Sequential Model Optimization in the era of Large Language Models

Abstract

Active learning, a semi-supervised learning approach, tackles labeled data scarcity by focusing on the most informative data points. Traditionally, Sequential Model Optimization (SMO), a Bayesian optimization technique, has been key in guiding data point selection for labeling. However, large language models (LLMs) have emerged, showing impressive generalization in minimal or zero-shot scenarios across various domains. This paper investigates whether LLMs can outperform traditional SMO methods in active learning, particularly for tabular datasets where LLMs are less commonly applied. We explore three research questions: (1) Can LLMs outperform Bayesian optimization in active learning with tabular data? (2) How do LLMs compare to Bayesian methods in runtime efficiency? (3) Does few-shot learning surpass zero-shot learning in this context? Our findings reveal that SMO continues to outperform LLMs in active learning, especially with tabular data, where LLMs, though capable, are hindered by significantly longer runtimes. Additionally, few-shot learning outperforms zero-shot learning when LLMs encounter data outside their training corpus. This study offers a novel analysis of LLM performance in active learning for tabular datasets, introduces a method of using LLMs as an oracle in multi-objective optimization, and provides a comparative assessment against state-of-the-art SMO. A reproduction package is available online for further exploration and validation.

Keywords

Do, Not, Us, This, Code, Put, the, Correct, Terms, for, Your, Paper

1 Introduction

Active learning is a semi-supervised learning approach where the model actively selects new data points to label, guided by optimization methods designed to maximize the impact of the labeled data. This method effectively addresses the challenge of labeled data scarcity by focusing on the most informative or uncertain data points, thereby reducing the amount of labeled data needed to achieve strong model performance.

Traditionally, guided optimization in active learning has been handled through Sequential Model Optimization (SMO), a technique rooted in Bayesian optimization. SMO selects new data points based on calculated likelihoods and prior probabilities, making it an efficient approach for identifying the most valuable data to label. However, this method was developed in a context where models had limited capacity to generalize from prior data and lacked the ability to effectively query new samples based on existing information.

The advent of large language models (LLMs) has shifted this landscape, as these models demonstrate impressive capabilities in generalizing from minimal examples or even in zero-shot scenarios across various domains. Given their ability to infer and generate new information with limited data, it is crucial to explore whether LLMs can surpass Bayesian methods in the realm of multi-objective optimization. Testing LLMs against traditional Bayesian approaches could reveal new insights into their potential to outperform existing methods, particularly in settings where data is scarce or highly complex. This exploration could pave the way for more advanced and adaptive active learning strategies that leverage the strengths of both LLMs and traditional optimization techniques.

The research of this paper began with the question *can few shot learning outperform sequential model optimization?*. Few shot lot learning is known to perform well in Natural Language Understanding tasks which are predominantly text based, However most of the domains where active learning is practiced involves tabular data, with a high number of columns containing numercial data. As shown in the literature review of this paper a large number of papers have contributed on the use of LLMs for active learning methods that involve text based datasets However in this paper we propose a unique method to assess the potential use of LLMs in active learning involving tabular datasets.

To assess, we ask three questions:

- **RQ1:**Are LLMs better than bayesian optimization functions for Active learning involving tabular datasets?

  Result
  Sequential Model Optimization (SMO) remains the state-of-the-art in active learning, outperforming language models despite their strong generalization capabilities.
- **RQ2:**Do language models have very high run times compared to the bayesian counterparts?

  Result
  Language models take significantly longer to run, especially in few-shot settings, compared to acquisition functions, indicating their meticulous search for similar rows in the dataset.
- **RQ3:** Does Few shot learning perform better than zero shot learning in the context of active learning?

  Result
  Large language models generally perform well without examples, but in cases where the data wasn't in their training, few-shot learning outperforms zero-shot learning.

In summary the contributions of this paper are:

- An important analysis on the performance of LLMS in active learning for tabular datasets which was missing in prior studies [].
- An unique way of using LLMs as an oracle in multi-objective optimization in tabular datasets.
- Comparision between the performance of LLMS with the state of the art SMO in active learning.

- a reproduction package which could be cloned from *Module*

The results of this paper are based on the specific LLMs used in this work. It is well known that each large language model is designed for a particular purpose—whether for general instruction (GPT), code generation (Code Llama), mathematical tasks (Minerva), etc.And their performance depends on numerous factors like parameter count, training data, and more. Given these differences and the limited resources available, a comprehensive comparison of all models is unlikely

2 Related Work

2.1 Active Learning and Bayesian Optimization

Bayesian Optimization is a powerful method for optimizing expensive-to-evaluate functions, especially when dealing with black-box functions where the internal workings are unknown or too complex to model directly. It is predominantly used in areas to maximize an objective function given the prior evidence and evidence to get the posterior function (based on "Bayes theorem"). Examples include hyperparameter optimization in machine learning models, optimizing the design of engineering systems, or even finding the best combination of ingredients in a chemical process and etc.

$$P(M|E) \propto P(E|M)P(M) \tag{1}$$

It is readily employed in active learning to label new rows based on previously available/labeled rows using a pool based sampling strategy. The rows should be categorized into either "best" or "rest". The Bayesian optimization function is used to find rows that are highly probable of belonging to either of these categories. This permits the selection of the next observation through either exploration (targeting the areas of uncertainty) or exploitation (targeting the areas that are most similar).

$$B = loglikes(best, r, d, 2) \tag{2}$$

$$R = loglikes(rest, r, d, 2) \tag{3}$$

- "best" are samples in the desired part of the total set.
- "rest" are samples other than best in the set.
- "r" is the current row for optimization
- "d" is the length of the done set.
- B and R are variables to represent the log-likelihoods of a row r belonging to best and rest, respectively.

Acquisition functions are used to guide the search for the rows to the optima of the objective function. It could either maximum of certain region or high uncertainty or a completely different objective. The acquisition functions used in the study are Explore, Exploit and Focus.

- `Explore`: Explore is used to find regions of space that are most uncertain with respect to currently available data. By doing so it ensures to capture data with the least confidence of all the regions. The formula for Explore is given by:

$$Explore = \frac{(B - R)}{(B + R) + \epsilon} \tag{4}$$

- `Exploit`: Exploit is used to target regions that have high likelihood of yeilding best regions from the data. Unlike Explore that targets regions of uncertainty, Exploit uses prior knowledge to navigate the regions that are likely to provide best samples. The formula for Exploit is given by:

$$Exploit = B \tag{5}$$

- `Focus`: Focus is an acquisition function that combines the principles of Explore and Exploit, adjusting dynamically as the process progresses. The idea is to start by exploring more when the budget is large and gradually shift towards exploitation as the budget diminishes, making the process more efficient. The formula for Focus is:

$$Focus = \left[\frac{(B+1)^{m_i} + (R+1)}{|B - R| + \epsilon} \right]_{i=0}^{n-1} \tag{6}$$

2.2 Sampling Methods

- `Random`: Random sampling strategy is used to pool random data from the samples which could be used a efficient baseline to compare with the rest of the strategies in this list.
- `Uncertainty`: Uncertainty-based sampling selects data points for labeling that the model is least certain about, typically those with the highest prediction entropy or lowest confidence. This approach aims to improve model performance by focusing on the most ambiguous cases.
- `Diversity`: Diversity-based sampling selects data points that are different from those already chosen or labeled, ensuring a wide variety of examples. This strategy helps the model learn from a broad spectrum of data, reducing the risk of over-fitting to specific patterns.
- `Similarity`: Similarity-based sampling selects data points that are most similar to the already labeled or selected examples. This method leverages the assumption that similar instances can provide additional useful information for refining the model's performance in familiar regions.

2.3 Active Learning and Language Models

Zero Shot and Few Shot learning with language models are extremely popular given the models ability to generalize through seen and unseen tasks. [20] presents a case by employing LLMs as a labeling oracle in active learning for text based datasets. As briefed in [] the major sampling strategies used in active learning are Uncertainty, Diversity and Similarity.

[20] uses a similarity based sampling strategy to actively label text based tasks through the use a small language model to label and a large language model to scrutinize the labeling which has yielded the highest performance among all the approaches.

2.4 Ranking Studies

A constant question among researchers is whether transformer based models can be readily employed in active learning. Studies from [6] have proven that transformers could only achieve limited performance compared to tree based models for tabular datasets which could be attributed to various reasons like converging on unimportant features etc. In July 2024 we searched scholar.google.com for the conjunction of "active learning" and "large language models" published after 2020.

"active learning" "large language models"

From that list, we sorted the top 100 pages of results by citations and found a knee in that curve at the 185 point. This reduced our population of papers down to 45. This list was then skimming the papers looking for any reference to:

- Comparison to non-LLMs
- Presentation of some acquisition function;
- Any techniques, other than random selection, for initializing the population of labelled examples;
- Testing on tabular data
- multi-objevtive optimization
- Testing on more than or equal to T datasets (T = 5)

This took us to 14 papers. After reading the titles and abstracts of those papers, and skimming the contents of the potentially interesting papers and snow balling through important papers, we found five other potentially interesting papers, which lead us to the 19 papers of Table 1.

The key observation made from the survey is that majority of the studies explored active learning using llms for text based datasets, However there was hardly any study that presented the capabilities of LLM few shot learning for active learning. Despite results from [6] proving that neural models struggle to achieve good results for tabular dataset. RQ1 addresses this issue and the remaining sections of these paper explore further on the few shot and zero shot capabilities of LLMS in handling tabular data.

3 Experimental Design

This experiment compares the performance of large language models, llama3(8B) and phi3-mini(3.8B) (under both zero shot and few shot learning strategies) with the state of the art Bayesian optimization based acquisition functions (Explore, Exploit and Focus) across 8 tabular regression datasets. These datasets are part of the 49 datasets used in https://github.com/timm/ezr. The description of these datasets could be found in table 2.

The performance along with the runtime of all the treatments are reported via statistical tests accross various budgets. Due to resource limitations, we have restricted the budget of both llms having a combination of budgets (20, 30) and both the models sharing a common value of 10 repeats per budget. Whereas the treatments based on acquisition functions are allowed with a wider range of budget (20, 25, 30, ..., 60) with 20 repeats per budget.

3.1 Algorithm for LLM Based Optimization

This code defines a few shot learner function that utilizes a simulated annealing approach for selecting and ranking data rows based on their "distance to heaven" (d2h). The learner function comprises three main sub-functions: *_ranked* for sorting rows by their d2h values, *llm_guesser* for using a language model to predict the best unlabeled row, and *_smo1* for iteratively selecting and refining the set of done rows. The primary goal is to improve the labeling of data by leveraging both distance-based ranking and model predictions.

3.2 Prompt Template

The model performance greatly depends on the input we provide it with. So inorder to exctract the highest performance from a language model the prompts must be carefully designed. In this work we have designed a generalized prompt template that could be applied to all the 8 datasets in **??** for both few shot and zero shot learning.

3.2.1 Few Shot Prompt. A few-shot prompt should be carefully structured to guide the model in understanding its role, the task at hand, and the expected response format. The prompt is divided into three key sections.

The first section is the System Message, where the model is instructed about its role and provided with meta-information about the task. This section sets the context, specifying what the model is expected to do, such as acting as an assistant, evaluator, or any other defined role. It also includes any relevant constraints or guidelines that the model should follow, such as the tone, length, or type of response required. The System Message helps ensure that the model's outputs are consistent with the desired outcome and that it adheres to any specific rules throughout the interaction.

The second section is the Examples Section, which contains a set of N examples that demonstrate the exact format of input and output expected during the task. These examples should be representative of the types of queries the model will encounter and should clearly illustrate the pattern that the model needs to learn. The examples serve as a guide, helping the model understand how to process the input and generate the correct output. By providing diverse yet consistent examples, this section helps the model generalize from the few provided cases to similar scenarios in future queries.

The final section is the Human Message, which contains the current query that needs to be addressed by the model. This query should follow the same format as the examples provided earlier to maintain consistency. The model uses the context and patterns learned from the System Message and Examples Section to generate an appropriate response to the query presented in the Human Message. This structure ensures that the model has all the necessary information to perform the task accurately, with a clear understanding of what is expected in its responses. By combining a well-defined role, illustrative examples, and a consistent query format, the few-shot prompt effectively guides the model in delivering precise and relevant outputs.

3.2.2 Zero Shot Prompt. A zero-shot prompt template is quite similar to a few-shot prompt template, but it omits the example section where the model is shown examples of input-output

Table 1: Highly cited Active Learning with Language Models studies

Ref	Year	Citations	A = Compare to Non-LLMs?	Data = Tabular?	Acquire	Initialization = Guided?	Multi-Objective Optimization?	T=5 Datasets?	No details on sample size?	Row counts (max=6)
here	2024	0	✓	✓	various	✓	✓	✓	✓	
[12]	2024	12	✓	✓	Similarity	✓	✓	✓	×	6
[1]	2024	0	✓	✓	Adaptive	×	✓	✓	×	5
[20]	2023	9	✓	×	Similarity	✓	✓	✓	×	5
[3]	2010	3237	×	✓	Uncertainty	×	✓	✓	×	4
[4]	2021	496	✓	×	Similarity	✓	✓	×	×	4
[19]	2022	450	✓	✓	-	×	✓	✓	×	4
[9]	2022	355	×	×	Diversity	✓	✓	✓	×	4
[18]	2022	130	×	×	Similarity	✓	✓	✓	×	4
[7]	2021	2035	-	✓	-	×	✓	✓	✓	3
[15]	2021	76	×	×	Uncertainty	×	✓	✓	×	3
[8]	2020	415	-	×	Diversity	×	✓	✓	✓	3
[21]	2020	173	×	×	Adaptive	×	✓	×	×	2
[2]	2022	268	×	×	-	✓	×	✓	✓	2
[17]	2023	250	×	×	-	✓	-	✓	✓	2
[11]	2023	257	×	×	-	✓	×	×	✓	1
[5]	2020	621	×	×	Similarity	×	×	×	×	1
[13]	2021	288	×	×	-	×	✓	×	✓	1
[10]	2024	0	×	×	Uncertainty	×	✓	×	×	1
[16]	2020	392	-	×	-	×	×	×	✓	0
[22]	2021	290	-	×	-	×	-	×	✓	0
[14]	2020	208	×	×	-	×	×	×	✓	0
Column count (max=21):			5	5		5	13	12	9	

Figure 1: Code for LLM Based optimization

```python
def learner(i:data):

    def llm_guesser(current: row, done: rows) -> row:
        cut = int(.5 + len(done) ** 0.5)
        best = clone(i, done[:cut]).rows
        rest = clone(i, done[cut:]).rows
        best = [b[:len(i.cols.x)] for b in best]
        rest = [r[:len(i.cols.x)] for r in rest]
        messages = load_prompt(args.dataset).getTemplate(best, rest, current[:len(i.cols.x)], cols = i.cols.x)
        prompt = model.tokenizer.apply_chat_template(messages, tokenize=False, add_generation_prompt=True)
        model.model.config.pad_token_id = model.model.config.eos_token_id
        outputs = model(prompt, max_new_tokens=256, do_sample=True, temperature=0.5, top_p=0.9)
        print(outputs[0]['generated_text']) if args.intermediate else None
        if "best" in outputs[0]['generated_text'][len(prompt):].lower(): return current
        return None

    def _smo1(todo:rows, done:rows) -> rows:
        "Guess the `top`  unlabeled row, add that to `done`, resort `done`, and repeat"
        count = 0
        for k in todo:
            count += 1
            if len(done) >= args.last: break
            top = llm_guesser(k, done)
            if(top == None): continue
            btw(d2h(i,top))
            done += [top]
            done = _ranked(done, top, count)
        return done

    i_sampled = random.choices(i.rows, k = k)
    return _smo1(i_sampled[args.label:], _ranked(i_sampled[:args.label]))
```

pairs. Instead, the model is directly instructed to perform the intended task based solely on the information provided in the System Message and the Human Message.

The System Message in a zero-shot prompt still plays a crucial role in setting the context for the model, defining its role, and providing any necessary meta-information about the task. This section ensures that the model understands the nature of the task, the expected format of the response, and any specific constraints or guidelines it should follow.

In a zero-shot prompt, since there are no examples to guide the model, the Human Message becomes the focal point. This section presents the query or task that the model needs to address. The Human Message should be clear, concise, and well-structured, as the model will rely entirely on this information to generate its response. Without prior examples, the

Table 2: Dataset Description

Dataset Name	#Rows	#Indep. vars	#Dep. vars
SS-A	1343	3	2
SS-B	206	3	2
SS-D	196	3	2
SS-X	86058	11	3
auto93	398	5	3
X264-Measurements	1152	16	1

Figure 2: Few shot prompt for LLM Based Optimization

"System message" : You are an excellent assistant.you need to evaluate the specifications of a car and answer in one word if the example falls into best or rest categories. Here are the attributes provided for each car in the same order: Clndrs, Volume, HpX, Model, origin.
"Examples"
"Input" : [4, 97, 46, 73, 2]
"Output": Best

"Input" : [4, 79, 67, 74, 2]
"Output": Best

"Input" : [8, 360, 175, 73, 1]
"Output": Rest

"Input" : [8, 440, 215, 70, 1]
"Output": Rest
"Human message" : [6, 350, 215, 70, 1]?

Figure 3: Zero shot prompt for LLM Based Optimization

"System message" : You are an excellent assistant.you need to evaluate the specifications of a car and answer in one word if the example falls into best or rest categories. Here are the attributes provided for each car in the same order: Clndrs, Volume, HpX, Model, origin.

"Human message" : Based on the above attributes, answer in one word if the following example is similar to best or rest [6, 350, 215, 70, 1].

model interprets the task based on the instructions given and produces an output that aligns with the specified requirements.

3.3 Statistical Analysis

When comparing the results of language models to the acquisition functions, we use a statistical significance test. Significance test are useful for detecting if two populations differ merely by random noise. Also, effect sizes are useful for checking that two populations differ by more than just a trivial amount. For the significance test, we used the Scott-Knott test. This technique recursively bi-clusters a sorted set of numbers. If any two clusters are statistically indistinguishable, Scott-Knott reports them both as one group. Along with the Skott-Knot test we also report the runtime of every treatment w.r.to the budget to compare the efficiency of all the treatments.

4 Results

RQ1: Are LLMs better than bayesian optimization functions for Active learning?

Sequential Model Optimization (SMO) is considered state-of-the-art in active learning, particularly when it comes to identifying the next best data point (or row) to label. This

Table 3: Scott-Knot test on Bayesian Optimization and LLM few-shot results for auto93.csv

Rank	Treatment	Budget	Mean	Std	Time(s)
0	exploit	50	0.17	0.14	1.96
0	exploit	55	0.17	0.02	2.22
0	exploit	60	0.17	0.09	2.47
0	Focus	60	0.17	0.03	4.86
0	exploit	30	0.19	0.06	1.02
0	exploit	35	0.19	0.12	1.24
1	exploit	40	0.19	0.00	1.47
1	exploit	45	0.19	0.14	1.71
1	Focus	45	0.19	0.10	2.98
1	Focus	55	0.19	0.09	4.14
1	Focus	50	0.19	0.09	3.54
1	Focus	40	0.20	0.07	2.47
2	random	55	0.20	0.06	0.00
3	phi3-mini(3.8B)	30	0.24	0.11	2260.39
4	exploit	20	0.24	0.12	0.60
4	random	60	0.24	0.08	0.00
4	random	40	0.26	0.08	0.00
5	random	45	0.26	0.11	0.00
5	random	50	0.26	0.13	0.00
5	Focus	25	0.26	0.08	1.21
5	random	30	0.26	0.05	0.00
5	Focus	30	0.26	0.09	1.59
5	random	35	0.26	0.08	0.00
5	phi3-mini(3.8B)	20	0.27	0.05	1090.29
5	llama3-8b(8.0B)	20	0.27	0.13	4580.62
5	random	25	0.27	0.08	0.00
5	Focus	35	0.27	0.12	2.01
5	explore	40	0.28	0.17	1.47
6	llama3-8b(8.0B)	30	0.30	0.05	6234.75
6	Focus	20	0.30	0.07	0.86
6	exploit	25	0.31	0.13	0.80
6	explore	35	0.31	0.03	1.24
6	explore	45	0.31	0.03	1.73
6	explore	50	0.31	0.08	1.97
6	explore	60	0.31	0.05	2.55
6	random	20	0.31	0.07	0.00
6	explore	55	0.32	0.05	2.23
6	explore	30	0.33	0.07	1.02
6	explore	25	0.33	0.10	0.81
7	explore	20	0.34	0.11	0.61
7	rrp	10	0.35	0.14	0.21
8	baseline	398	0.55	0.16	0.00

is accomplished through the use of various acquisition functions that guide the selection process. While language models demonstrate a strong capability in sequentially selecting optimal rows, leveraging both their pre-trained knowledge and their ability to generalize from a few examples, they still do not reach the performance levels achieved by SMO.

This performance gap is evident in the results presented in 3 and 4, where SMO consistently outperforms language models in this context. The structured approach of SMO, which systematically evaluates and selects the most informative data points, gives it an edge over language models, which, despite their flexibility and generalization capabilities, may not fully capture the specific intricacies required for optimal active learning performance.

RQ2: Do language models have very high run times? It can be observed in 3 and 4 that language models have an average runtime exceeding 1000 seconds for just 10 repeats, whereas the acquisition functions within Sequential Model Optimization complete their runs in just a few seconds for

Table 4: Scott-Knot test on Bayesian Optimization and LLM zero shot results for auto93.csv

Rank	Treatment	Budget	Mean	Std	Time(s)
0	exploit	50	0.17	0.02	1.36
0	exploit	55	0.17	0.02	1.53
1	exploit	60	0.17	0.02	1.71
1	Focus	60	0.17	0.02	3.27
1	exploit	45	0.19	0.00	1.19
2	Focus	45	0.19	0.09	2.08
3	exploit	40	0.19	0.12	1.05
4	Focus	55	0.19	0.09	2.85
5	random	60	0.20	0.05	0.00
5	exploit	35	0.24	0.13	0.87
5	random	40	0.24	0.07	0.00
5	random	55	0.24	0.06	0.00
6	random	50	0.25	0.03	0.00
7	Focus	50	0.26	0.08	2.44
8	Focus	35	0.26	0.11	1.46
8	random	35	0.27	0.06	0.00
9	random	20	0.27	0.07	0.00
9	Focus	25	0.27	0.05	0.85
9	random	45	0.28	0.06	0.00
9	random	30	0.30	0.15	0.00
9	Focus	40	0.30	0.06	1.75
9	exploit	20	0.31	0.09	0.43
9	random	25	0.31	0.06	0.00
9	exploit	30	0.31	0.12	0.72
9	Focus	30	0.31	0.04	1.13
9	explore	50	0.31	0.05	1.37
9	explore	45	0.31	0.04	1.20
10	explore	55	0.31	0.07	1.54
10	exploit	25	0.31	0.19	0.57
10	explore	40	0.31	0.05	1.04
10	Focus	20	0.32	0.02	0.61
10	explore	60	0.32	0.10	1.72
10	explore	25	0.33	0.13	0.58
10	explore	35	0.33	0.08	0.89
10	phi3-mini(3.8B)	30	0.33	0.10	2764.00
10	explore	20	0.35	0.12	0.43
11	phi3-mini(3.8B)	20	0.36	0.01	1860.56
11	rrp	10	0.38	0.11	0.15
11	explore	30	0.38	0.10	0.73
11	llama3-8b(8.0B)	20	0.42	0.09	9504.92
12	baseline	398	0.55	0.16	0.00
12	llama3-8b(8.0B)	30	0.58	0.16	13920.05

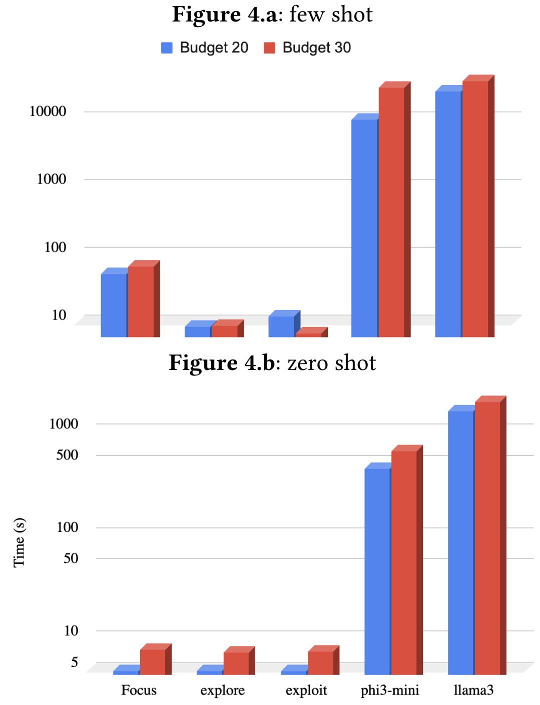
20 such repeats. This stark contrast highlights the computational efficiency of acquisition functions compared to language models.

Additionally, as depicted in 4 , language models operating in a few-shot setting require significantly more time than in a zero-shot setting. This suggests that LLMs engage in a more meticulous and thorough search for similar rows within the dataset when provided with a few examples. The extended processing time reflects their effort to leverage the examples effectively, ensuring they capture the nuances of the task, which contrasts with the more generalized approach in zero-shot scenarios.

RQ3: Does Few shot learning perform better than zero shot learning in the context of active learning?

Large language models (LLMs) are designed to deliver impressive results across a wide range of tasks without requiring ground knowledge or examples. This is largely due to the vast and diverse training corpora they are built upon, enabling them to generalize well to various unseen tasks. However, when the task involves datasets that were not part of the

Figure 4: Run time of all the treatments for budgets 20 & 30



model's training data, as in our experiments, few-shot learning tends to outperform zero-shot learning.

The underlying reason for this could be that few-shot learning provides the model with relevant context and examples, which help it adapt more effectively to the specific nuances of the new dataset. This contextual adaptation seems crucial when dealing with data that differ significantly from what the model has been exposed to during training. The results indicate that even a minimal amount of task-specific data can significantly enhance performance compared to zero-shot scenarios, where the model relies entirely on its pre-existing knowledge base.

5 Conclusion

The ranking studies of prior work on active learning using large language models (LLMs) have demonstrated that there has been comparatively little research focused on tabular datasets. This gap in the literature is significant, especially when considering the potential of LLMs in handling diverse data modalities.

Our experiments involved a thorough evaluation of two LLMs, phi3-mini (3.8B) and Llama3 (8B), across various treatments for the datasets outlined in Section 2. The performance of these models was benchmarked against traditional Bayesian optimization techniques. The results, as detailed in Section **??**, reveal that Bayesian optimization methods significantly outperform LLMs in both performance metrics and computational efficiency. This suggests that, for tabular datasets, the sophisticated statistical underpinnings of Bayesian methods

Table 5: Few-shot learning Results

	Statistical ranks										
Frequency	0	1	2	3	4	5	6	7	8	9	10
llama3-8b(8.0B)	17	17	17	17		33					
phi3-mini(3.8B)	17		33	17	17		17				
rrp	17			17		17		50			
exploit	83	17									
random	17		50	17	17						
baseline						33		50			17
Focus	67	17					17				
explore			17	17	67						
Evaluations											
llama3-8b(8.0B)	20 (0)	30 (0)	20 (0)	20 (0)	0 (0)	20 (0)	0 (0)	0 (0)	0 (0)	0 (0)	0 (0)
phi3-mini(3.8B)	30 (0)	0 (0)	25 (0)	30 (0)	30 (0)	0 (0)	0 (0)	20 (0)	0 (0)	0 (0)	0 (0)
rrp	17 (0)	0 (0)	0 (0)	9 (0)	0 (0)	11 (0)	0 (0)	10 (0)	0 (0)	0 (0)	0 (0)
exploit	36 (0)	25 (0)	0 (0)	0 (0)	0 (0)	0 (0)	0 (0)	0 (0)	0 (0)	0 (0)	0 (0)
random	20 (0)	0 (0)	50 (0)	30 (0)	55 (0)	0 (0)	0 (0)	0 (0)	0 (0)	0 (0)	0 (0)
baseline	0 (0)	0 (0)	0 (0)	0 (0)	0 (0)	0 (0)	43605 (0)	0 (0)	646 (0)	0 (0)	206 (0)
Focus	41 (0)	40 (0)	0 (0)	0 (0)	0 (0)	0 (0)	45 (0)	0 (0)	0 (0)	0 (0)	0 (0)
explore	0 (0)	0 (0)	0 (0)	40 (0)	60 (0)	38 (0)	0 (0)	0 (0)	0 (0)	0 (0)	0 (0)
Deltas											
llama3-8b(8.0B)	49 (0)	34 (0)	86 (0)	89 (0)	0 (0)	63 (0)	0 (0)	0 (0)	0 (0)	0 (0)	0 (0)
phi3-mini(3.8B)	47 (0)	0 (0)	67 (0)	56 (0)	80 (0)	0 (0)	0 (0)	66 (0)	0 (0)	0 (0)	0 (0)
rrp	51 (0)	0 (0)	0 (0)	89 (0)	0 (0)	71 (0)	0 (0)	34 (0)	0 (0)	0 (0)	0 (0)
exploit	83 (0)	34 (0)	0 (0)	0 (0)	0 (0)	0 (0)	0 (0)	0 (0)	0 (0)	0 (0)	0 (0)
random	42 (0)	0 (0)	78 (0)	30 (0)	89 (0)	0 (0)	0 (0)	0 (0)	0 (0)	0 (0)	0 (0)
baseline	0 (0)	0 (0)	0 (0)	0 (0)	0 (0)	0 (0)	0 (0)	0 (0)	0 (0)	0 (0)	0 (0)
Focus	65 (0)	100 (0)	0 (0)	0 (0)	0 (0)	0 (0)	65 (0)	0 (0)	0 (0)	0 (0)	0 (0)
explore	0 (0)	0 (0)	0 (0)	30 (0)	9 (0)	58 (0)	0 (0)	0 (0)	0 (0)	0 (0)	0 (0)

Table 6: Zero-shot Learning Results

	Statistical Ranks												
Frequency	0	1	2	3	4	5	6	7	8	9	10	11	12
llama3-8b(8.0B)	17	17	17					17		17		17	
phi3-mini(3.8B)	33	33						17			17		
rrp	17		17		17		17			17		17	
exploit	100												
random		50	17		17		17						
baseline			17		33			17	17				17
Focus	33	33	17				17						
explore	17	17	17					33	17				
Evaluations													
llama3-8b(8.0B)	0 (0)	30 (0)	20 (0)	30 (0)	0 (0)	0 (0)	0 (0)	30 (0)	20 (0)	0 (0)	0 (0)	20 (0)	0 (0)
phi3-mini(3.8B)	0 (0)	30 (0)	30 (0)	0 (0)	0 (0)	0 (0)	0 (0)	30 (0)	0 (0)	0 (0)	30 (0)	0 (0)	0 (0)
rrp	0 (0)	17 (0)	0 (0)	11 (0)	0 (0)	9 (0)	0 (0)	11 (0)	9 (0)	0 (0)	0 (0)	10 (0)	0 (0)
exploit	40 (0)	0 (0)	0 (0)	0 (0)	0 (0)	0 (0)	0 (0)	0 (0)	0 (0)	0 (0)	0 (0)	0 (0)	0 (0)
random	0 (0)	40 (0)	45 (0)	0 (0)	50 (0)	40 (0)	0 (0)	0 (0)	0 (0)	0 (0)	0 (0)	0 (0)	0 (0)
baseline	0 (0)	0 (0)	86058 (0)	0 (0)	0 (0)	0 (0)	770 (0)	0 (0)	0 (0)	206 (0)	1152 (0)	0 (0)	398 (0)
Focus	33 (0)	53 (0)	45 (0)	0 (0)	0 (0)	20 (0)	0 (0)	0 (0)	0 (0)	0 (0)	0 (0)	0 (0)	0 (0)
explore	35 (0)	35 (0)	20 (0)	0 (0)	0 (0)	0 (0)	0 (0)	0 (0)	28 (0)	45 (0)	0 (0)	0 (0)	0 (0)
Deltas													
llama3-8b(8.0B)	0 (0)	49 (0)	30 (0)	75 (0)	0 (0)	0 (0)	0 (0)	86 (0)	66 (0)	0 (0)	0 (0)	24 (0)	0 (0)
phi3-mini(3.8B)	0 (0)	63 (0)	67 (0)	0 (0)	0 (0)	0 (0)	0 (0)	75 (0)	0 (0)	0 (0)	40 (0)	0 (0)	0 (0)
rrp	0 (0)	47 (0)	0 (0)	75 (0)	0 (0)	15 (0)	0 (0)	86 (0)	61 (0)	0 (0)	0 (0)	31 (0)	0 (0)
exploit	74 (0)	0 (0)	0 (0)	0 (0)	0 (0)	0 (0)	0 (0)	0 (0)	0 (0)	0 (0)	0 (0)	0 (0)	0 (0)
random	0 (0)	54 (0)	100 (0)	0 (0)	89 (0)	56 (0)	0 (0)	0 (0)	0 (0)	0 (0)	0 (0)	0 (0)	0 (0)
baseline	0 (0)	0 (0)	0 (0)	0 (0)	0 (0)	0 (0)	0 (0)	0 (0)	0 (0)	0 (0)	0 (0)	0 (0)	0 (0)
Focus	77 (0)	85 (0)	34 (0)	0 (0)	0 (0)	55 (0)	0 (0)	0 (0)	0 (0)	0 (0)	0 (0)	0 (0)	0 (0)
explore	34 (0)	85 (0)	5 (0)	0 (0)	0 (0)	0 (0)	0 (0)	0 (0)	51 (0)	44 (0)	0 (0)	0 (0)	0 (0)

provide a clear advantage over the more generalized, yet less specialized, capabilities of LLMs. Our analysis highlights that the *Exploit* strategy consistently outperforms not only all LLM treatments but also other acquisition functions within the Bayesian optimization framework. This finding underscores the robustness and reliability of the *Exploit* strategy in active learning scenarios, particularly for tabular data.

In conclusion, while LLMs show promise in many areas of machine learning, their current performance on tabular datasets, especially in the context of active learning, falls short when compared to Bayesian optimization methods. These findings suggest that, at least for the foreseeable future, Bayesian methods remain the preferred choice for optimizing performance and efficiency in tabular data-driven tasks.

6 Limitations and Future Work

Despite these insights, there remains an open question about the potential of larger and more advanced LLMs. It is unclear whether such models, possibly with greater parameter counts or enhanced fine-tuning, could close the gap with Bayesian methods or even surpass them. Future research should explore this avenue, testing the limits of LLM capabilities in comparison to traditional optimization techniques.

In this work we have limited the initialization of data points to be random. However prior work have indicated that guided initialization methods might perform better that random initialization which is a potential area of research. Additionally, exploring hybrid approaches that combine the strengths of LLMs and Bayesian methods could provide a fruitful direction for further investigation.

References

[1] Virginia Aglietti, Ira Ktena, Jessica Schrouff, Eleni Sgouritsa, Francisco JR Ruiz, Alexis Bellot, and Silvia Chiappa. 2024. FunBO: Discovering Acquisition Functions for Bayesian Optimization with FunSearch. *arXiv preprint arXiv:2406.04824* (2024).

[2] Monica Agrawal, Stefan Hegselmann, Hunter Lang, Yoon Kim, and David Sontag. 2022. Large language models are few-shot clinical information extractors. *arXiv preprint arXiv:2205.12689* (2022).

[3] Eric Brochu, Vlad M Cora, and Nando De Freitas. 2010. A tutorial on Bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning. *arXiv preprint arXiv:1012.2599* (2010).

[4] Samuel Budd, Emma C Robinson, and Bernhard Kainz. 2021. A survey on active learning and human-in-the-loop deep learning for medical image analysis. *Medical image analysis* 71 (2021), 102062.

[5] Qi Fan, Wei Zhuo, Chi-Keung Tang, and Yu-Wing Tai. 2020. Few-shot object detection with attention-RPN and multi-relation detector. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 4013–4022.

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

[7] Timothy Hospedales, Antreas Antoniou, Paul Micaelli, and Amos Storkey. 2021. Meta-learning in neural networks: A survey. *IEEE transactions on pattern analysis and machine intelligence* 44, 9 (2021), 5149–5169.

[8] Stephen James, Zicong Ma, David Rovick Arrojo, and Andrew J Davison. 2020. Rlbench: The robot learning benchmark & learning environment. *IEEE Robotics and Automation Letters* 5, 2 (2020), 3019–3026.

[9] Eric Jang, Alex Irpan, Mohi Khansari, Daniel Kappler, Frederik Ebert, Corey Lynch, Sergey Levine, and Chelsea Finn. 2022. Bc-z: Zero-shot task generalization with robotic imitation learning. In *Conference on Robot Learning*. PMLR, 991–1002.

[10] Jinggui Liang, Lizi Liao, Hao Fei, Bobo Li, and Jing Jiang. 2024. Actively Learn from LLMs with Uncertainty Propagation for Generalized Category Discovery. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*. 7838–7851.

[11] Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. 2023. Let's verify step by step. *arXiv preprint arXiv:2305.20050* (2023).

[12] Tennison Liu, Nicolás Astorga, Nabeel Seedat, and Mihaela van der Schaar. 2024. Large language models to enhance bayesian optimization. *arXiv preprint arXiv:2402.03921* (2024).

[13] Juhong Min, Dahyun Kang, and Minsu Cho. 2021. Hypercorrelation squeeze for few-shot segmentation. In *Proceedings of the IEEE/CVF international conference on computer vision*. 6941–6952.

[14] Baolin Peng, Chenguang Zhu, Chunyuan Li, Xiujun Li, Jinchao Li, Michael Zeng, and Jianfeng Gao. 2020. Few-shot natural language generation for task-oriented dialog. *arXiv preprint arXiv:2002.12328* (2020).

[15] Christopher Schröder, Andreas Niekler, and Martin Potthast. 2021. Revisiting uncertainty-based query strategies for active learning with transformers. *arXiv preprint arXiv:2107.05687* (2021).

[16] Ramanan Sekar, Oleh Rybkin, Kostas Daniilidis, Pieter Abbeel, Danijar Hafner, and Deepak Pathak. 2020. Planning to explore via self-supervised world models. In *International conference on machine learning*. PMLR, 8583–8592.

[17] Chan Hee Song, Jiaman Wu, Clayton Washington, Brian M Sadler, Wei-Lun Chao, and Yu Su. 2023. Llm-planner: Few-shot grounded planning for embodied agents with large language models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2998–3009.

[18] Hongjin Su, Jungo Kasai, Chen Henry Wu, Weijia Shi, Tianlu Wang, Jiayi Xin, Rui Zhang, Mari Ostendorf, Luke Zettlemoyer, Noah A Smith, et al. 2022. Selective annotation makes language models better few-shot learners. *arXiv preprint arXiv:2209.01975* (2022).

[19] Xingjiao Wu, Luwei Xiao, Yixuan Sun, Junhang Zhang, Tianlong Ma, and Liang He. 2022. A survey of human-in-the-loop for machine learning. *Future Generation Computer Systems* 135 (2022), 364–381.

[20] Ruixuan Xiao, Yiwen Dong, Junbo Zhao, Runze Wu, Minmin Lin, Gang Chen, and Haobo Wang. 2023. Freeal: Towards human-free active learning in the era of large language models. *arXiv preprint arXiv:2311.15614* (2023).

[21] Michelle Yuan, Hsuan-Tien Lin, and Jordan Boyd-Graber. 2020. Cold-start active learning through self-supervised language modeling. *arXiv preprint arXiv:2010.09535* (2020).

[22] Renrui Zhang, Rongyao Fang, Wei Zhang, Peng Gao, Kunchang Li, Jifeng Dai, Yu Qiao, and Hongsheng Li. 2021. Tip-adapter: Training-free clip-adapter for better vision-language modeling. *arXiv preprint arXiv:2111.03930* (2021).

Received 20 February 2007; revised 12 March 2009; accepted 5 June 2009