine learning tasks. *arXiv preprint arXiv:2206.06565*, 2022.

Dua, D. and Graff, C. UCI machine learning repository, 2017. URL http://archive.ics.uci.edu/ml.

Freund, Y. and Schapire, R. E. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of computer and system sciences*, 55(1):119–139, 1997.

Friedman, J. H. Greedy function approximation: a gradient boosting machine. *Annals of statistics*, pp. 1189–1232, 2001.

Friedman, J. H. Stochastic gradient boosting. *Computational statistics & data analysis*, 38(4):367–378, 2002.

Gorishniy, Y., Rubachev, I., Khrulkov, V., and Babenko, A. Revisiting deep learning models for tabular data. In Ranzato, M., Beygelzimer, A., Dauphin, Y., Liang, P., and Vaughan, J. W. (eds.), *Advances in Neural Information Processing Systems*, volume 34, pp. 18932–18943. Curran Associates, Inc., 2021. URL https://proceedings.neurips.cc/paper/2021/file/9d86d83f925f2149e9edb0ac3b49229c-Paper.pdf.

Haluptzok, P., Bowers, M., and Kalai, A. T. Language models can teach themselves to program better. *arXiv preprint arXiv:2207.14502*, 2022.

Hao, Y., Sun, Y., Dong, L., Han, Z., Gu, Y., and Wei, F. Structured prompting: Scaling in-context learning to 1,000 examples. *arXiv preprint arXiv:2212.06713*, 2022.

Herzig, J., Nowak, P. K., Müller, T., Piccinno, F., and Eisenschlos, J. M. Tapas: Weakly supervised table parsing via pre-training. *arXiv preprint arXiv:2004.02349*, 2020.

Ho, N., Schmid, L., and Yun, S.-Y. Large language models are reasoning teachers. *arXiv preprint arXiv:2212.10071*, 2022.

Hollmann, N., Müller, S., Eggensperger, K., and Hutter, F. Tabpfn: A transformer that solves small tabular classification problems in a second. *arXiv preprint arXiv:2207.01848*, 2022.

Hou, B., O'Connor, J., Andreas, J., Chang, S., and Zhang, Y. Promptboosting: Black-box text classification with ten forward passes. *arXiv preprint arXiv:2212.09257*, 2022.

Hu, S., Ding, N., Wang, H., Liu, Z., Li, J., and Sun, M. Knowledgeable prompt-tuning: Incorporating knowledge into prompt verbalizer for text classification. *arXiv preprint arXiv:2108.02035*, 2021.

Huang, J., Gu, S. S., Hou, L., Wu, Y., Wang, X., Yu, H., and Han, J. Large language models can self-improve. *arXiv preprint arXiv:2210.11610*, 2022.

Koczkodaj, W. W. Somerville happiness survey data set. https://archive.ics.uci.edu/ml/datasets/Somerville+Happiness+Survey, 2018.

Kojima, T., Gu, S. S., Reid, M., Matsuo, Y., and Iwasawa, Y. Large language models are zero-shot reasoners. *arXiv preprint arXiv:2205.11916*, 2022.

Lester, B., Al-Rfou, R., and Constant, N. The power of scale for parameter-efficient prompt tuning. *arXiv preprint arXiv:2104.08691*, 2021.

Lewkowycz, A., Andreassen, A., Dohan, D., Dyer, E., Michalewski, H., Ramasesh, V., Slone, A., Anil, C., Schlag, I., Gutman-Solo, T., et al. Solving quantitative reasoning problems with language models. *arXiv preprint arXiv:2206.14858*, 2022.

Liu, P., Yuan, W., Fu, J., Jiang, Z., Hayashi, H., and Neubig, G. Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. *ACM Computing Surveys*, 55(9):1–35, 2023.

Liu, X., Zheng, Y., Du, Z., Ding, M., Qian, Y., Yang, Z., and Tang, J. Gpt understands, too. *arXiv preprint arXiv:2103.10385*, 2021.

Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., and Stoyanov, V. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019.

Narayan, A., Chami, I., Orr, L., and Ré, C. Can foundation models wrangle your data? *arXiv preprint arXiv:2205.09911*, 2022.

Nielsen, F. *Introduction to HPC with MPI for Data Science*. Springer, 2016.

Ouyang, L., Wu, J., Jiang, X., Almeida, D., Wainwright, C., Mishkin, P., Zhang, C., Agarwal, S., Slama, K., Ray, A., et al. Training language models to follow instructions with human feedback. *Advances in Neural Information Processing Systems*, 35:27730–27744, 2022.

Qin, G. and Eisner, J. Learning how to ask: Querying lms with mixtures of soft prompts. *arXiv preprint arXiv:2104.06599*, 2021.

Rakotonirina, N. C., Dessì, R., Petroni, F., Riedel, S., and Baroni, M. Can discrete information extraction prompts generalize across language models? *arXiv preprint arXiv:2302.09865*, 2023.

Reynolds, L. and McDonell, K. Prompt programming for large language models: Beyond the few-shot paradigm. In *Extended Abstracts of the 2021 CHI Conference on Human Factors in Computing Systems*, pp. 1–7, 2021.

Saha, S., Yu, X. V., Bansal, M., Pasunuru, R., and Celikyilmaz, A. Murmur: Modular multi-step reasoning for semi-structured data-to-text generation. *arXiv preprint arXiv:2212.08607*, 2022.

Schäfl, B., Gruber, L., Bitto-Nemling, A., and Hochreiter, S. Hopular: Modern hopfield networks for tabular data. *arXiv preprint arXiv:2206.00664*, 2022.

Schapire, R. E. The strength of weak learnability. *Machine learning*, 5:197–227, 1990.

Shin, T., Razeghi, Y., Logan IV, R. L., Wallace, E., and Singh, S. Autoprompt: Eliciting knowledge from language models with automatically generated prompts. *arXiv preprint arXiv:2010.15980*, 2020.

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

Vanschoren, J., Van Rijn, J. N., Bischl, B., and Torgo, L. Openml: networked science in machine learning. *ACM SIGKDD Explorations Newsletter*, 15(2):49–60, 2014.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Wang, H., Xu, C., and McAuley, J. Automatic multi-label prompting: Simple and interpretable few-shot classification. *arXiv preprint arXiv:2204.06305*, 2022a.

Wang, X., Wei, J., Schuurmans, D., Le, Q., Chi, E., and Zhou, D. Self-consistency improves chain of thought reasoning in language models. *arXiv preprint arXiv:2203.11171*, 2022b.

Wang, Z., Zhang, Z., Lee, C.-Y., Zhang, H., Sun, R., Ren, X., Su, G., Perot, V., Dy, J., and Pfister, T. Learning to prompt for continual learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 139–149, 2022c.

Wei, J., Wang, X., Schuurmans, D., Bosma, M., Chi, E., Le, Q., and Zhou, D. Chain of thought prompting elicits reasoning in large language models. *arXiv preprint arXiv:2201.11903*, 2022.

Yin, P., Neubig, G., Yih, W.-t., and Riedel, S. Tabert: Pretraining for joint understanding of textual and tabular data. *arXiv preprint arXiv:2005.08314*, 2020.

Yu, W., Iter, D., Wang, S., Xu, Y., Ju, M., Sanyal, S., Zhu, C., Zeng, M., and Jiang, M. Generate rather than retrieve: Large language models are strong context generators. *arXiv preprint arXiv:2209.10063*, 2022.

Zhang, Z., Zhang, A., Li, M., and Smola, A. Automatic chain of thought prompting in large language models. *arXiv preprint arXiv:2210.03493*, 2022.

Zhou, H., Nova, A., Larochelle, H., Courville, A., Neyshabur, B., and Sedghi, H. Teaching algorithmic reasoning via in-context learning. *arXiv preprint arXiv:2211.09066*, 2022a.

Zhou, Y., Muresanu, A. I., Han, Z., Paster, K., Pitis, S., Chan, H., and Ba, J. Large language models are human-level prompt engineers. *arXiv preprint arXiv:2211.01910*, 2022b.

A. Related Works

Deep Learning for Tabular Data. Tabular data represents samples as a collection of discrete or continuous attributes (Borisov et al., 2021). Due to their flexibility, tabular data are ubiquitous, but they lack the inherent structure found in images or text which makes applying deep learning to them challenging. They are often domain-specific and may have a relatively small number of data points. As a result, traditional deep learning methods, which thrive on large datasets and high-dimensional data, have seen limited success when applied to tabular data (Gorishniy et al., 2021; Shwartz-Ziv & Armon, 2022).

Recently, however, there has been increasing interest in applying deep learning to tasks related to tables such as data integration, imputation (Narayan et al., 2022), semantic parsing, and even running SQL queries (Herzig et al., 2020; Yin et al., 2020). Deep learning models have also been successful at learning tabular data classification by optimizing loss functions (Hollmann et al., 2022; Schäfl et al., 2022; Dinh et al., 2022). Unlike these approaches, we study how we can use LLM for classifying tabular data *without* finetuning or building a new language model. Since many tabular data can be grounded in natural language, texts are in fact a *natural representation* for tabular data. Motivated by the observation that LLMs can convert tables to text through prompting alone (Saha et al., 2022), we utilize LLMs to do this conversion. After the conversion, our classification algorithm also interacts with existing LLMs strictly through prompts. This creates an abstraction between the underlying language model and the learning procedure which may be desirable for various applications since access to the gradients or parameter updates are not required.

Prompting Prompting (Liu et al., 2023) refers to providing initial text or instructions to guide the response of a language model. The advancements in Language Model-based Learning (LLM) have unveiled new capabilities, such as chain of thought reasoning (Wei et al., 2022), zero-shot reasoning (Kojima et al., 2022), compositional problem solving (Zhou et al., 2022a), and self-improvement (Huang et al., 2022; Ho et al., 2022; Haluptzok et al., 2022). As a result, prompting has gained widespread application across various Natural Language Processing (NLP) tasks, including arithmetic, commonsense reasoning (Wang et al., 2022b), among others (Brown et al., 2020). While prompts offer flexibility, it is crucial to note that LLMs interpret them differently from humans. Therefore, the process of *prompt tuning*, which involves carefully engineering prompts, becomes essential for obtaining accurate and relevant outputs (Reynolds & McDonell, 2021). At its core, prompt tuning is an optimization process that aims to find the best prompt for a certain downstream task. Though a long line of works propose gradient-guided search to optimize "continuous prompt" instead of the language tokens (Liu et al., 2021; Qin & Eisner, 2021; Lester et al., 2021; Shin et al., 2020; Rakotonirina et al., 2023; Wang et al., 2022c; Diao et al., 2023), gradient-based updates can be limiting, as LLMs become bigger and the access to these models become increasingly API-based. Our approach aligns more with discrete search methods based on the fact that LLMs can automatically generate prompts for themselves (Zhou et al., 2022b; Zhang et al., 2022; Yu et al., 2022). Specifically, we prompt the LLM to summarize the tabular dataset. The summary in turn acts as a prompt that the LLM uses to make predictions as it encodes knowledge of the dataset. A sequence of such prompts summarizing different subsets of the data can be seen as weak learners for a boosting procedure.

Boosting Boosting (Schapire, 1990; Freund & Schapire, 1997) is a widely used technique to improve the accuracy of a model by combining weak learners (models that perform slightly better than random guessing) to make a strong learner (model with high accuracy). Common boosting algorithms include AdaBoost (Freund & Schapire, 1997), Gradient Boosting (Friedman, 2001), and Stochastic Gradient Boosting (Friedman, 2002); the XGBoost library (Chen & Guestrin, 2016) in particular is a commonly used implementation of gradient boosting. In concurrent work most relevant to ours, Hou et al. (2022) integrate LLM into AdaBoost for natural language inference (NLI) tasks, by training an MLP projection of the final hidden state of a special token. Our proposed method diverges from theirs in two key aspects. Firstly, our method avoids the overhead of learning additional parameters, is gradient-free, and does not require access to the model's internal states. Secondly, instead of storing knowledge in parameters, our approach concentrates on condensing knowledge into an intermediary representation referred to as "summary." This alternative strategy enhances interpretability and strictly learns through prompts, rendering it particularly suitable for small tabular data, where the prior knowledge in LLM can significantly benefit the learning process.

B. The Illustrated Data to Text Conversion

The process of converting tabular data to text is depicted in Figure 2. We ask the LLM to generate a description of the record by giving it some information about the dataset (which is generally available as metadata for tabular datasets) and a textual

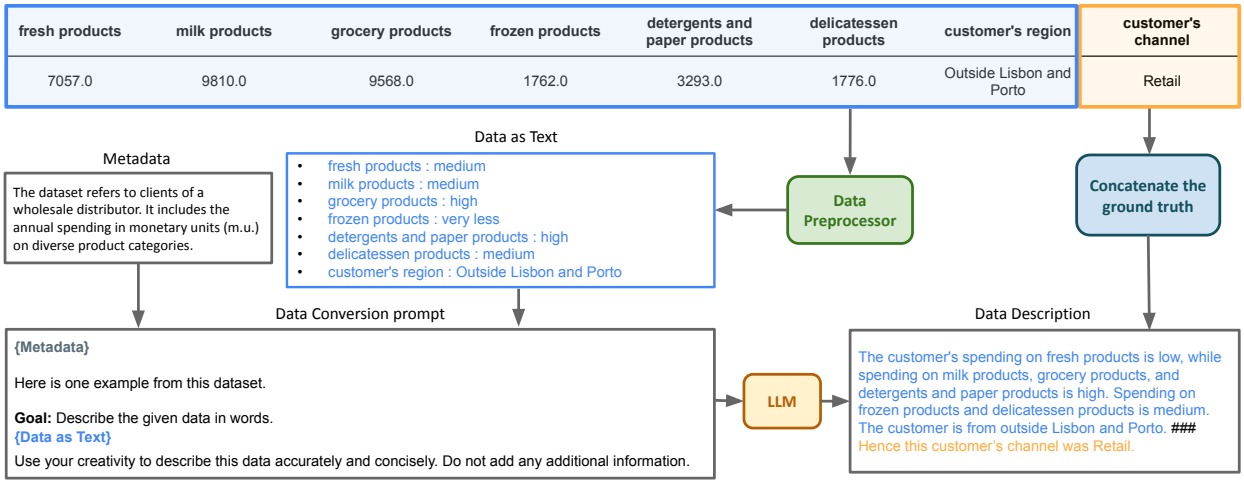

Figure 2. The conversion for a data point on the Wholesale customers dataset (OpenML ID 1511).

representation of the record (e.g., parsed JSON). Specifically, to ensure examples can serve as both training data and query inputs, we extract the descriptions of the features and concatenate them with the target label using a separator token. We will further discuss the rationale behind the design of this prompt under the "Failure modes" section (Section E.1).

Infact, we find that descriptions generated by LLM this way often perform better than those from a template. This ablation study can be found in Section C.

C. Ablations

Summarization forms the core of our methodology for generating weak learner. Consequently, it becomes important to identify the ideal setting that can induce high-quality summaries. We perform ablation studies over the `Summary` method, to decide hyperparameters for getting a good weak learner.

Table 2. Test errors for prompting-based methods on all datasets (↓), along with the error bars. Error represents one standard deviation. (acronym and notation in Table 1)

Dataset	Data Type	Size	Zero-shot	Few-shot	Summary	Summary Boosting
cae (42901)	1c4d	80	0.425± 0.04	0.388± 0.02	0.350± 0.04	0.300± 0.04
iris (61)	4c0d	150	0.680± 0.02	0.460± 0.01	0.275± 0.07	0.193± 0.03
tae (48)	1c4d	151	0.556± 0.07	0.494± 0.01	0.474± 0.02	0.454± 0.03
glass (41)	9c0d	214	0.486± 0.01	0.473± 0.01	0.466± 0.02	0.370± 0.02
bc (13)	7c5d	277	0.754± 0.02	0.516± 0.02	0.337± 0.02	0.288± 0.02
ve (678)	3c0d	111	0.522± 0.01	0.366± 0.01	0.304± 0.02	0.268± 0.03
ac (535)	2c2d	100	0.200± 0.00	0.200± 0.00	0.170± 0.01	0.170± 0.01
wine (43571)	13c0d	178	0.820± 0.03	0.674± 0.02	0.475± 0.01	0.320± 0.01
btc (1464)	4c0d	748	0.544± 0.01	0.430± 0.00	0.258± 0.04	0.240± 0.04
shs (Koczkodaj, 2018)	0c7d	143	0.416± 0.03	0.385± 0.03	0.422± 0.02	0.350± 0.02
vehicle (54)	18c0d	846	0.765± 0.00	0.560± 0.01	0.510± 0.02	0.410± 0.04
stath (Dua & Graff, 2017)	6c7d	270	0.551± 0.01	0.528± 0.01	0.444± 0.05	0.430± 0.01
vc (1524)	6c0d	310	0.714± 0.03	0.435± 0.06	0.327± 0.01	0.262± 0.01
ecoli (1011)	7c0d	336	0.581± 0.02	0.562± 0.01	0.480± 0.01	0.270± 0.03
hs (43)	3c0d	306	0.308± 0.02	0.262± 0.01	0.277± 0.01	0.250± 0.01
dia (37)	8c0d	768	0.446± 0.04	0.400± 0.00	0.360± 0.01	0.344± 0.01
hams (708)	5c0d	73	0.464± 0.03	0.481± 0.05	0.360± 0.02	0.207± 0.00
wc (1511)	6c1d	440	0.364± 0.01	0.347± 0.01	0.349± 0.02	0.330± 0.00

Preprocessing of continuous attributes. We tried several encoding techniques for continuous features, including binning, percentiles, and standard deviations. We chose the approach of describing them in technical language terms as well as

Table 3. Test errors for chosen methods on all datasets (↓). (acronym and notation in Table 1)

Dataset	Data Type	Size	Summary Boosting	LIFT	KNN	TabPFN	Xgboost
cae (42901)	1c4d	80	0.300± 0.04	0.312± 0.02	0.300± 0.00	0.425± 0.07	0.412± 0.05
iris (61)	4c0d	150	0.193± 0.03	0.100± 0.01	0.106± 0.02	0.027± 0.00	0.054± 0.04
tae (48)	1c4d	151	0.454± 0.03	0.480± 0.04	0.532± 0.01	0.450± 0.13	0.464± 0.01
glass (41)	9c0d	214	0.370± 0.02	0.218± 0.02	0.294± 0.03	0.158± 0.05	0.254± 0.05
bc (13)	7c5d	277	0.288± 0.02	0.318± 0.01	0.277± 0.02	0.264± 0.01	0.270± 0.01
ve (678)	3c0d	111	0.268± 0.03	0.430± 0.04	0.308± 0.01	0.370± 0.04	0.279± 0.02
ac (535)	2c2d	100	0.170± 0.01	0.180± 0.06	0.170± 0.01	0.090± 0.01	0.110± 0.04
wine (43571)	13c0d	178	0.320± 0.01	0.065± 0.01	0.214± 0.05	0.040± 0.01	0.040± 0.01
btc (1464)	4c0d	748	0.240± 0.04	0.270± 0.01	0.238± 0.00	0.209± 0.01	0.219± 0.01
shs (Koczkodaj, 2018)	0c7d	143	0.350± 0.02	0.419± 0.02	0.326± 0.03	0.392± 0.00	0.406± 0.00
vehicle (54)	18c0d	846	0.410± 0.04	0.111± 0.16	0.636± 0.01	0.178± 0.01	0.260± 0.00
stath (Dua & Graff, 2017)	6c7d	270	0.430± 0.01	0.122± 0.17	0.244± 0.03	0.148± 0.03	0.215± 0.00
vc (1524)	6c0d	310	0.262± 0.01	0.192± 0.03	0.318± 0.02	0.135± 0.00	0.187± 0.04
ecoli (1011)	7c0d	336	0.270± 0.03	0.126± 0.03	0.211± 0.03	0.036± 0.02	0.066± 0.01
hs (43)	3c0d	306	0.250± 0.01	0.314± 0.03	0.278± 0.00	0.262± 0.02	0.281± 0.02
dia (37)	8c0d	768	0.344± 0.01	0.324± 0.04	0.353± 0.02	0.238± 0.03	0.234± 0.00
hams (708)	5c0d	73	0.207± 0.00	0.334± 0.08	0.528± 0.02	0.328± 0.01	0.411± 0.01
wc (1511)	6c1d	440	0.330± 0.00	0.125± 0.04	0.043± 0.00	0.088± 0.00	0.098± 0.02

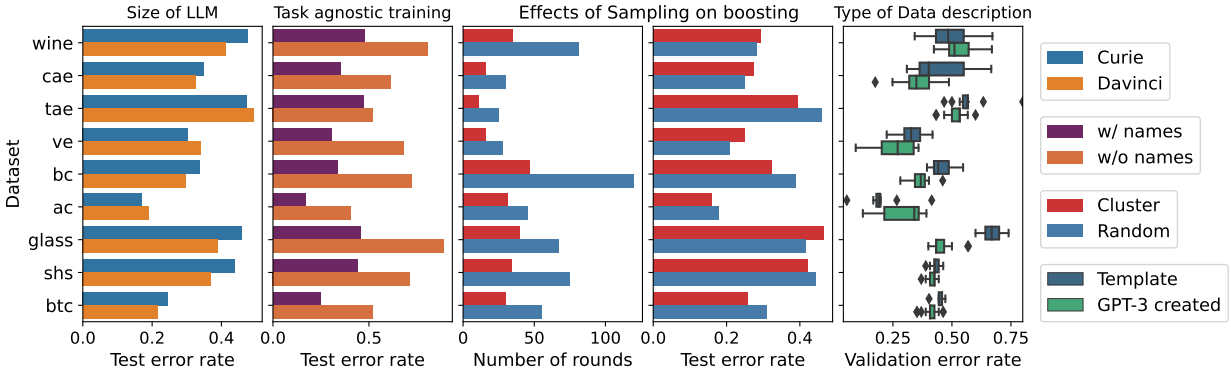

Figure 3. Results of ablation experiments compared for the `Summarization` method. The dataset acronymns can be referred from 1. **Left:** Performance of `curie` vs `davinci`. **Second from left**: Performance of models that can see task-specific attribute names vs models that see attribute names. **Middle**: Effect of sampling strategy cluster vs. random sampling on the number of rounds till convergence for each dataset **Second from Right**: the final test error for each sampling. **Right:** Performance of templatized data descriptions vs LLM generated data descriptions.

assigning quantifiers for each level, as illustrated in Figure 4 (right top). We observed that binning with quantifiers such as "low", "medium", and "high" was most effective for comparing examples and generating high-quality summaries. After hyperparameter tuning, we identified that using 5 bins provides sufficient granularity to distinguish variations in the continuous values. More details can be found in Section E.6.

Does the LLM explore prior knowledge to infer? To demonstrate the LLM's utilization of prior knowledge, we conduct an ablation study by masking the attribute names and using a template *"This example has features f1 = {}, f2 = {} and so on."* Figure 3 (second from left) shows the result. Using true variable names in the data descriptions leads to superior few-shot learning performance compared to using dummy names. This confirms that the model indeed leverages its prior knowledge of variables for predictions.

How does model size affect the performance? A natural question to ask is how the model size affects the downstream performance. We compare the `Summary` performances of GPT-3-`davinci` (175B parameters) and GPT-3-`curie` (13B parameters) on 5 datasets in Figure 3 (left). Surprisingly, we find that the larger model (`davinci`) does not consistently improve upon the smaller model. We also compare ChatGPT in Section E.12 and discuss the effects of RLHF (Ouyang et al., 2022) on tabular data.

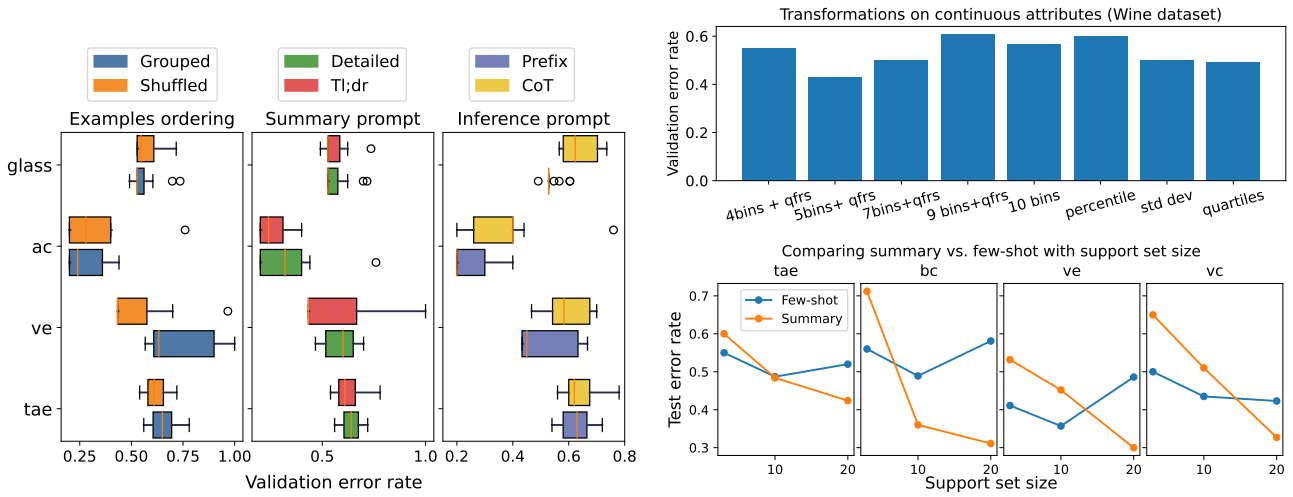

Figure 4. Additional ablations for the `Summary` method. **(Left)** Prompt design choices. First plot shows the effect of shuffling examples vs presenting them by class. Center plot compares `tldr` vs a more explicit prompt for inducing summary. Last plot compares prompts for doing inference. **(Right Top)** Performance of methods for discretizing continuous attributes on the *Wine* dataset. **(Right Bottom)** Performance of `Few-shot` and `Summary` as a function of number of examples in the context.

How does the performance scale with more examples? In Figure 4 (right bottom), we study how the behavior of `Few-shot` and `Summary` change with different support set sizes (i.e., the number of data descriptions that are summarized). `Few-shot` performance reaches an optimal size around the medium context length and degrades with more examples. In contrast, `Summary` improves with more data, which is the more desirable behavior.

Ordering of examples. Unlike conventional machine learning models, the ordering of examples in summarization affects the generation of hypotheses. There are two approaches: 1. presenting descriptions randomly (`shuffled`), and 2. grouping descriptions by classes (`grouped`). In Figure 4 (left), we compare these approaches on 4 datasets with `Summary` and find no significant difference. We use `shuffled` for all other experiments.

Different summary and inference prompts. The LLM easily generates concise summaries on standard datasets like `iris` using a simple "`tl;dr`" prompt, but requires a more explicit prompt on complex datasets like "vertebra-column". Comparing their performance in Figure 4 (left), both prompting modes are equally effective, so detailed prompts were mostly used in other experiments. See Table 4 for the complete list of the summary directives used. In the same Figure 4, we also compare two strategies for inference - prefix prompt (e.g. "This flower will be classified as"), and two-stage chain-of-thought prompting (e.g. "Let's think step by step") (Kojima et al., 2022). We observe no statistically significant difference between them[3]. Since the prefix prompt more often completes the query well under lesser compute, we use prefix prompt for all the other experiments.

Texts generated from template vs. GPT In Figure 3 (right), we see that using GPT-generated data descriptions consistently achieves better results. Refer section E.9 for examples of these templates. This may be due to the fact that the data description generated by LLMs conforms to natural text distribution more closely, which is desirable for performing inference using LLMs.

Effect of cluster sampling. Cluster sampling improves the performance of the LLM by selecting a representative set of texts that generalize better, reducing validation error faster compared to random sampling during boosting. Although it may require more resampling to achieve a weak learner, cluster sampling converges much faster than random sampling as we see in Figure 3 - middle and second from right. However, with sufficient boosting rounds, the performances of the two sampling methods become statistically similar.

[3]We do observe that CoT performs better with larger models.

D. Limitations

Although the weak learning approach we develop here shows promise, there are currently several drawbacks. Although summarization and boosting alleviate manual prompt tuning to large extent, we still had to minimally tune some parts of the pipeline to get ideal performance (see Table 4). Additionally, when the dataset contains many continuous attributes, there is a non-trivial gap between `Summary Boosting` and the other methods such as XGBoost or finetuning. Finally, the max input length of GPT-3 makes it harder to generate good summaries with just subset sampling. Eventually on larger datasets, after a certain number of boosting rounds, the summaries derived from a subset of examples may not further decrease the weighted error across all training examples. Rigorous techniques such as structured prompting circumvent this issue by rescaling attention weights (Hao et al., 2022). We believe this issue could be solved with more powerful LLMs such as GPT-4.

E. Failure Modes

The design of the prompt plays a pivotal role in our entire process. Specifying instructions precisely can create a significant difference, whether it comes to effectively describing tabular data, generating reliable summaries or inferring accurate predictions (shown in Figures 2 and 1). While soft prompting has been successful at instructing LLMs (Lester et al., 2021), it cannot be applied to our setting because our classification algorithm learns a human-level summary as a prompt for classifying data, rather than a soft prompt. Instead we chose to write prompts that ask the LLM to perform these tasks. In this way, our prompting method is entirely gradient-free. Hand-written prompts also offer flexibility, aligning well with our core methodology for creating weak learner. While carefully handcrafting prompts this way might seem intensive, we will show that once we identify the ideal hyperparameter settings they can be framed with little effort.

E.1. Challenges in Data Conversion

Language models (LLMs) have demonstrated impressive performance on standard tasks with minimal supervision (Wang et al., 2022b; Brown et al., 2020). However, for converting tabular data to text, there were several considerations to meet the requirements of our task. As highlighted in Section 2.1, we will refer to these texts as *data descriptions*.

Ensuring Uniform Length Firstly, the data descriptions should not be too long or short, also be of comparable length. Excessively long descriptions limit the number of examples that can be fit inside the prompt and summarized. We observed in Figure 4 (right bottom) that the summary performance also scales with more examples, so it makes sense to have descriptions of approximately uniform length.

A straightforward way of achieving this uniformity would be by specifying a max word length as part of the conversion prompt itself, as in "Describe in not more than 80 words". However, we found this approach can falter, sometimes leading to overly simplistic descriptions like "These are annual spendings of a customer." (in the `wholesale-customers` dataset).

Consequently, we adopt more nuanced strategy by first modifying the prompt with the terms "concisely" and "accurately" to emphasize the brevity and preciseness of the generated descriptions (shown in Figure 2). Then, we implement a resampling strategy, that generates descriptions until finding the one with a desired length ranging between 20 to 80 words. This process achieves consistent and uniformly long descriptions.

Including Metadata Prepending metadata to the prompt enhances the contextual awareness of the task, resulting in higher-quality descriptions (shown in Figure 2).

Separating Features from Labels In our method, the data descriptions function dual role, both as training examples and as query for inferring class labels. This suggests that, when converting data to text, the features need to be described separately from the target label as illustrated in Figure 2. The resulting strings are then concatenated to form the data description. Instead, if the whole tabular record were passed to the LLM, it often produces texts that assimilate the classification label information in the meat of the description itself, rendering it difficult to extract a query for doing inference.

While one might cleverly come up with prompts that can allow the LLM to describe the features and target label in separate sentences, we found it to be more sensible to supply just the features for describing and not reveal any information about the target task. Sometimes that can liberate the LLM to hallucinate some facts about the task and form biased data to begin with.

Natural-Sounding Descriptions While the LLM generates a different-styled response every time, to explicitly ensure that the generated descriptions are not template-like by chance, add a directive at the end of the prompt: "Use your creativity". This encourages the LLM to produce more natural narratives of the record. Alternatively, setting a higher temperature during decoding achieves a similar effect.

E.2. Summarization

There are several aspects worth considering that can contribute to high-quality summaries.

Sampling candidate summaries A well-crafted summary is a one that captures salient information of the dataset, in a way that facilitates inferring predictions off it. However, the process of generating summary using a LLM is inherently stochastic due to temperature sampling, as a result, the generated summary can be noisy. From our experiments with tuning this temperature, we found 0.80 to be ideal through Bayesian optimization. Even at this value, on average only 1 out of 3 summaries were meaningful.

A noisy summary can be distinguished quite easily. For instance, on the `vehicle` dataset, the `tl;dr` prompt elicits summaries as naive as "The given data describes a bus, van, or saab silhouette." or "The data in this table identifies a vehicle as a bus, saab, opel, or van. The compactness, circularity, distance circularity, radius ratio, hollows ratio, and symmetry are all predictive of the vehicle's type." which does not offer actionable insight.

This observation indicates that summaries need to be sampled quite a few times and the best one can be determined based on the validation error. As a result, for the `Summary` learning procedure in Section 2.2, we resample approximately 25 times to find a good summary. Also, given that our datasets are small, it is not unusual for the summaries to have the same validation error. When tied, we pick one having a higher training error rate, i.e. lower generalization gap.

Differently, in our `Summary boosting` procedure explained in Section 2.3, we resample only until finding a summary whose training error is better than random guessing and return immediately.

Ordering examples inside the *summarization* prompt Unlike gradient descent, prompting is not robust to the presentation of the examples to the learning algorithm. While we show via ablation studies in Section C that there is no statistically significant difference in performance between either shuffling the examples or listing them by class, we can generally expect that depending on the dataset and the number of target classes, one might be preferred over the other.

For instance, in a multi-class setting, listing examples by class might be more helpful in reaching a weak learner quickly. However, in a two-class setting, the summary might actually benefit from the randomness in shuffled examples.

Customizing the *summarization* prompt The approach of asking the LLM to summarize examples can also give rise to good/bad summaries. For instance, one can prompt the LLM with a simple `tl;dr` or specify the task more elaborately. We will refer to the latter option as `explicit`. As we demonstrated in Figure 4 (left), both are means to the goal and do not statistically differ in terms of performance induced.

However, in our experiments on certain datasets, we would rather be incentivized choosing the `explicit` over the `tl;dr` to attain a weak learner more quickly. This choice becomes important purely for compute reasons as it will take relatively lesser resampling, while the `tl;dr` still works. For instance, this scenario can happen when the LLM cannot decipher what the summary is supposed to say, by just observing the examples. As examples, the `tl;dr` prompt suffices on datasets such as `iris`, `diabetes`, and `wine` that are commonly encountered in prediction context, whereas the LLM might not be very familar with the goals of `vertebra-column` or `somerville-happiness` data, necessitating the use of the `explicit` prompt. For these other datasets, the fact that it is a classification problem based on some features and target classes may not be very apparent from just the examples and metadata. So, providing a directive such as "Summarize in detail how we can tell apart people with normal and abnormal vertebra-column" reduces ambiguity in the task setup and reduces probability of a noisy summary.

While manual intervention is necessary, framing this prompt can be done with little effort. We provide a comprehensive list of these parameters for all datasets in Table 4.

Including Metadata Similar to data conversion, including meta-data information in the prompt offers better penetration into the world of the dataset, as a result improves boosting performance.

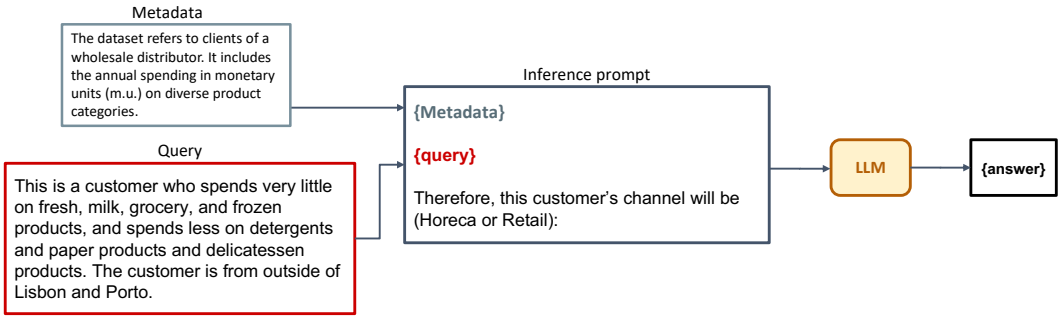

Figure 5. Steps of Zeroshot prompting

E.3. Inference

Mapping labels from LLM responses Answer mapping refers to the process of assigning the model's answer to a target output class. This step might be trivial when the answer is the class itself, for example when the LLM responds with "non-relapse" or "recurrence" to a query on the `breast-cancer` dataset. However, in other instances, it can become tricky when the LLM's responses are "will not recur" or "has higher chance of non-relapse than recurrence", requiring a more complex decoding logic to identify the target class.

Previous works have handled this problem by disguising the task as a Cloze prompt and learning a verbalizer, i.e. MLP projection of hidden state of the **[MASK]** token, that maps the predicted token to the target classes (Cui et al., 2022; Hu et al., 2021). By training a verbalizer, one can determinsically go from token vocabulary to the label space. There also exist unsupervised statistical techniques for achieving label mapping (Wang et al., 2022a).

In our method however, we strictly interact with the LLM through prompts and do not access the hidden state nor gradients. As a result, our inference process shown in Figure 1 focusses on inferring the class label solely through prefix prompts, without relying on learning an explicit mapping. Specifically, by conditioning on a suitable prefix, we constrain the LLM to return exactly the class label string. For example, the prefix "Therefore this iris flower is likely to be (setosa, versicolor, virginica):" works for the `iris` dataset. A key observation guiding the design of such a prefix prompt is the fact that specifying the output classes entices the LLM to predict from among these classes. With a rather plain prefix like "Predict what will be the type of this flower.", the LLM's answer space is unconstrained and it might liberally go on to explain a chain of reasoning such as "The flower has short petals and long sepals, hence it is versicolor, and not setosa." preventing a simple keyword search for the class label.

For a full list of these inference prompts, refer Table 4.

Two-step prompting, Davinci vs. Curie It is worth mentioning that a two-step prompting trick, by first calling "Lets think step by step" then concatenating the response with the prefix prompt also results in accurate answers, as we have shown in Figure 4 (left). However, it could only be implemented on the larger model `Davinci` but not `Curie` which is primarily used in our experiments. Interestingly `Davinci`'s chain of thought reasoning even outperforms its prefix prompt counterpart. In all our experiments with Curie however, the prefix technique works reasonably well.

The Davinci API also offers a suffix argument which can be invoked for predicting in a more natural way. For instance, for the `breast-cancer` dataset, the prompt can be posed with prefix "All in all, this woman is more likely to have a " and a suffix " of breast cancer." directly expecting the LLM to fill in with "recurrence" or "non-relapse."

E.4. Zeroshot Setting

We extend the analysis of prompting-based methods in Section 3.1 by further delving into the `Zeroshot` experiment. We illustrate this experimental setting in Figure 5. It only consists of the inference prompt, wherein the LLM is presented with the metadata and a query. To facilitate answer mapping, the output classes are also indicated in the prompt. Unlike `Summary`, the `zeroshot` process is not stochastic as there is no learning involved. For inference, the predicted tokens are sampled greedily at temperature $= 0$.

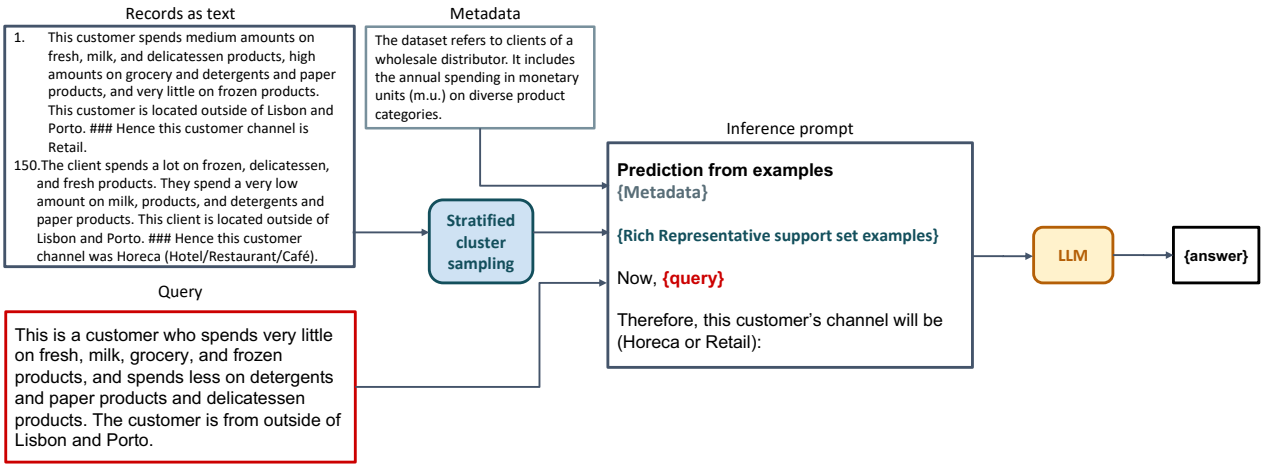

Figure 6. Workflow in fewshot prompting

E.5. Few-shot Setting

Extending the analysis from Section 3, we explain the `Fewshot` setting. As illustrated in Figure 6, it is an instance of in-context learning, wherein the support set examples are enlisted in the prompt along with a query in the end. The support set was chosen from the training set through the stratified cluster sampling outlined in Algorithm 1. This results in a semantically diverse collection of examples evenly spread across the classes. Since we observe in Figure 4 (bottom right) that the `Fewshot` performance drops with more examples, we choose approximately 15 examples to fit in the prompt. To preserve consistency, we standardize the number of fewshot examples approximately to this 15 for all datasets.

Similar to the `Summary` method, this inference prompt carries meta-information about the dataset, and also indicates the output classes. The prompt also injects support set examples that are stringed together as we show in Figure 6. The predictions are made greedily at a temperature of 0.

Again, the `Fewshot` method is a stochastic process whose outcome depends on the selection of the support set. So finding the ideal prompt, requires sampling different support sets from the training set quite a few times. We perform this resampling approximately 25 times and pick the best prompt based on the validation error.

E.6. Preprocessing continuous attributes

Extending from ablation studies in Section C, we demonstrate in Table 5 concrete examples of these encoding techniques applied to continuous features. Every numerical column in the dataset was subject to the transformation independently of the rest.

We applied several encoding techniques for continuous features, including binning, percentiles, and standard deviations. Our approach involved using technical language terms to describe these ranges, such as *"falls in the nth bin/nth percentile or n deviations above/below mean"*. We also characterize them in a more naturalistic way by assigning quantifiers such as *low*, *medium*, and *high* to each level in the binning technique.

To create effective textual descriptions, we examined three high-level approaches: 1. presenting only numerical values, 2. using solely textual encodings, and 3. concatenating both. We observed that utilizing textual encoding alone outperformed the other methods. As a result, we focused on mainly comparing textual encoding methods as shown in Figure 4 (right bottom). Through Bayesian optimization, we found that binning with "5" quantifiers was ideal for generating high-quality summaries.

We describe each encoding technique as follows:

- **Binning**: It involves creating a histogram with the given number of bins. As outputs, the values are directly described as *"falling in the n-th bin"* as illustrated in the `10 bins` experiment. However, in the presence of *degree quantifiers* which are categorical names assigned to the bins, these tags are used instead. We found that as opposed to calling out

the bin number, describing in terms of these quantifiers further aids the LLM in comparing the relative extent to which features match and improving the estimation of similarities. This led us to tune the number of bins against these degree quantifiers, selecting values in the range of 4, 5, 7, and 9 bins. The first four rows in Table 3 show how these tags get translated into the record.

- **Percentile**: It is given by computing the percentile rank of a value relative to that series of values. Then, the value is described as falling in that percentile rank in words. This is closer to representation of the true numerical values per se, but helps the LLM draw comparisons on a scale of 1-100.
- **Standard deviations**: In this procedure, the values are segmented into six ranges based on distance from the mean, given by one/two/three standard deviations above/below the mean.
- **Quartiles**: Here, we consider the first and third quartiles, and the median as landmarks to bucketize the values into four partitions.

Among these methods, the "5 bins with quantifiers" strikes a balance in granularity scale. It is not excessively fine-grained as "percentile", nor overly abstract, as the "4-bin" approach. This balance ultimately leads to optimal performance.

E.7. Clustering Sampling components

The procedure for stratified cluster sampling is outlined in Algorithm 1.

Algorithm 1 Cluster Sampling

Input: `X`, all training data; `y`, all training label; `R`, ratio of classes; `p`, AdaBoost weights of the current round; `s`, target number of samples. ▷ `R[k]` is the proportion of examples in class `k`.
`S` ← new empty set
`w` ← new array with same length as `X` filled with -1. ▷ `w[i]` is probability of sampling example i.
for `k` = 1 to number of target classes in `y` **do**
  `E` ← **GPTEmbedding**(`X[y == k]`) ▷ `E` refers to the embeddings of the data descriptions
  C ← **AgglomerativeClustering**(`E`). ▷ C_j is set of data indices present in the jth cluster.
  `c` ← new empty array same size as C. ▷ `c[j]` will store sampling probability of cluster j.
  for `j` = 1 to `len`(C) **do**
    `c[j]` ← $\frac{\texttt{len(X)}}{\texttt{len}(C_j)}$
  end for
  for `i` = 1 to `len(X)` **do**
    `w[i]` ← `c[j]`, such that, `i` $\in C_j$
  end for
  `w` ← **Normalize**(**Normalize**(`w`) × `p`) ▷ **Normalize** turns weights to a probability distribution.
  Sample `s` × `R[c]` examples from `X` using categorical distribution `w` and append to `S`.
end for
Return `S`

GPT-Embedding is OpenAI's text similarity model `text-embedding-ada-002` that takes a maximum input size of 8192 tokens. It returns a 1536-dimensional embedding for text. OpenAI recommends cosine distance for comparing *ada* embeddings in downstream tasks.

As a result, the **AgglomerativeClustering** algorithm applies hierarchical clustering over these features using cosine distance, average linkage and a heuristically selected distance threshold of 0.05. It yields a set of clusters C and each C_j contains a list of indices of data points that belong to that cluster j.

E.8. Summary Boosting Algorithm and other Optimizations

The idea of applying `adaboost` to summary prompts was discussed in Section 2.3. In practice, we perform several run-time optimizations to this boosting procedure. Thus we present the complete version of this procedure in Algorithm 2. In the algorithm, the **Summary** method summarizes the examples in the prompt via the process discussed in Section 2.2. Each summary can be treated as a hypothesis that can classify new data.

Algorithm 2 Summary Boosting

1: **Input**: X, all training data; y, all training label; T: maximum number of rounds; s: size of the sampled subset; R: ratio of classes. ▷ R[k] denotes the proportion of examples in class k.
2: h, P, ϵ, $\alpha \leftarrow$ empty arrays of length T.
3: N $\leftarrow$ len(X)
4: c $\leftarrow$ set of target classes
5: K $\leftarrow$ len(c)
6: w $\leftarrow$ new array of length N filled with $\frac{1}{\text{N}}$. ▷ w is the data distribution
7: **for** r = 1 to T **do**
8: $(X_s, y_s) \leftarrow$ **ClusterSampling**(X, y, R, w, s) ▷ sample s training examples from distribution w.
9: h[r] $\leftarrow$ **Summary** (X_s, y_s) ▷ h[r] is the weak learner in the current round
10: $\hat{y} \leftarrow h[r](X[i])$ ▷ $\hat{y}$ refers to predictions on training set
11: $\xi \leftarrow$ empty hashmap ▷ $\xi[p]$ will have error rate of the corresponding label mapping p
12: **for** p in **PermutedLabelMappings**(c) **do**
13: $\xi[p] \leftarrow \frac{\sum_{i=1}^{N} \text{w}[i] \times \mathbb{1}\{p[\hat{y}[i]] \neq y[i]\}}{\sum_{i=1}^{N} \text{w}[i]}$
14: **end for**
15: $p^* \leftarrow \arg\min_p \xi[p]$ ▷ p^* is the label mapping with the least training error
16: **if** $\xi[p^*] > 1 - \frac{1}{K} - \mu$ OR **AllSame**($\hat{y}$) **then** ▷ μ is a positive quantity indicating the increased performance threshold.
17: Goto Step 8.
18: **else**
19: P[r] $\leftarrow p^*$ ▷ P holds the label mapping applied to hypothesis h at round r
20: ϵ[r] $\leftarrow \xi[p^*]$ ▷ ϵ[r] is the weighted error at round r.
21: **end if**
22: **if** ϵ[r] == 0 **then**
23: Break
24: **end if**
25: α[r] $\leftarrow \log\left(\frac{1-\epsilon[\text{r}]}{\epsilon[\text{r}]}\right) + \log(\text{K} - 1)$ ▷ α[r] refers to coefficient of the hypothesis at round r.
26: **for** i = 1 to N **do**
27: $\text{w}[i] = \text{w}[i] \times \exp(\alpha[\text{r}]\mathbb{1}\{\text{P[r]}[h[\text{r}](X[i])] \neq y[i]\})$
28: **end for**
29: w $\leftarrow$ **Normalize**(w)
30: **end for**
31: **Return** h, α

- **Raising the bar for a weak learner:** Our goal is to produce good weak learners, i.e. high-quality summaries that can generalize well to unseen data points. Ideally the weak learners should reduce the validation error rate quickly and accelerate convergence of the boosting procedure. Thus we raise the performance threshold to a notch slightly higher than random guessing probability (see Step 15 in Algorithm 2), provoking more insightful summaries.

  We resample until finding a weak learner that satisfies this threshold.

  The positive quantity μ is a hyperparameter that typically takes values 0.08 for 2-class problem and 0.16 for 3-class problem, and so on.

  Although this step increases compute, it yields better weak learners and improves convergence overall.

- **Permuting the predicted class label assignments:**

  We harness the potential of permuting the class assignments by exploring $K!$ different mappings of predictions to classes using the **PermutedLabelMappings** function in steps 11-14. This process helps us identify the mapping that minimizes the training error to the greatest extent.

  By considering multiple permutations of predictions across the label space, as outlined in Steps 11-14 of Algorithm 2, we obtain a hashmap p from the **PermutedLabelMappings** function. This hashmap maps the predictions $\hat{y}$ to the permuted label space. Selecting the mapping that results in the lowest training error effectively diminishes the

cumulative training error during boosting iterations and proves to be an effective strategy for generating strong weak learners. This technique is particularly advantageous in scenarios involving more than two classes.

- **Sanity checks:** Finally, to ensure robustness of the weak learner when faced with skewed datasets, we have implemented a policy that disallows a naive all-ones classifier. The condition calling **AllSame** in Step 15 of Algorithm 2) performs this check.

E.9. Text Templates

In the ablation study which involves comparing the descriptions created manually vs. by a LLM, as illustrated in Figure 3 (right), we transform the descriptive attributes into the textual format by applying a pre-defined template. In Table 6 we provide examples of these templates for selected datasets.

E.10. Complexity Analysis

We provide the time complexity analysis comparing our boosting procedure to finetuning the LLM.

For finetuning, the complexity is $\mathcal{O}(TNf)$, where f is runtime of the LLM, T is number of epochs, N is the number of data points.

For summary boosting, the complexity is $\mathcal{O}(TRf)$, where f is runtime of the LLM, T is number of boosting rounds and R is the number of resampling per round.

Concretely, for a dataset with 175 examples, finetuning takes 20 epochs $\times$ 175 examples $\times$ 2 = 7000 passes through the LLM. 2 stands for both forward and backward passes through the model.

For the same dataset boosting requires 50 rounds $\times$ 25 resampling on average = 1250 passes through the LLM.

Thus, we believe the complexity of our algorithm is at least comparable to, if not better than, that of finetuning (without considering the cost of the actual API calls).

E.11. Estimating the cost of API calls

While our method is applicable to any large language model (LLM), we primarily conducted experiments using GPT-3. Each API call to GPT-3 incurs a specific dollar cost.

After analyzing the running time complexity of summary boosting, which is $\mathcal{O}(TRf)$, we can provide a rough estimation of the cost associated with training a classifier on any given dataset.

To begin, when making a call to summarize examples, the prompt is filled up to the maximum context length, which is 2048 tokens for the query prompt and completion. We'll refer to these summary tokens as $S_t = 2048$.

Additionally, if N represents the size of the dataset and we allocate (50 + 10)

Now, to obtain a weak learner at boosting round r, we may need to resample up to R candidate summaries. Furthermore, we calculate the training error for each candidate summary to determine if it performs better than random guessing. Once the desired weak learner is found, we compute the validation error for that round only once. Therefore, each round requires querying $R \times (S_t + 0.5N \times P_t) + 0.1N \times P_t$ tokens.

Considering that the maximum number of rounds is denoted as T, the total number of tokens exchanged would be $T \times [R \times (S_t + 0.5N \times P_t) + 0.1N \times P_t]$.

For instance, let's consider a dataset with 175 examples. In this case, the cost would be 30 rounds $\times$ [20 resampling $\times$ (2048 summary tokens + (0.5 $\times$ 175 training examples) $\times$ 210 prediction tokens) + (0.1 $\times$ 175 validation examples) $\times$ 210 prediction tokens] = 12364050 tokens, which approximately costs \$25 for Curie at a rate of \$0.002/1K tokens.

E.12. Can ChatGPT function as a weak learner?

One would expect that it is more advantageous to try newer LLMs such as ChatGPT that produce increasingly more human-like text and are far more sample-efficient, i.e. can summarize more examples since they come with a larger context length. To investigate this, we conduct experiments by feeding ChatGPT with the same tabular data descriptions and using identical prompts to create weak learners. The results are presented in Table 7.

Surprisingly ChatGPT outperforms Curie in classifying datasets with more numerical features, such as `wine`, `wholesale-customers`, and `iris`. This observation suggests that LLMs are becoming more adept at quantitative reasoning from finetuning with more data. However, the reinforcement learning from human feedback (RLHF) (Ouyang et al., 2022) poses a limitation as it still ensures the generated text does not deviate too much from its prior. The generated text distribution adheres closely to the behavior programmed into the LLM induced by optimizing with such a reward model. Consequently it becomes challenging to bias the LLM with adversarial examples that might occasionally emerge in the training set.

For example, ChatGPT does not mostly generalize well on datasets with medical information such as `verterbra-column`, `breast-cancer`, `caesarian` and `blood-transfusion-center` where there can be examples contrary to common medical beliefs. In these cases, the RLHF is more restrictive due to its conformity to human preferences and does not neutrally summarize examples at hand from a classification standpoint. However, boosting imposes a significantly higher penalty on examples that the model fails to classify correctly, causing ChatGPT to not decrease training error after a few epochs. While these models exhibit promise in terms of higher-order problem-solving skills, their capabilities can also be limited by their alignment with human preferences.

Table 4. **Prompt design:** Prompt parameter settings for every dataset.

Dataset	Prompting hyperparameters
caesarian	*metadata:* This dataset contains information about caesarian section results of 80 pregnant women with the most important characteristics of delivery problems in the medical field.The goal is to predict whether a woman will undergo normal or caesarian delivery. *classes:* [normal, caesarian] *summary directive:* Tl;dr *inference directive:* Hence this woman's delivery mode is likely to be (normal or caesarian):
iris	*metadata:* This is the iris dataset, perhaps the best known database to be found in the pattern recognition literature. Fisher's paper is a classic in the field and is referenced frequently to this day. (See Duda & Hart, for example.) The data set contains 3 classes of 50 instances each, where each class refers to a type of iris plant. One class is linearly separable from the other 2; the latter are NOT linearly separable from each other.Predicted attribute- class of iris plant- setosa, versicolor, virginica *classes:* [setosa, versicolor, virginica] *summary directive:* Tl;dr *inference directive:* Based on the above information, predict if this flower will be classified as setosa, versicolor, virginica
tae	*metadata:* The data consist of evaluations of teaching performance over three regular semesters and two summer semesters of 151 teaching assistant (TA) assignments at the Statistics Department of the University of Wisconsin-Madison. The scores were divided into 3 roughly equal-sized categories ("low", "medium", and "high") to form the class variable. *classes:* [low, medium, high] *summary directive:* Tl;dr *inference directive:* Predict whether this class will score low or medium or high:
glass	*metadata:* This is the glass dataset from USA Forensic Science Service; 6 types of glass; defined in terms of their oxide content (i.e. Na, Fe, K, etc). The study of classification of types of glass was motivated by criminological investigation. At the scene of the crime, the glass left can be used as evidence...if it is correctly identified! *classes:* [building_windows_float_processed, building_windows_non_float_processed, vehicle_windows_float_processed, containers, tableware, headlamps] *summary directive:* Tl;dr *inference directive:* There are 6 possible type of glass: building_windows_float_processed, building_windows_non_float_processed, vehicle_windows_float_processed, containers, tableware, headlamps. Predict which one will this sample be:
breast-cancer	*metadata:* This is one of three domains provided by the Oncology Institute that has repeatedly appeared in the machine learning literature. This data set includes 201 instances of one class and 85 instances of another class. The instances are described by 9 attributes, some of which are linear and some are nominal. It contains information about women that had a recurrence or non-relapse of breast cancer after their first time. *classes:* [recurrence, non-relapse] *summary directive:* Based on the above examples, figure out under what conditions will a woman have recurrence or non-relapse of breast cancer? *inference directive:* Predict whether this woman will have a recurrence or non-relapse:
visualizing-environmental	*metadata:* This is the visualizing-environmental dataset, one of the 22 data sets from the book Visualizing Data published by Hobart Press (books@hobart.com). This data describes indicators for a positive/negative environment based on ozone, radiation and temperature. *classes:* [positive, negative] *summary directive:* Tl;dr *inference directive:* There are clear signs of this environment being (positive or negative):
analcatdata-chlamydia	*metadata:* This chlamydia dataset is one of the data sets used in the book "Analyzing Categorical Data" by Jeffrey S. Simonoff, Springer-Verlag, New York, 2003. It contains results of individuals that tested for chlamydia. *classes:* [positive, negative] *summary directive:* Tl;dr *inference directive:* Predict if this person will test positive or negative for chlamydia:
wine	*metadata:* This is the Wine recognition data. Updated Sept 21, 1998 by C.Blake. It contains results of a chemical analysis of wines grown in the same region in Italy but derived from three different cultivars. The analysis determined the quantities of 13 constituents found in each of the three types of wines. *classes:* [1, 2, 3] *summary directive:* Using these examples and based on the contents of constituents, summarize what distinguishes wines of type 1 or 2 or 3? *inference directive:* Hence this wine will be classified as ->type
blood-transfusion-center	*metadata:* Data taken from the Blood Transfusion Service Center in Hsin-Chu City in Taiwan - this is a classification problem. The goal is to predict whether a given individual will consent or avoid donating blood. *classes:* [consent, avoid] *summary directive:* Tl;dr *inference directive:* Therefore, this individual is likely to (avoid/consent):
somerville-happiness-survey	*metadata:* This is the Somerville Happiness Survey Data Set. It has ratings collected from a survey of Somerville residents. From the responses of a resident, the goal is to predict whether they feel happy or unhappy about the place. *classes:* [unhappy, happy] *summary directive:* Based on the Somerville happiness survey, how can we predict whether a resident is happy or unhappy with their place? *inference directive:* So this resident is (happy or unhappy):
vehicle	*metadata:* This is the Statlog (Vehicle Silhouettes) Data Set. The purpose is to classify a given silhouette as one of four types of vehicle - bus, saab, opel or a van, using a set of features extracted from the silhouette. The vehicle may be viewed from one of many different angles. *classes:* [bus, saab, opel, van] *summary directive:* Using these examples, summarize how can we differentiate if a silhouette is that of a bus, saab, opel or a van. *inference directive:* Out of saab, bus, van and opel, this vehicle is likely to be a
statlog-heart	*metadata:* This dataset is a heart disease database similar to a database already present in the repository (Heart Disease databases) but in a slightly different form. It has data on individuals having and not having heart disease. *classes:* [present, absent] *summary directive:* Differentiate people with heart disease present from ones absent. *inference directive:* In this case, heart disease is likely to be (present/absent):
verterbra-column	*metadata:* This dataset contains values for six biomechanical features used to classify orthopaedic patients into 3 classes (normal, disk hernia or spondilolysthesis) or 2 classes (normal or abnormal). Biomedical data set built by Dr. Henrique da Mota during a medical residence period in the Group of Applied Research in Orthopaedics (GARO) of the Centre MÃ©dico-Chirurgical de RÃ©adaptation des Massues, Lyon, France. The task is to classify patients as belonging to one out of two categories: Normal (100 patients) or Abnormal (210 patients). *classes:* [abnormal, normal] *summary directive:* Based on the above examples, summarize how will you distinguish patients that have normal vs. abnormal vertebral column. *inference directive:* Therefore, this individual's vertebral column is likely to be (abnormal or normal):

ecoli	*metadata:* This data contains protein localization sites. Reference: "A Knowledge Base for Predicting Protein Localization Sites in Eukaryotic Cells", Kenta Nakai & Minoru Kanehisa, Genomics 14:897-911, 1992. *classes:* [1, 2] *summary directive:* Using these examples, how can we tell apart cells with protein localized in sites 1 and 2? *inference directive:* Hence protein localization will be at site ->
haberman-survival	*metadata:* The dataset contains cases from a study that was conducted between 1958 and 1970 at the University of Chicago's Billings Hospital on the survival of patients who had undergone surgery for breast cancer. *classes:* [survived, died] *summary directive:* Based on these examples, figure out what commonalities are predictive of patients surviving more than 5 years and less. *inference directive:* So, 5 years down the line, this person (survived/died):
diabetes	*metadata:* This dataset is originally from the National Institute of Diabetes and Digestive and Kidney Diseases. The objective is to predict based on diagnostic measurements whether a patient has high/low risk of developing diabetes. *classes:* [low, high] *summary directive:* Based on these examples, distinguish patients having low vs. high risk of diabetes. *inference directive:* Based on the reasoning, this patient is likely to have a (low/high):
visualizing-hamster	*metadata:* This is the visualizing-hamster dataset contains 22 data sets from the book Visualizing Data published by Hobart Press (books@hobart.com). It contains examples of hamsters that are ill and healthy. *classes:* [ill, healthy] *summary directive:* Using these examples, identify predictive indicators of ill and healthy hamsters. *inference directive:* Predict whether this hamster will be ill or healthy:
wholesale-customes	*metadata:* The data set refers to clients of a wholesale distributor. It includes the annual spending in monetary units (m.u.) on diverse product categories. This data gives information about spending patterns and region of operations of Retail and Horeca (Hotel/Restaurant/Café) customers of the wholesale distributor. *classes:* [retail, horeca] *summary directive:* Using these examples, summarize how can we differentiate Retail customers and Horeca customers. *inference directive:* Therefore, which one of Retail or Horeca this customer is likely to be:

Table 5. Continuous variable transformations applied to an example from the `wholesale-customers` dataset. The raw tabular record is as follows: spending on fresh products: 6353.0, spending on milk products: 8808.0, spending on grocery products: 7684.0, spending on frozen products: 2405.0, spending on detergents and paper products: 3516.0, spending on delicatessen products: 7844.0 and customer's region: Outside Lisbon and Porto.

Method	Data Representation	Example as text
`4 bins + quantifiers` {very low, low, high, very high}	- *spending on fresh products :* low - *spending on milk products :* very high - *spending on grocery products :* high - *spending on frozen products :* high - *spending on detergents and paper products :* high - *spending on delicatessen products :* very high - *customer's region :* Outside Lisbon and Porto	This customer spends low amounts on fresh products, very high amounts on milk products, high amounts on grocery products, frozen products, detergents and paper products, and very high amounts on delicatessen products. They are located outside of Lisbon and Porto.
`5 bins + quantifiers` {very low, low, medium, high, very high}	- *spending on fresh products :* medium - *spending on milk products :* very high - *spending on grocery products :* high - *spending on frozen products :* high - *spending on detergents and paper products :* high - *spending on delicatessen products :* very high - *customer's region :* Outside Lisbon and Porto	This customer from outside Lisbon and Porto spends medium on fresh products, very high on milk products, high on grocery products, high on frozen products, high on detergents and paper products, and very high on delicatessen products.
`7 bins + quantifiers` {extremely low, very low, low, medium, high, very high, extremely high}	- *spending on fresh products :* low - *spending on milk products :* very high - *spending on grocery products :* high - *spending on frozen products :* high - *spending on detergents and paper products :* very high - *spending on delicatessen products :* extremely high - *customer's region :* Outside Lisbon and Porto	This customer situated outside Lisbon and Porto spends low on fresh products, very high on milk products, high on grocery products, high on frozen products, very high on detergents and paper products, and extremely high on delicatessen products.
`9 bins + quantifiers` {lowest, extremely low, very low, low, medium, high, very high, extremely high, highest}	- *spending on fresh products :* low - *spending on milk products :* extremely high - *spending on grocery products :* high - *spending on frozen products :* high - *spending on detergents and paper products :* very high - *spending on delicatessen products :* highest - *customer's region :* Outside Lisbon and Porto	This customer spends low amounts on fresh products, extremely high amounts on milk products, high amounts on grocery products, frozen products, detergents and paper products, and highest amounts on delicatessen products. They are located outside Lisbon and Porto.
`10 bins`	- *spending on fresh products :* falls in the first out of ten bins of values - *spending on milk products :* falls in the second out of ten bins of values - *spending on grocery products :* falls in the first out of ten bins of values - *spending on frozen products :* falls in the first out of ten bins of values - *spending on detergents and paper products :* falls in the first out of ten bins of values - *spending on delicatessen products :* falls in the second out of ten bins of values - *customer's region :* Outside Lisbon and Porto	This customer spends relatively little on fresh, grocery, frozen and detergents/paper products, and more on milk and delicatessen products. They are based outside Lisbon and Porto.
`Percentile`	- *spending on fresh products :* falls in the forty-first percentile - *spending on milk products :* falls in the eighty-second percentile - *spending on grocery products :* falls in the sixty-fifth percentile - *spending on frozen products :* falls in the sixty-third percentile - *spending on detergents and paper products :* falls in the seventy-second percentile - *spending on delicatessen products :* falls in the ninety-eighth percentile - *customer's region :* Outside Lisbon and Porto	This customer has an annual spending of 41st percentile on fresh products, 82nd percentile on milk products, 65th percentile on grocery products, 63rd percentile on frozen products, 72nd percentile on detergents and paper products, and 98th percentile on delicatessen products, and is located outside of Lisbon and Porto.
`Standard deviation`	- *spending on fresh products :* is within one std-dev below the mean value - *spending on milk products :* is within one std-dev above the mean value - *spending on grocery products :* is within one std-dev below the mean value - *spending on frozen products :* is within one std-dev below the mean value - *spending on detergents and paper products :* is within one std-dev above the mean value - *spending on delicatessen products :* is two std-dev above the mean value - *customer's region :* Outside Lisbon and Porto	The customer has annual spending on fresh products, milk products, grocery products, frozen products, detergents and paper products, and delicatessen products within one standard deviation of the mean, except for delicatessen products which is two standard deviations above the mean. The customer is located outside Lisbon and Porto.
`Quartiles`	- *spending on fresh products :* is between the first quartile and median values - *spending on milk products :* is more than the third quartile value - *spending on grocery products :* is between median and third quartile values - *spending on frozen products :* is between median and third quartile values - *spending on detergents and paper products :* is between median and third quartile values - *spending on delicatessen products :* is more than the third quartile value - *customer's region :* Outside Lisbon and Porto	This customer spends more than the third quartile value on milk, delicatessen and detergents and paper products. The customer's spending on fresh, grocery, and frozen products falls between the median and third quartile values, while the customer is located outside of Lisbon and Porto.

Table 6. **Templatized descriptions:** Templates used to format examples for the ablation study between LLM-created data descriptions vs. template descriptions

Dataset	Descriptive attribute values	Template
caesarian	*age:* [very young, young, middle-aged, old, very old] *delivery_number:* [first, second, third, fourth, fifth] *delivery_time:* [timely, premature, latecomer] *blood_pressure:* [low, normal, high] *heart_problem:* [has, doesn't have] *delivery_mode:* [normal, caesarian]	This {*age*} woman is in her {*delivery_number*} delivery and it is {*delivery_time*}. She has a {*blood_pressure*} blood pressure and {*heart_problem*} heart problems. ### Based on these attributes, this woman is likely to deliver by {*delivery_mode*}
iris	*sepal_length*, *petal_length*: [very short, short, medium length, long, very long] *sepal_width*, *petal_width*: [very narrow, narrow, medium width, wide, very wide] *flower_type:* [setosa, versicolor, virginica]	This iris flower has {*sepal_length*} and {sepal_width} sepals. It also has {*petal_length*} and {*petal_width*} petals. ### Hence this flower is a {*flower_type*}
vertebral-column	*pelvic_incidence*, *pelvic_tilt*, *lumbar_lordosis_angle*, *sacral_slope*, *pelvic_radius, grade_of_spondylolisthesis:* [very low, low, medium, high, very high] *result:* [normal, abnormal]	This patient has a {*pelvic_incidence*} pelvic incidence, {*pelvic_tilt*} pelvic tilt, and {*lumbar_lordosis_angle*} lumbar lordosis angle, {*sacral_slope*} sacral slope, {*pelvic_radius*} pelvic radius and {*grade_of_spondylolisthesis*} grade of spondylolisthesis. ## As a result, the patient's vertebral-column is likely to be {*result*}
statlog-heart	age: [very young, young, middle-aged, old, very old] *sex:* [male, female] *chest_pain_type:* [asymptomatic, nonanginal pain, atypical angina, typical angina] *bp, cholesterol, st_depression, heart_rate, num_major_vessels:* [very low, low, medium, high, very high] *fasting_blood_sugar:* [high, low] *electrocardiographic_results:* [having left ventricular hypertrophy, normal, having ST-T wave abnormality] *slope_st_segment:* [flat, upsloping, downsloping] *exercise_induced_angina:* [has, do not have] *defect_type:* normal, reversible, fixed *presence_of_heart_disease:* [present, absent]	This individual is a/an {*age*} {*sex*} with {*chest_pain_type*} chest pain, {*bp*} resting blood pressure, and {*cholesterol*} serum cholesterol. Their fasting blood sugar {*fasting_blood_sugar*} >120 mg/dl, they are {*electrocardiographic_results*} and a {*heart_rate*} maximum heart rate. They {*exercise_induced_angina*} exercise-induced angina, and have a {*st_depression*} ST depression induced by exercise relative to rest. Their peak exercise ST segment has a {*slope_st_segment*} slope, and they have a {*num_major_vessels*} number of major vessels. The defect type is {*defect_type*}. ### Hence heart disease is likely to be {*presence_of_heart_disease*}.
haberman-survival	*age_at_time_of_op:* [very young, young, middle-aged, old, very old] *year_of_op:* [1964, 1962, 1965, 1959, 1958, 1960, 1966, 1961, 1967, 1963, 1969,1968] *num_pos_axillary_nodes:* [very low, low, medium, high, very high] *survival_status:* [survived, died]	This patient was {*age_at_time_of_op*} at the time of operation in {*year_of_op*}. They had a {*num_pos_axillary_nodes*} number of positive axillary nodes detected. ### Therefore 5 years down the line, the patient {*survival_status*}

Table 7. Comparing test error rate of `Summary Boosting` backended by Curie and ChatGPT on all datasets (↓). Refer to the caption of Table 1 for the notations.

Dataset	**Data Type**	**Size**	`Curie`	`ChatGPT`
caesarian [cae] (42901)	`1c4d`	80	$0.300_{\pm 0.04}$	$0.406_{\pm 0.03}$
iris (61)	`4c0d`	150	$0.193_{\pm 0.03}$	$0.083_{\pm 0.01}$
tae (48)	`1c4d`	151	$0.454_{\pm 0.03}$	$0.443_{\pm 0.04}$
glass (41)	`9c0d`	214	$0.370_{\pm 0.02}$	$0.492_{\pm 0.02}$
breast-cancer [bc] (13)	`7c5d`	277	$0.288_{\pm 0.02}$	$0.360_{\pm 0.01}$
visualizing-environmental [ve] (678)	`3c0d`	111	$0.268_{\pm 0.03}$	$0.333_{\pm 0.04}$
analcatdata-chlamydia [ac] (535)	`2c2d`	100	$0.170_{\pm 0.01}$	$0.300_{\pm 0.06}$
wine (43571)	`13c0d`	178	$0.320_{\pm 0.01}$	$0.250_{\pm 0.01}$
blood-transfusion-center [btc] (1464)	`4c0d`	748	$0.240_{\pm 0.04}$	$0.433_{\pm 0.01}$
somerville-happiness-survey [shs] (Koczkodaj, 2018)	`0c7d`	143	$0.350_{\pm 0.02}$	$0.430_{\pm 0.02}$
vehicle (54)	`18c0d`	846	$0.410_{\pm 0.04}$	$0.350_{\pm 0.16}$
statlog-heart [stath] (Dua & Graff, 2017)	`6c7d`	270	$0.430_{\pm 0.01}$	$0.370_{\pm 0.17}$
verterbra-column [vc] (1524)	`6c0d`	310	$0.262_{\pm 0.01}$	$0.669_{\pm 0.03}$
ecoli (1011)	`7c0d`	336	$0.270_{\pm 0.03}$	$0.193_{\pm 0.03}$
haberman-survival [hs] (43)	`3c0d`	306	$0.250_{\pm 0.01}$	$0.415_{\pm 0.03}$
diabetes [dia] (37)	`8c0d`	768	$0.344_{\pm 0.01}$	$0.297_{\pm 0.04}$
visualizing-hamster [hams] (708)	`5c0d`	73	$0.207_{\pm 0.00}$	$0.400_{\pm 0.08}$
wholesale-customers [wc] (1511)	`6c1d`	440	$0.330_{\pm 0.00}$	$0.199_{\pm 0.04}$