

# Large Language Models for Predictive Analysis: How Far Are They?

Qin Chen<sup>\* 1</sup>, Yuanyi Ren<sup>\* 1</sup>, Xiaojun Ma<sup>† 1</sup>, Yuyang Shi<sup>2</sup>,

<sup>1</sup>Peking University, <sup>2</sup>Harvard University,

{chenqink, yyren, mxj}@pku.edu.cn, yuyangshi@fas.harvard.edu

## Abstract

Predictive analysis is a cornerstone of modern decision-making, with applications in various domains. Large Language Models (LLMs) have emerged as powerful tools in enabling nuanced, knowledge-intensive conversations, thus aiding in complex decision-making tasks. With the burgeoning expectation to harness LLMs for predictive analysis, there is an urgent need to systematically assess their capability in this domain. However, there is a lack of relevant evaluations in existing studies. To bridge this gap, we introduce the **PredictiQ** benchmark, which integrates 1130 sophisticated predictive analysis queries originating from 44 real-world datasets of 8 diverse fields. We design an evaluation protocol considering text analysis, code generation, and their alignment. Twelve renowned LLMs are evaluated, offering insights into their practical use in predictive analysis. Generally, we believe that existing LLMs still face considerable challenges in conducting predictive analysis. See [Github](#).

## 1 Introduction

Predictive analysis (Kumar and Garg, 2017) involves making predictions about future outcomes based on past data, using statistical modeling, data mining, and machine learning techniques. It is widely used in decision-making across various fields like business (Che et al., 2024) and healthcare (Dixon et al., 2024).

To facilitate off-the-shelf predictive analysis for users without data analysis expertise, Large Language Models (LLMs) (Brown et al., 2020; OpenAI, 2023; Anil et al., 2023; Abdin et al., 2024) serve as powerful tools, supporting users with coherent and contextually relevant responses. [Figure 1](#) shows how users engage with LLMs by submitting predictive queries based on specific datasets.

<sup>\*</sup>Equal contribution.

<sup>†</sup>Corresponding author.

Given the substantial potential of LLM-based predictive analysis, a comprehensive evaluation of leading LLMs is essential. While no studies specifically assess LLM-based predictive analysis, related studies (Zhao et al., 2023b; Chen, 2022; Saeed et al., 2023; Gao et al., 2023; He et al., 2023; Abolghasemi et al., 2024; Pratt et al., 2024; Hong et al., 2024) in relevant fields primarily focus on evaluating either the model’s outcomes only (e.g., a total sales figure answering a database query) or the generated codes that, when executed, produce these outcomes. However, the former raises scalability issues as LLMs may struggle to process the entire lengthy dataset due to limited context length. The latter lacks textual explanations, such as justifications for algorithm selection, which constrains its applicability and undermines user trust. This makes existing evaluation efforts insufficient when directly adapted to predictive analysis, where tasks are inherently more complex.

LLM-based predictive analysis typically requires handling tasks such as data preprocessing, algorithm selection, result interpretation, and so on. These tasks require both **textual explanations** and **code implementations**. In practical applications, textual analysis is crucial for explaining algorithm selection and enhancing reliability. Moreover, effective alignment between text and code improves user comprehension of LLM-generated solutions. Thus, **textual analysis**, **code generation**, and **text-code alignment** are integral to the functionality of AI-driven assistants.

In this paper, we propose the PredictiQ (Q stands for query) benchmark to evaluate the potential of current LLMs in predictive analysis. We first collect datasets covering eight common fields, including economics, traffic, and more. Diverse datasets form the basis for PredictiQ to assess LLMs’ performance in predictive analysis. Based on the collected datasets, we collaborate with data analysis experts to formulate predictive queries tailored to

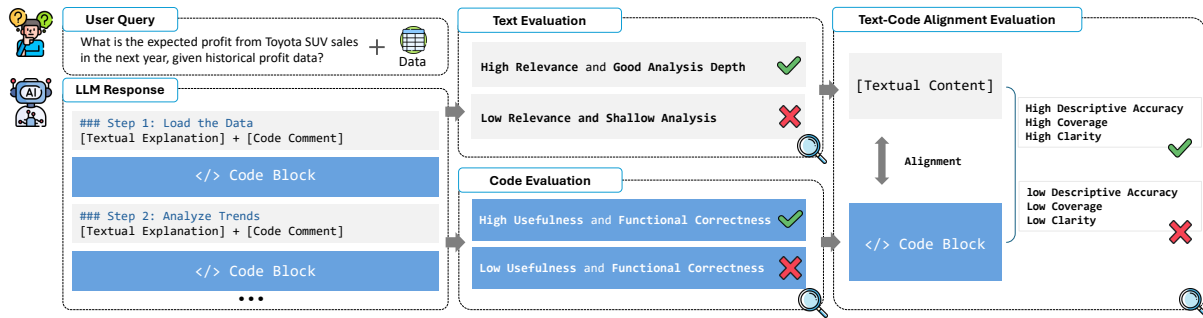


Figure 1: An example of users conducting predictive analysis via Large Language Models.

the datasets. Each query is limited to a single, self-contained question that relies solely on the dataset, without requiring external knowledge. We then formulate the data-specific queries and corresponding datasets into prompts, which are provided to LLMs to generate responses. We propose an evaluation protocol encompassing three domains: textual analysis, code generation, and text-code alignment, focusing on seven key aspects. The responses from the LLMs are then evaluated by (i) data analysis experts and (ii) LLMs. The LLM whose responses align most closely with those of the human experts is selected as the primary evaluator. In this study, we adopt GPT4Turbo, which achieves an average alignment ratio of 90.5% with human experts. PredictiQ involves 1130 queries from 44 datasets across 8 fields, requiring 300 human hours for query generation and examination. Our evaluation on PredictiQ costs 900 human hours for response evaluation, 72.18 million input tokens, and 20.4 million output tokens.

We conduct extensive experiments on PredictiQ with twelve renowned LLMs. We find that (i) Fine-tuning LLMs on code enhances both text analysis and code generation, occasionally pushing the model beyond its parameter limits. (ii) Code generation and text analysis are interconnected processes that shape LLMs’ overall predictive analysis proficiency. (iii) Several LLMs’ predictive analysis abilities vary significantly across different fields, with wide margins, and exhibit diversified score distributions. We believe existing LLMs are far from mastering predictive analysis, both in terms of performance and efficiency. Our contributions are summarized as follows.

- We propose the PredictiQ benchmark—a comprehensive evaluation framework for LLMs in predictive analysis. It covers the entire analytic process by integrating 44 datasets across

8 real-world fields, 1,130 rigorously curated data-specific queries, and evaluation protocols. This approach goes beyond existing benchmarks that focus only on basic descriptive tasks.

- We systematically evaluate the performance of twelve LLMs on the PredictiQ benchmark, highlighting insights that could improve the efficacy of LLMs in predictive analysis.

## 2 Preliminary

**Predictive Analysis** refers to the use of statistical algorithms and machine learning techniques to analyze data patterns, predict future events, trends, or behaviors (Siegel, 2013; Kumar and Garg, 2018). It is widely used in various fields, including finance, marketing, healthcare, and risk management, to anticipate potential risks, identify opportunities, and inform the decision-making process. See Figure 1 for an illustration.

**Problem Definition.** The predictive analysis process with LLMs is formalized as follows:

$$(\text{query}, \text{data}) \xrightarrow{\text{LLM}} (\text{text}, \text{code}). \quad (1)$$

The input consists of a predictive analysis *query* associated with corresponding *data*. A *query* is a data-specific predictive question that requests future predictions, identifies implicit data patterns, and so on. The *data* consists of multiple columns and rows. The output includes *text*, which explains how the prediction is made and justifies the chosen method, and *code*, which implements the analysis. Ideally, the *text* should provide clear and in-depth analysis based on the *query* and *data*. The *code* should provide related and functionally correct implementation that solves the problem raised by the *query*.

### 3 The PredictiQ Benchmark

This section outlines the details of the PredictiQ benchmark, covering (i) data collection, (ii) query formulation, and (iii) response evaluation.

#### 3.1 Data Collection

The PredictiQ benchmark evaluates LLMs’ predictive analysis capabilities across diverse fields. It includes various publicly available datasets from real-world scenarios, ensuring the benchmark’s diversity and validity. The datasets are organized into eight fields that are prevalent in real-world predictive analysis applications, as presented in Table 1. See Appendix A for detailed descriptions.

#### 3.2 Query Collection

Table 1: Statistics of datasets and generated queries.

Field	#Datasets	#Queries
Economics	12	270
Marketing and Sales	6	200
Industry Analysis	7	180
Traffic	5	130
Healthcare	4	130
Social Study	4	110
Human Resource	3	80
Education	3	70
<b>Total</b>	<b>44</b>	<b>1130</b>

In this subsection, our goal is to formulate predictive questions tailored to each dataset. To achieve this, we engaged data science experts to develop well-defined queries following our instructions and examples. The provided instructions clarify the context of predictive analysis and outline the constraints on expected outcomes, as detailed below.

**Context of Predictive Analysis.** Based on our research into the application of predictive analysis across various fields (Wazurkar et al., 2017; Mishra et al., 2023; Saranya and Asha, 2019; Liu et al., 2023), we categorize common questioning methods into two types: those focused on predicting future trends and those focused on identifying previously unrecognized patterns within the dataset. We also encourage participants to consider queries related to time-series data.

**Constraints on Expected Outputs.** To ensure the quality of the outputs, we have implemented three specific instructions: (i) Participants are required to articulate a clear predictive target using

unambiguous language. (ii) Queries should be based solely on the information within the dataset, excluding any reference to external data that could compromise the evaluation’s integrity. (iii) Each query should be restricted to a single question, ensuring a more balanced distribution of workload across all queries. In addition, to improve real-world relevance and adaptability, participants are provided with background information on the sources and construction methods of the dataset.

To ensure diversity, we engage experts to select unique questions that cover distinct aspects of the dataset. The number of queries for each dataset is then tailored to its informativeness. For further details, see Appendix A. In total, we have generated 1130 queries from 44 datasets, covering common application scenarios in predictive analysis. The process of generating and examining the queries requires roughly 300 human hours.

#### 3.3 Response Generation

With the datasets and collected queries, we organize the data to formulate the input prompts. Each prompt consists of three parts:

- **Query and Instruction:** We instruct the LLM to perform predictive analysis, assuming the role of a professional data scientist.
- **Data Summary:** We provide a list of all columns in the dataset with their respective data types (e.g., int). We also include details such as the maximum and minimum values for numerical columns and the total number of categories for categorical columns. These summaries help the LLM better understand the dataset.
- **Data Details:** The dataset is provided in a CSV format, with columns separated by commas and rows by line breaks. This detailed format gives the LLM the essential information needed for predictive analysis.

The prompts are then submitted to the LLMs to generate responses. An example of a prompt and its corresponding response is provided in Appendix E.

#### 3.4 Evaluation Protocols

As mentioned in section 1, LLM-based predictive analysis involves both code and textual descriptions. It is essential to evaluate all three components: code generation, textual explanations, and

the alignment between the two. Textual analysis offers key insights that enhance reliability, such as justifying algorithm choices, while code implementation operationalizes the analysis. Proper alignment helps users better understand the solutions generated by LLMs. Building on this, we propose an evaluation protocol spanning three domains and seven aspects, each rated on a scale from 0 to 4. A summary is provided below, with full details available in Table 9.

**Text Analysis.** The text generated by LLMs is evaluated on two aspects: *Relevance*, which assesses how closely the analysis aligns with the given data and the specific question, and *Depth*, which evaluates the comprehensiveness of the justification for selecting a particular model or algorithm. We expect the textual analysis to provide a detailed and thorough examination of the analysis process, tailored to the specific data and queries.

**Code Generation.** This domain focuses on the quality of the code produced by LLMs with two aspects. Firstly, *Usefulness* evaluates how well the code snippet addresses the given problem. Secondly, *Functional Correctness* examines the code’s execution correctness. We expect the generated code to accurately and effectively implement the associated predictive analysis.

**Text-Code Alignment.** We further evaluate the congruence between the generated code and textual analysis across three aspects: (i) *Descriptive Accuracy*, which assesses how precisely the text reflects the code; (ii) *Coverage*, evaluating whether the generated text addresses all relevant aspects of the code, including its functions and nuances; and (iii) *Clarity*, which examines the clarity of the alignment between the code and text. We expect the generated text and code to be well-aligned, and comprehensive, enhancing the user’s understanding of the entire analysis process.

The proposed protocol offers a comprehensive evaluation of the responses. Advanced LLMs enable large-scale, stable, and effective assessment of the responses. To investigate how different LLM evaluators agree with experts’ ratings and choose the appropriate evaluator, we employ (i) human experts and (ii) LLMs to assess the responses according to this protocol. We then compare the score distributions from both groups to determine which LLM aligns most closely with human experts. Among the LLMs tested — GPT4Turbo, GPT4O, and Phi3Medium — GPT4Turbo demonstrated the highest alignment with human evalu-

ations, as shown in our experiment in Figure 4. Consequently, we adopt GPT4Turbo as the primary evaluator. For assessing *Functional Correctness*, we manually execute the generated code and evaluate the ratio of successful executions without errors.

## 4 Evaluation Results on PredictiQ

### 4.1 Involved LLMs

We evaluate eight popular LLMs on PredictiQ. (i) **GPT Family.** GPT3.5Turbo, GPT4Turbo, GPT4O, GPT4O1, and GPT4O3Mini (Brown et al., 2020; OpenAI, 2023, 2024) are potent large-scale language models fine-tuned for both chat and code generation. (ii) **Llama Family.** This family includes CodeLlama2-7B, ChatLlama2-7B, ChatLlama2-13B, and ChatLlama2-70B (Touvron et al., 2023). CodeLlama2-7B is fine-tuned for both chat and code generation. Other versions are reported to be fine-tuned for chat. (iii) **Other Models.** We also incorporate other popular LLMs, including Phi3Medium (Microsoft, 2024a), Phi4 (Microsoft, 2024b) and CohereRPlus<sup>1</sup>, into our evaluation. For LLM settings, please refer to Appendix C.

### 4.2 Evaluation on PredictiQ

GPT4O3Mini outperforms all others, achieving the highest total score of 24.11 out of 28. Within the GPT family, GPT4O1 incurs significantly higher token costs, up to twice those of GPT4O. GPT4O3Mini, a refined version, improves both efficiency and performance compared to GPT4O1. Models from the Llama family (from 7B to 70B), whether fine-tuned for code generation or not, exhibit relatively low scores. Generally, larger parameter scale enhances overall performance but also increases token usage.

**Finding 1:** LLMs often fail to produce complete and executable solutions, frequently overlooking steps like data pre-processing. While larger models handle import errors better, they still struggle with logic errors.

LLMs often overlook essential procedures like data pre-processing (e.g., handling missing values) and filtering in the generated code, as revealed in Table 3. For GPT4O3Mini, only 49% of the generated code includes proper pre-processing and filtering, while the rest operate on unclean data. This

<sup>1</sup><https://docs.cohere.com/v2/docs/command-r-plus>



Table 2: Evaluation results on PredictiQ across seven perspectives (scored 0–4) outlined in Table 9. *Functional Correctness* represents the ratio of executable code, mapped to a 0-4 scoring scale. We also report average token costs, with additional reasoning costs for GPT4O1 and GPT4O3Mini.

LLM	Text		Code		Text-Code Alignment			Total	Token Cost
	Relevance	Depth	Usefulness	Functional Correctness	Descriptive Accuracy	Coverage	Clarity		
GPT3.5Turbo	3.00 $\pm$ 0.96	1.76 $\pm$ 0.69	2.40 $\pm$ 0.86	2.12 (53%)	2.66 $\pm$ 1.01	2.47 $\pm$ 0.94	2.80 $\pm$ 0.96	17.21 $\pm$ 5.19	1934.34
GPT4Turbo	3.39 $\pm$ 0.79	2.18 $\pm$ 0.64	2.78 $\pm$ 0.72	3.12 (78%)	3.09 $\pm$ 0.80	2.95 $\pm$ 0.77	3.18 $\pm$ 0.77	20.68 $\pm$ 4.23	2072.95
GPT4O	3.60 $\pm$ 0.65	2.39 $\pm$ 0.71	3.12 $\pm$ 0.67	3.24 (81%)	3.36 $\pm$ 0.68	3.31 $\pm$ 0.67	3.41 $\pm$ 0.64	22.43 $\pm$ 3.82	3390.74
GPT4O1	3.61 $\pm$ 0.51	2.80 $\pm$ 0.53	3.45 $\pm$ 0.60	3.40 (85%)	3.47 $\pm$ 0.68	3.48 $\pm$ 0.67	3.48 $\pm$ 0.67	23.70 $\pm$ 3.40	6534.45
GPT4O3Mini	3.63 $\pm$ 0.53	2.91 $\pm$ 0.42	3.53 $\pm$ 0.55	3.48 (87%)	3.52 $\pm$ 0.61	3.52 $\pm$ 0.61	3.52 $\pm$ 0.61	24.11 $\pm$ 3.13	4402.94
ChatLlama2-7B	2.01 $\pm$ 0.87	1.31 $\pm$ 0.65	1.49 $\pm$ 0.68	0.72 (18%)	0.83 $\pm$ 0.82	0.85 $\pm$ 0.79	1.14 $\pm$ 0.94	8.34 $\pm$ 4.50	1763.87
CodeLlama2-7B	2.04 $\pm$ 0.91	1.34 $\pm$ 0.66	1.64 $\pm$ 0.71	0.60 (15%)	0.99 $\pm$ 0.90	1.00 $\pm$ 0.89	1.22 $\pm$ 1.00	8.83 $\pm$ 4.91	1786.59
ChatLlama2-13B	1.97 $\pm$ 0.88	1.24 $\pm$ 0.64	1.53 $\pm$ 0.69	0.72 (18%)	1.02 $\pm$ 0.84	1.03 $\pm$ 0.79	1.24 $\pm$ 0.89	8.75 $\pm$ 4.49	2032.84
ChatLlama2-70B	2.32 $\pm$ 0.90	1.51 $\pm$ 0.67	1.78 $\pm$ 0.73	0.84 (21%)	1.25 $\pm$ 0.92	1.27 $\pm$ 0.90	1.60 $\pm$ 0.96	10.57 $\pm$ 4.85	2487.18
Phi3Medium	2.90 $\pm$ 1.25	1.74 $\pm$ 0.81	2.33 $\pm$ 1.04	1.64 (41%)	2.45 $\pm$ 1.21	2.33 $\pm$ 1.14	2.58 $\pm$ 1.18	15.97 $\pm$ 6.50	3314.35
Phi4	2.94 $\pm$ 0.24	2.55 $\pm$ 0.58	2.87 $\pm$ 0.35	2.16 (54%)	2.84 $\pm$ 0.43	2.82 $\pm$ 0.45	2.88 $\pm$ 0.39	19.06 $\pm$ 2.26	3815.88
CohereRPlus	2.89 $\pm$ 0.95	1.70 $\pm$ 0.64	2.38 $\pm$ 0.79	1.68 (42%)	2.50 $\pm$ 0.92	2.42 $\pm$ 0.86	2.62 $\pm$ 0.87	16.20 $\pm$ 4.86	2851.20

issue is even more pronounced in smaller models. As shown in Table 3, we observe the following: (i) Smaller models frequently face import errors, such as using non-imported functions, which decrease as model capacity grows. (ii) Logic errors, including syntax issues and function call errors, persist across models, indicating the need for future improvements. (iii) Fine-tuning on code helps CodeLlama2-7B reduce import errors but increases logic errors, resulting in a lower executable code rate than ChatLlama2-7B.

Table 3: Analysis on the portion of the codes without pre-processing, and the frequencies of error types.

LLM	No Pre-processing	Import Error	Logic Error
GPT3.5Turbo	71%	3.8%	43.2%
GPT4Turbo	66%	1.3%	20.7%
GPT4O	66%	0.4%	18.6%
GPT4O1	50%	0.3%	14.7%
GPT4O3Mini	51%	0.3%	12.7%
ChatLlama2-7B	92%	41.8%	40.1%
CodeLlama2-7B	89%	38.2%	46.8%
ChatLlama2-13B	87%	36.1%	45.9%
ChatLlama2-70B	87%	15.0%	64.0%
Phi3Medium	72%	5.9%	53.1%
Phi4	58%	3.8%	42.2%
CohereRPlus	78%	4.6%	53.4%

**Finding 2:** Fine-tuning on code generation improves the overall performance of models, sometimes allowing them to perform better than their parameter size would suggest.

With 46% fewer parameters, CodeLlama2-7B outperforms ChatLlama2-13B in total scores, align-

ing with the conclusion of (Zhou et al., 2023). Fine-tuning on code remarkably improves model performance in code relevance, code-text alignment, and the quality of textual description.

**Finding 3:** Fine-tuning for code generation increases the frequency and length of code produced by CodeLlama2-7B, but it **negatively impacts** the model’s ability to generate executable data analysis code.

To our surprise, CodeLlama2-7B achieves 15% of the executable code rate, falling behind even ChatLlama2-7B. We find that, at this parameter level, fine-tuning on code generation helps to boost the performance of other domains, including textual analysis and text-code alignment. However, in terms of executable code rate, its performance is downgraded. This may be attributed to the overly specialized fine-tuning. Also, the reason may be that fine-tuning on code does not always focus on the data-analysis-related codes. For example, there may be some C++ kernel code that is not related to data analysis.

We systematically analyze the lengths and lines of the generated codes in Table 4. Despite our prompt instructions to generate codes, we find that models from the Llama family often ignore code generation. It turns out that fine-tuning on code only helps CodeLlama2-7B generate code more frequently, with a tendency to generate longer code. However, for a complex task like data-aware predictive analysis, it won’t improve the quality of the generated code at this scale.

Table 4: Analysis of the average length and number of lines of the generated code, as well as the ratio of cases with no code.

LLM	Length	#Lines	Absent
GPT3.5Turbo	967.45	27.22	4%
GPT4Turbo	1276.49	32.46	0
GPT4O	2423.53	65.79	0
GPT4O1	4375.41	127.37	0
GPT4O3Mini	3561.61	90.63	0
ChatLlama2-7B	660.72	17.41	64%
CodeLlama2-7B	863.85	21.64	60%
ChatLlama2-13B	627.51	16.68	63%
ChatLlama2-70B	611.32	16.54	53%
Phi3Medium	1336.94	30.79	0
Phi4	1469.85	33.85	0
CohereRPlus	1749.12	52.90	0

**Finding 4:** LLMs vary in score distribution across data source domains. GPT4O3Mini and GPT4O are well-balanced, while ChatLlama2-70B excels in the Education domain.

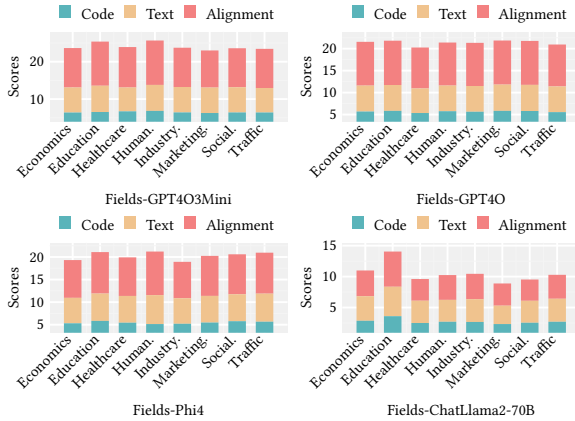


Figure 2: Score distributions of LLMs on eight fields. For clarity we present the total scores of text, code, and their alignment.

We present the performance of leading models — GPT4O3Mini, GPT4O, Phi4, and ChatLlama2-70B — within their respective families across eight fields (see Figure 2). The data highlight the varying effectiveness of these LLMs across domains. GPT4O3Mini and GPT4O demonstrate balanced performance, with deviations of 1.67 and 1.82 points, respectively. In contrast, Phi4 and ChatLlama2-70B exhibit greater variability, with score differences of 4.38 and 5.16 points. The

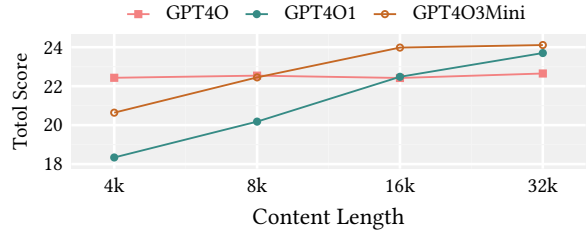


Figure 3: Analysis on impact of context length limit.

score distributions differ across fields. For instance, ChatLlama2-70B excels in *Education*, exceeding its average score by 31.4%, suggesting domain-specific strengths and weaknesses among models.

### 4.3 Impact of Context Length

We evaluate the impact of context length on GPT4O, GPT4O1, and GPT4O3Mini in Figure 3. At 4k tokens, GPT4O performs well, and increasing the token limit does not improve its performance. However, GPT4O1 and GPT4O3Mini, which need extra tokens for reasoning, perform much worse initially<sup>2</sup>. As context length increases, their performance improves rapidly, with GPT4O3Mini achieving similar results to GPT4O1 using fewer tokens.

### 4.4 LLMs as Evaluators

Table 5: Results on PredictIQ with different evaluators.

Evaluator	GPT4Turbo	GPT4O	Phi3Medium
GPT3.5Turbo	17.21±4.82	21.68±3.69	25.49±1.29
GPT4Turbo	20.34±3.92	25.34±2.08	26.45±1.42
GPT4O	22.43±3.72	26.13±1.97	26.86±1.93
GPT4O1	23.70±3.40	26.45±2.33	26.96±1.22
GPT4O3Mini	24.15±3.13	26.64±2.05	27.04±1.50
ChatLlama2-7B	8.34±4.50	13.94±5.02	22.93±4.56
CodeLlama2-7B	8.83±4.91	14.59±4.89	22.62±4.51
ChatLlama2-13B	8.75±4.49	15.75±4.13	23.67±3.34
ChatLlama2-70B	10.57±4.85	16.36±4.54	23.54±4.13
Phi3Medium	15.97±5.49	21.19±4.67	25.23±1.43
Phi4	19.06±2.26	21.19±4.67	25.23±1.43
CohereRPlus	16.20±4.83	20.87±4.54	25.07±1.28

To investigate how different LLM evaluators align with human preferences and select the proper evaluator, we engage (i) five experts in data analysis and (ii) LLMs including GPT4Turbo, GPT4O, and Phi3Medium to grade the responses according to the evaluation protocol<sup>3</sup>. We then calculate

<sup>2</sup>As stated in Appendix C, we set the context length to 32,768 for GPT4O1 and GPT4O3Mini to unlock their full potential, resulting in the outcomes in Table 2.

<sup>3</sup>We exclude GPT4O1 and GPT4O3Mini as evaluators due to their high computational cost and slow running speed.

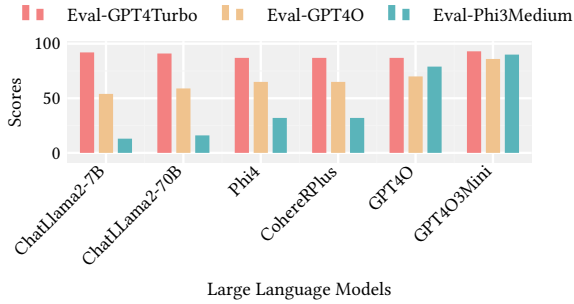


Figure 4: Alignment scores of different evaluators with human experts. See Figure 5 for full results.

the average scores from human experts and compare them with those from LLMs. Score pairs with an absolute difference smaller than 4 are labeled positive (1), while those with a larger difference are labeled negative (0). The alignment between LLM and expert scores is visualized in Figure 4, with higher scores indicating better alignment. Full results are presented in Figure 5.

**Finding 5:** GPT4Turbo’s evaluations align most closely with those of the experts, while GPT4O tends to be more positively biased. Phi3Medium is not applicable because it lacks differentiation in its evaluations.

As shown in Figure 4, GPT4Turbo aligns most closely with expert ratings, matching 90.5% of instances and maintaining a score distribution consistent with human evaluators across all target models. In contrast, GPT4O shows weaker alignment, particularly with the Llama family, where it frequently assigns disproportionately high scores to poor responses. Phi3Medium demonstrates the poorest alignment, assigning nearly indistinguishable scores across all targets. This may be attributed to the *round number bias* (Honda et al., 2022; Stureborg et al., 2024), where certain scores are assigned more frequently, irrespective of the context, impeding precise quality assessments. This could result from biased training data, sentiment-supportive fine-tuning, or limitations in reasoning abilities (Murugadoss et al., 2024; Wei et al., 2024; Wang et al., 2024).

**Finding 6:** The tendency of LLMs to assign themselves the highest scores when serving as evaluators is reduced in the context of predictive analysis.

Studies (Ye et al., 2024; Panickssery et al., 2024; Koo et al., 2024) show that when LLMs are used as evaluators, they often exhibit a bias towards assigning higher scores to their own responses. However, in our experimental setup, this bias appears to be mitigated. As demonstrated in Table 5, the gap between total scores ( $\text{Score}_{\text{GPT4O}} - \text{Score}_{\text{GPT4Turbo}}$ ), when evaluated by GPT4Turbo, is 2.09. In contrast, the gap narrows to 0.79 when evaluated by GPT4O. Interestingly, when the evaluator is switched from GPT4Turbo to GPT4O, the performance gap narrows rather than widens, indicating that the "ego-centric" (Koo et al., 2024) or "self-preference" (Panickssery et al., 2024) scoring tendency is reduced in this evaluation.

#### 4.5 Ablation Study

Table 6: Ablation study. For clarity, we present the total scores of three domains.

	Text	Code	Alignment	Total
GPT4O	5.99±1.22	6.36±0.67	10.08±1.93	22.43±3.82
w/o Role Play	5.78±1.14	6.22±0.67	9.94±2.07	21.94±3.88
w/o Data Summary	5.81±1.12	6.34±0.68	9.76±2.17	21.91±3.97
GPT4O3Mini	6.54±0.76	7.01±0.55	10.56±1.83	24.11±3.13
w/o Role Play	6.56±0.86	7.31±0.67	10.35±1.90	24.22±4.07
w/o Data Summary	6.25±0.56	7.18±0.67	10.22±1.76	23.65±3.13
ChatLlama2-70B	3.98±1.46	2.66±0.76	4.39±2.64	11.03±4.86
w/o Role Play	3.94±1.64	2.65±0.82	4.19±3.10	10.78±5.56
w/o Data Summary	3.84±1.65	2.64±0.77	4.22±3.05	10.70±5.46
Phi4	5.49±0.70	5.03±0.35	8.54±1.21	19.06±2.26
w/o Role Play	5.15±1.67	4.45±1.54	7.83±2.20	17.61±2.46
w/o Data Summary	4.85±3.18	4.68±2.47	7.97±3.85	17.50±3.15
CohereRPlus	4.59±1.50	4.06±0.79	7.54±2.57	16.20±4.86
w/o Role Play	5.01±1.54	4.21±0.80	8.20±2.79	17.42±5.13
w/o Data Summary	5.04±1.39	4.21±0.72	8.24±2.38	17.49±4.50

**Finding 7:** Prompt engineering primarily influences textual descriptions and text-code alignment, with relatively little impact on code generation.

We conduct ablation studies using a subset of 160 queries covering all fields from Table 1. We present the results of the top-performing models from each family in Table 6, with the full results provided in Table 16. The results show that prompt engineering primarily impacts textual descriptions and text-code alignment, with minimal effect on code generation. They enhance the performance of most models, except for CohereRPlus. We also find that prompt engineering has little impact on GPT4O3Mini, likely because the reasoning modules have similar functionalities, which is consis-

tent with the findings in its report <sup>4</sup>.

#### 4.6 LLMs for Predictive Analysis

Based on the aforementioned evaluations, existing LLMs still have significant room for improvement in predictive analysis, particularly in terms of performance and efficiency. While GPT4O3Mini achieves relatively high scores, it lacks depth in providing justifications for algorithmic choices (Depth score: 2.91 out of 4) and struggles with data preprocessing (51% of codes omit this step). Additionally, its high token consumption (with an average exceeding 4K tokens) and the requirement for a large context window (up to 32K tokens) present challenges for real-world applications, especially in privacy-sensitive scenarios where only local, smaller models are viable.

### 5 Related Works

#### 5.1 Large Language Models

Recent advancements in natural language processing (NLP) are largely attributed to Large Language Models (LLMs) (Zhao et al., 2023a; Chang et al., 2023) like GPT series (Brown et al., 2020; OpenAI, 2023, 2024), Gemini series (Google, 2024), Llama series (Touvron et al., 2023; Rozière et al., 2023; Meta, 2024), and so on. These models undergo extensive pre-training on vast text datasets and are further refined through techniques like reinforcement learning from human feedback (RLHF) (Kirk et al., 2024) and instruction fine-tuning (IFT) (Brown et al., 2020).

In parallel, models specifically fine-tuned by code-formatted data emerged, considering the wide-ranging demand for code understanding and program generation from natural language prompts. For example, Codex (Chen et al., 2021), CodeT (Chen et al., 2022), Code Llama (Rozière et al., 2023) and HiRoPE (Zhang et al., 2024) have displayed excellent performance in code-related tasks, such as code completion and description-to-code generation. Meanwhile, these fine-tuned LLMs have shown extra progress in solving math problems (Zhou et al., 2023; Xu et al., 2024).

#### 5.2 LLMs for data analysis

LLMs have recently demonstrated significant potential in various data analysis fields. Some studies concentrate on rudimentary data tasks in preparation for further analysis like early-stage data prepa-

ration (Zhang et al., 2023) and table-to-text summarization (Zhao et al., 2023b). Some studies (Chen, 2022; Saeed et al., 2023; Gao et al., 2023) focus on LLMs' performance in transforming texts into SQL queries with their execution on the corresponding database, mainly for elementary descriptive analysis. For more advanced data analysis, most existing research works (Yu et al., 2023; Jin et al., 2023; Schoenegger and Park, 2023; Pratt et al., 2024; Hong et al., 2024) are limited to domain-specific time series forecasting tasks and lack generalizability. Text2Analysis (He et al., 2023) introduces queries for four types of advanced data analysis. However, their evaluation is exclusively centered on code generation, neglecting the essential elements of textual explanation and the alignment between text and code. Generally, there is a noticeable lack of comprehensive evaluations of LLMs in predictive analysis.

### 6 Conclusion

In this paper, we propose the PredictiQ benchmark, addressing the gap in evaluating LLMs' capabilities in predictive analysis. The benchmark provides domain-diversified datasets, data-specific queries, and in-depth evaluation protocols. We evaluate twelve popular LLMs on the benchmark, highlighting their capabilities and limitations in predictive analysis. From the evaluation, we find the following key insights: (i) Fine-tuning LLMs on code boosts their performance, sometimes exceeding the limits of model parameters. (ii) Code generation and text analysis abilities are interconnected, jointly influencing LLMs' predictive analysis skills. (iii) LLMs exhibit varying levels of predictive ability across domains. Overall, existing LLMs still have significant room for improvement in achieving both high performance and efficiency in predictive analysis.

### 7 Limitations

This work exhibits the following limitations. (i) As discussed in section 2, this work centers on evaluating the performance of LLMs in predictive analysis, and does not extend to other advanced data analysis fields such as prescriptive analysis or diagnostic analysis. Future research is expected to investigate broader analysis dimensions and assess effectiveness of LLMs across diverse analytical domains. (ii) The dataset employed in this study is confined to a limited number of commonly

<sup>4</sup><https://platform.openai.com/docs/guides/reasoning>



encountered fields, potentially overlooking requirements and challenges present in less-represented or novel fields. This limitation may impact the generalizability of the findings and suggests that future work should incorporate more diverse datasets to explore how LLMs perform across a broader range of fields.

## 8 Ethics Statement

This work benchmarks the efficacy of predictive analysis in LLMs. The evaluation involves datasets collected from publicly available platforms, detailed in [Appendix A](#). We ensure that the collected datasets do not contain any personally identifiable information. Additionally, we rigorously verify the data licenses and copyright permissions to confirm that they authorize public use for research purposes. We need to further clarify that this study focuses on the evaluation of language models and does not introduce new model architectures or deployment strategies. While we acknowledge that flawed model predictions, particularly in sensitive or high-stakes domains, can contribute to biased or harmful outcomes, addressing such risks requires domain-specific considerations and interventions that fall beyond the scope of this work. The evaluation intends to inform and support future research aimed at improving model reliability and mitigating downstream risks in applied settings.

In the study, volunteers consisting of master's students in data science with an Asian background conducted human annotation to generate and evaluate the queries. They also participate in manual code execution and LLM response evaluation. While these annotators possess a solid foundation of predictive analysis, there is a potential risk that individuals from a specific cultural background may exhibit biases in their comprehension of query content and LLM responses.

We have used ChatGPT to assist us in refining the expression of our paper.

## References

Marah Abdin, Jyoti Aneja, Harkirat Behl, Sébastien Bubeck, Ronen Eldan, Suriya Gunasekar, Michael Harrison, Russell J Hewett, Mojan Javaheripi, Piero Kauffmann, et al. 2024. Phi-4 technical report. *arXiv preprint arXiv:2412.08905*.

Mahdi Abolghasemi, Odkhishig Ganbold, and Kristian Rotaru. 2024. [Humans vs. large language models:](#)

[Judgmental forecasting in an era of advanced ai](#). *International Journal of Forecasting*.

Rohan Anil, Andrew M Dai, Orhan Firat, Melvin Johnson, Dmitry Lepikhin, Alexandre Passos, Siamak Shakeri, Emanuel Taropa, Paige Bailey, Zhifeng Chen, et al. 2023. Palm 2 technical report. *arXiv preprint arXiv:2305.10403*.

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

Yupeng Chang, Xu Wang, Jindong Wang, Yuan Wu, Linyi Yang, Kaijie Zhu, Hao Chen, Xiaoyuan Yi, Cunxiang Wang, Yidong Wang, et al. 2023. A survey on evaluation of large language models. *ACM Transactions on Intelligent Systems and Technology*.

Chang Che, Zengyi Huang, Chen Li, Haotian Zheng, and Xinyu Tian. 2024. [Integrating generative ai into financial market prediction for improved decision making](#). *Preprint*, arXiv:2404.03523.

Bei Chen, Fengji Zhang, A. Nguyen, Daoguang Zan, Zeqi Lin, Jian-Guang Lou, and Weizhu Chen. 2022. [Codet: Code generation with generated tests](#). *ArXiv*, abs/2207.10397.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, Dave Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William Hebguss, Alex Nichol, Alex Paino, Nikolas Tezak, Jie Tang, Igor Babuschkin, Suchir Balaji, Shantanu Jain, William Saunders, Christopher Hesse, Andrew N. Carr, Jan Leike, Josh Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. 2021. [Evaluating large language models trained on code](#). *Preprint*, arXiv:2107.03374.

Qin Chen, Liang Wang, Bo Zheng, and Guojie Song. 2025. [Dagprompt: Pushing the limits of graph prompting with a distribution-aware graph prompt tuning approach](#). In *Proceedings of the ACM on Web Conference 2025*, pages 4346–4358.

Wenhu Chen. 2022. Large language models are few (1)-shot table reasoners. *arXiv preprint arXiv:2210.06710*.

Dursun Delen and Sudha Ram. 2018. Research challenges and opportunities in business analytics. *Journal of Business Analytics*, 1(1):2–12.

- Diny Dixon, Hina Sattar, Natalia Moros, Srija Reddy Kesireddy, Huma Ahsan, Mohit Lakkimsetti, Madiha Fatima, Dhruvi Doshi, Kanwarpreet Sadhu, and Muhammad Junaid Hassan. 2024. Unveiling the influence of ai predictive analytics on patient outcomes: A comprehensive narrative review. *Cureus*, 16.
- Dawei Gao, Haibin Wang, Yaliang Li, Xiuyu Sun, Yichen Qian, Bolin Ding, and Jingren Zhou. 2023. [Text-to-sql empowered by large language models: A benchmark evaluation](#). *Preprint*, arXiv:2308.15363.
- Google. 2024. [Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context](#). *Preprint*, arXiv:2403.05530.
- W.H. Greene. 2019. *Econometric Analysis, Global Edition*. Pearson Education.
- Xinyi He, Mengyu Zhou, Xinrun Xu, Xiaojun Ma, Rui Ding, Lun Du, Yan Gao, Ran Jia, Xu Chen, Shi Han, et al. 2023. Text2analysis: A benchmark of table question answering with advanced data analysis and unclear queries. *arXiv preprint arXiv:2312.13671*.
- Hidehito Honda, Rina Kagawa, and Shirasuna Masaru. 2022. [On the round number bias and wisdom of crowds in different response formats for numerical estimation](#). *Scientific Reports*, 12.
- Sirui Hong, Yizhang Lin, Bang Liu, Bangbang Liu, Binhao Wu, Ceyao Zhang, Chenxing Wei, Danyang Li, Jiaqi Chen, Jiayi Zhang, Jinlin Wang, Li Zhang, Lingyao Zhang, Min Yang, Mingchen Zhuge, Taicheng Guo, Tuo Zhou, Wei Tao, Xiangru Tang, Xiangtao Lu, Xiawu Zheng, Xinbing Liang, Yaying Fei, Yuheng Cheng, Zhibin Gou, Zongze Xu, and Chenglin Wu. 2024. [Data interpreter: An llm agent for data science](#). *Preprint*, arXiv:2402.18679.
- Ming Jin, Shiyu Wang, Lintao Ma, Zhixuan Chu, James Y Zhang, Xiaoming Shi, Pin-Yu Chen, Yuxuan Liang, Yuan-Fang Li, Shirui Pan, et al. 2023. Time-llm: Time series forecasting by reprogramming large language models. *arXiv preprint arXiv:2310.01728*.
- Robert Kirk, Ishita Mediratta, Christoforos Nalpanitis, Jelena Luketina, Eric Hambro, Edward Grefenstette, and Roberta Raileanu. 2024. [Understanding the effects of rlhf on llm generalisation and diversity](#). *Preprint*, arXiv:2310.06452.
- Ryan Koo, Minhwa Lee, Vipul Raheja, Jong Inn Park, Zae Myung Kim, and Dongyeop Kang. 2024. Benchmarking cognitive biases in large language models as evaluators. In *Findings of the Association for Computational Linguistics ACL 2024*, pages 517–545.
- Vaibhav Kumar and M. L. Garg. 2017. [Deep learning in predictive analytics: A survey](#). In *2017 International Conference on Emerging Trends in Computing and Communication Technologies (ICETCCT)*, pages 1–6.
- Vaibhav Kumar and ML Garg. 2018. Predictive analytics: a review of trends and techniques. *International Journal of Computer Applications*, 182(1):31–37.
- Xuyuan Liu, Yinghao Cai, Qihui Yang, and Yujun Yan. 2024. [Exploring consistency in graph representations: from graph kernels to graph neural networks](#). *arXiv preprint arXiv:2410.23748*.
- Zichen Liu, Xuyuan Liu, Yanlong Wen, Guoqing Zhao, Fen Xia, and Xiaojie Yuan. 2023. [Treeman: tree-enhanced multimodal attention network for icd coding](#). *arXiv preprint arXiv:2305.18576*.
- Xiaojun Ma, Qin Chen, Yi Wu, Guojie Song, Liang Wang, and Bo Zheng. 2023. [Rethinking structural encodings: Adaptive graph transformer for node classification task](#). In *Proceedings of the ACM web conference 2023*, pages 533–544.
- Meta. 2024. [The llama 3 herd of models](#). *Preprint*, arXiv:2407.21783.
- Microsoft. 2024a. [Phi-3 technical report: A highly capable language model locally on your phone](#). *Preprint*, arXiv:2404.14219.
- Microsoft. 2024b. [Phi-4 technical report](#). *Preprint*, arXiv:2412.08905.
- Harsh Mishra, Siddharth Swarup Rautaray, and Manjusha Pandey. 2023. [Review on big data analytics and its impact on marketing strategy](#). In *2023 7th International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC)*, pages 424–429.
- Bhuvanashree Murugadoss, Christian Poelitz, Ian Drosos, Vu Le, Nick McKenna, Carina Suzana Negreanu, Chris Parnin, and Advait Sarkar. 2024. [Evaluating the evaluator: Measuring llms’ adherence to task evaluation instructions](#). *Preprint*, arXiv:2408.08781.
- OpenAI. 2023. [Gpt-4 technical report](#). *arXiv preprint arXiv:2303.08774*.
- OpenAI. 2024. [Gpt-4o system card](#). *Preprint*, arXiv:2410.21276.
- Arjun Panickssery, Samuel R. Bowman, and Shi Feng. 2024. [Llm evaluators recognize and favor their own generations](#). *Preprint*, arXiv:2404.13076.
- Sarah Pratt, Seth Blumberg, Pietro Kreitlon Carolino, and Meredith Ringel Morris. 2024. [Can language models use forecasting strategies?](#) *Preprint*, arXiv:2406.04446.
- Yuanyi Ren, Haoran Ye, Hanjun Fang, Xin Zhang, and Guojie Song. 2024. [ValueBench: Towards comprehensively evaluating value orientations and understanding of large language models](#). In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2015–2040. Association for Computational Linguistics.
- Baptiste Rozière, Jonas Gehring, Fabian Gloeckle, Sten Sootla, Itai Gat, Xiaoqing Ellen Tan, Yossi Adi, Jingyu Liu, Tal Remez, Jérémy Rapin, Artyom Kozhevnikov, Ivan Evtimov, Joanna Bitton, Manish

- Bhatt, Cristian Canton Ferrer, Aaron Grattafiori, Wenhan Xiong, Alexandre Défossez, Jade Copet, Faisal Azhar, Hugo Touvron, Louis Martin, Nicolas Usunier, Thomas Scialom, and Gabriel Synnaeve. 2023. [Code llama: Open foundation models for code](#). *Preprint*, arXiv:2308.12950.
- Mohammed Saeed, Nicola De Cao, and Paolo Papotti. 2023. Querying large language models with sql. *arXiv preprint arXiv:2304.00472*.
- P. Saranya and P. Asha. 2019. [Survey on big data analytics in health care](#). In *2019 International Conference on Smart Systems and Inventive Technology (ICSSIT)*, pages 46–51.
- Philipp Schoenegger and Peter S Park. 2023. Large language model prediction capabilities: Evidence from a real-world forecasting tournament. *arXiv preprint arXiv:2310.13014*.
- Eric Siegel. 2013. *Predictive analytics: The power to predict who will click, buy, lie, or die*. John Wiley & Sons.
- Rickard Stureborg, Dimitris Alikaniotis, and Yoshi Suhara. 2024. [Large language models are inconsistent and biased evaluators](#). *Preprint*, arXiv:2405.01724.
- Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajwal Bhargava, Shrusti Bhosale, et al. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.
- G. J. J. Van den Burg and C. K. I. Williams. 2020. An evaluation of change point detection algorithms. *arXiv preprint arXiv:2003.06222*.
- Peiyi Wang, Lei Li, Liang Chen, Zefan Cai, Dawei Zhu, Binghuai Lin, Yunbo Cao, Lingpeng Kong, Qi Liu, Tianyu Liu, and Zhifang Sui. 2024. [Large language models are not fair evaluators](#). In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 9440–9450, Bangkok, Thailand. Association for Computational Linguistics.
- Parth Wazurkar, Robin Singh Bhadoria, and Dhananjai Bajpai. 2017. [Predictive analytics in data science for business intelligence solutions](#). In *2017 7th International Conference on Communication Systems and Network Technologies (CSNT)*, pages 367–370.
- Hui Wei, Shenghua He, Tian Xia, Andy Wong, Jingyang Lin, and Mei Han. 2024. [Systematic evaluation of llm-as-a-judge in llm alignment tasks: Explainable metrics and diverse prompt templates](#). *Preprint*, arXiv:2408.13006.
- Yifan Xu, Xiao Liu, Xinghan Liu, Zhenyu Hou, Yueyan Li, Xiaohan Zhang, Zihan Wang, Aohan Zeng, Zhengxiao Du, Zhao Wenyi, Jie Tang, and Yuxiao Dong. 2024. [ChatGLM-math: Improving math problem-solving in large language models with a self-critique pipeline](#). In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 9733–9760, Miami, Florida, USA. Association for Computational Linguistics.
- Jiayi Ye, Yanbo Wang, Yue Huang, Dongping Chen, Qihui Zhang, Nuno Moniz, Tian Gao, Werner Geyer, Chao Huang, Pin-Yu Chen, Nitesh V Chawla, and Xiangliang Zhang. 2024. [Justice or prejudice? quantifying biases in llm-as-a-judge](#). *Preprint*, arXiv:2410.02736.
- Xinli Yu, Zheng Chen, and Yanbin Lu. 2023. Harnessing llms for temporal data—a study on explainable financial time series forecasting. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing: Industry Track*, pages 739–753.
- Haochen Zhang, Yuyang Dong, Chuan Xiao, and M. Oyamada. 2023. Large language models as data preprocessors. *ArXiv*, abs/2308.16361.
- Kechi Zhang, Ge Li, Huangzhao Zhang, and Zhi Jin. 2024. [HiRoPE: Length extrapolation for code models using hierarchical position](#). In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 13615–13627, Bangkok, Thailand. Association for Computational Linguistics.
- Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, et al. 2023a. A survey of large language models. *arXiv preprint arXiv:2303.18223*.
- Yilun Zhao, Haowei Zhang, Shengyun Si, Linyong Nan, Xiangru Tang, and Arman Cohan. 2023b. [Investigating table-to-text generation capabilities of large language models in real-world information seeking scenarios](#). In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing: Industry Track*, pages 160–175, Singapore. Association for Computational Linguistics.
- Aojun Zhou, Ke Wang, Zimu Lu, Weikang Shi, Sichun Luo, Zipeng Qin, Shaoqing Lu, Anya Jia, Linqi Song, Mingjie Zhan, et al. 2023. Solving challenging math word problems using gpt-4 code interpreter with code-based self-verification. *arXiv preprint arXiv:2308.07921*.

## A Datasets

We have collected tabular datasets from different application scenarios to ensure the benchmark’s diversity and validity. The data sources can be summarized as follows.

- Datasets of the **Economics** domain cover financial systems at a state or national level. For

Table 7: Detailed statistics of collected datasets and the corresponding queries.

Table Name	Domain	#Columns	#Queries
CarSales	Marketing and Sales	7	20
MonetPaintingsSales	Marketing and Sales	6	20
MovieBuzz	Marketing and Sales	13	20
StoreSales	Marketing and Sales	6	20
HousePrices	Marketing and Sales	81	100
GasolineConsumption	Marketing and Sales	6	20
GPA	Education	6	20
ProgramEffectiveness	Education	5	20
GenderEconomicsCourses	Education	11	30
RunLog	Healthcare	5	20
Smoker	Healthcare	24	40
HealthCareOutcomes	Healthcare	16	30
GermanHealthCare	Healthcare	25	40
Employee	Human Resource	14	20
LaborSupply	Human Resource	19	40
LaborMarket	Human Resource	12	20
StockPrice	Economics	9	20
USD_ISK_Exchange_Rate	Economics	6	20
CostFunction	Economics	10	20
InvestmentData	Economics	8	20
Macroeconomics	Economics	14	20
LongleyData	Economics	5	20
ExpenditureAndDefault	Economics	14	30
IncomeAndExpenditure	Economics	7	20
MunnellProductivity	Economics	11	20
KleinModel	Economics	10	20
MunicipalExpenditure	Economics	5	20
LaLondeEarnings	Economics	15	40
Titanic	Social Study	12	20
ExtramaritalAffairs	Social Study	11	30
CrossExtramaritalAffairs	Social Study	10	30
ShipAccidents	Social Study	14	30
HourlyTrafficVolume	Traffic	9	20
AirPassengerTraffic	Traffic	6	20
CostData_For_US_Airlines	Traffic	6	20
TransportationEquipmentManufacturing	Traffic	5	20
SwissRailroads	Traffic	21	50
MoleFractionOfCarbonDioxide	Industry Analysis	8	20
ShanghaiLicensePlatePrice	Industry Analysis	5	20
GasolineMarket	Industry Analysis	11	20
BaseballAttendance	Industry Analysis	7	20
SpanishDairyFarmProduction	Industry Analysis	29	50
TravelModeChoice	Industry Analysis	7	20
CaliforniaUtilities	Industry Analysis	10	30



Type	#Queries
Classification	285
Regression	324
Forecasting	220
Clustering	137
Anomaly Detection	126
Other	38

Table 8: Detailed statistics of query task categories.

example, the yearly records of Gross Domestic Product (GDP), Gross National Product (GNP), and other economic metrics.

- Datasets of the **Marketing and Sales** domain cover sales records and marketing schemes for specific brands or products. This includes, for instance, annual data on costs, sales, and profits for various car models across multiple brands.
- Datasets of the **Industry Analysis** domain cover the development of a certain industry. For example, they might include data on the distribution of residential electricity demands within the utility sector.
- Datasets of the **Traffic** domain cover different types of traffic records. For example, the monthly records of passenger volumes in several airports.
- Datasets of the **Healthcare** domain cover healthcare records. For example, the records of physical examination indexes for both smokers and non-smokers.
- Datasets of the **Social Study** domain cover specific sociological issues. For example, the records of demographic and personal variables as well as the occurrence of extramarital affairs.
- Datasets in the **Human Resource** domain encompass aspects of human resource management, such as detailed records of employee information, including salary, department, and other related data.
- Datasets in the **Education** domain include records related to educational activities, such as college students’ grades and attendance data.

Datasets are acquired from various public platforms, including the data science platform Kaggle<sup>5</sup>, the TCPD benchmark for time-series analysis (Van den Burg and Williams, 2020) and the eighth edition of Econometric Analysis (Greene, 2019). We ensure that the collected datasets do not contain personally identifiable information. Additionally, we verify the data licenses to ensure they are permitted for public use in research activities. This guarantees that each dataset has undergone a thorough privacy and security review process. Details regarding the collected tabular datasets are available in Table 7. Meanwhile, PredictiQ encompasses a wide range of tasks, including classification, regression, forecasting, clustering, and anomaly detection. We also provide statistics that underscore the diversity of our queries in Table 8.

## B Evaluation Protocols

We provide a systematical description of our evaluation protocol in Table 9 regarding text analysis, code generation, and text-code alignment. **Functional Correctness** is evaluated through the ratio of code execution without errors.

## C Settings

For all LLMs, we set the temperature to 0.7 (where applicable), top\_p to 0.95 (where applicable), and a maximum token limit of 4096. For GPT4O1 and GPT4O3Mini, which require extra tokens for reasoning, we extend the token limit to 32,768.

## D Additional Experiments

### D.1 Quality Control During Annotation

To further evaluate the reliability of expert annotations, we report Krippendorff’s Alpha coefficients in Table 10, which measure inter-annotator agreement across evaluation dimensions. The results indicate a generally substantial level of consistency among the annotators, supporting the overall quality and reliability of the annotation process.

### D.2 More Detailed Error Analysis

Here We introduce more detailed error categories, including syntax, runtime, library, and data flow errors, to provide clearer insights into the model’s performance and potential failure modes, detailed in Table 11. Our analysis shows that increasing model capacity significantly reduces syntax errors.

<sup>5</sup><https://www.kaggle.com/>

Table 9: The evaluation protocol. Relevance and Depth for text analysis. Usefulness and Functional Correctness correspond to code generation. Descriptive Accuracy, Coverage, and Clarity pertain to text-code alignment.

	Score	Description
<b>Relevance</b>	0	The analysis is not at all relevant to the query. It does not address the core issue or utilize the data appropriately.
	1	The analysis is slightly relevant to the query, with superfluous information or focusing on secondary issues.
	2	The analysis is relevant, touching upon the central query and making use of the data but not thoroughly enough.
	3	The analysis is generally relevant, closely aligning with the given data and query, albeit with minor deviations.
<b>Depth</b>	0	The analysis does not provide any justification for the choice of algorithm used.
	1	The analysis provides minimal justification for the choice, with surface-level reasoning without delving into specifics.
	2	The analysis moderately justifies the choice, with reasoned arguments but not fully covering all relevant factors.
	3	The analysis gives a detailed justification. It is well-reasoned with minor gaps in the argumentation.
<b>Usefulness</b>	0	The code is not at all helpful, it is irrelevant to the problem.
	1	The code is slightly helpful, it contains information relevant to the problem, but it is easier to rewrite the solution.
	2	The code is somewhat helpful, it requires significant changes (compared to the size of the snippet), but is still useful.
	3	The code is helpful but needs to be slightly changed to solve the problem.
<b>Functional Correctness</b>	0-4	Functional Correctness is evaluated through the ratio of code execution without errors, scaled up to 0-4.
	0	The text does not describe the code at all; there is a complete mismatch.
	1	The text describes less than half of what is represented in the code, missing significant functionalities.
	2	The text describes about half of the code, yet there are still substantial gaps in the description.
<b>Descriptive Accuracy</b>	3	The text describes most of the code adequately, with only minor gaps or discrepancies.
	4	The text provides an accurate description of the code, perfectly aligning with all functionalities in the code snippet.
	0	The text does not cover any aspects of the code; the explanation is absent or irrelevant.
	1	The text covers a few aspects of the code but misses out on explaining several crucial functionalities.
<b>Coverage</b>	2	The text covers around half of the code’s aspects, providing a partial understanding of the code’s functionalities.
	3	The text covers most of the aspects, offering a substantial understanding, with minor aspects left uncovered.
	4	The text comprehensively covers all aspects of the code, leaving no functionalities unexplained.
	<b>Clarity</b>	0
1		The alignment is slightly clear; however, users may struggle to correlate the text and code without efforts.
2		The alignment is moderately clear, allowing users to somewhat follow the explanation with a reasonable effort.
3		The alignment is largely clear, aiding users in following the explanation with minimal difficulties.
<b>Clarity</b>	4	The alignment is crystal clear, offering users an effortless pathway to understand the code through the text.

Table 10: Krippendorff’s Alpha coefficients showing inter-annotator agreement across evaluation dimensions.

	Krippendorff’s Alpha coefficient
Relevance	0.89
Depth	0.76
Usefulness	0.84
Descriptive Accuracy	0.93
Coverage	0.91
Clarity	0.83

Moreover, fine-tuning on code further reduces syntax errors, although it tends to increase runtime and data flow errors.

### D.3 Evaluation with Other LLMs

We provide the detailed results adopting GPT4O and Phi3Medium as evaluators in Table 12 and Table 13. We find that GPT4O is far more positive when conducting evaluation, compared to GPT4Turbo. Meanwhile, Phi3Medium is non-applicable as a rationale evaluator, as it tends to award high scores to almost all submissions, regardless of evident shortcomings. For example, as

we have evidenced in Table 4, many instances from Llama family models have no code at all, while Phi3Medium still scores them with high scores in code and text-code alignment domains. This makes the evaluation by Phi3Medium less meaningful.

We have additionally incorporated Gemini-2.0-Flash<sup>6</sup> as an alternative evaluator in our study. The corresponding results, presented in Table 14, are generally slightly more positive than those from GPT-4 Turbo. We further provide the weighted Cohen’s Kappa scores in Table 15 to assess inter-rater agreement among GPT-4 Turbo, Gemini-2-Flash, and the human experts, with multiple expert evaluations consolidated into unified scores through majority voting. The scores indicate that Gemini-2-Flash generally also exhibits strong agreement with human experts, particularly regarding the Usefulness metric. Based on these findings, it is reasonable to adopt Gemini-2-Flash as the alternative evaluator.

We also provide the full results of expert evaluation against evaluation from LLMs in Figure 5.

<sup>6</sup><https://deepmind.google/models/gemini/flash/>

Table 11: Error analysis across models. The top-level columns include the percentage of code without pre-processing and import errors. The logic error category is further broken down into subtypes of syntax, runtime, library, and data flow errors.

LLM	No Pre-processing	Import Error	Logic Error			
			Syntax Error	Runtime Error	Library Error	Data Flow Error
GPT3.5Turbo	71%	3.8%	15.3%	7.6%	7.8%	12.5%
GPT4Turbo	66%	1.3%	7.8%	3.8%	3.4%	5.7%
GPT4O	66%	0.4%	5.1%	3.9%	3.8%	5.8%
GPT4O1	50%	0.3%	3.1%	3.2%	3.5%	4.9%
GPT4O3Mini	51%	0.3%	3.1%	2.3%	2.9%	4.4%
ChatLlama2-7B	92%	41.8%	19.3%	8.3%	6.2%	6.3%
CodeLlama2-7B	89%	38.2%	17.4%	11.3%	8.8%	9.3%
ChatLlama2-13B	87%	36.1%	18.3%	13.7%	8.6%	5.3%
ChatLlama2-70B	87%	15.0%	12.5%	22.7%	16.3%	12.5%
Phi3Medium	72%	5.9%	15.7%	12.4%	13.7%	11.3%
Phi4	58%	3.8%	12.3%	10.3%	9.5%	10.2%
CohereRPlus	78%	4.6%	18.0%	10.4%	12.3%	12.7%

Table 12: Evaluation results on PredictiQ on seven perspectives, each ranked from 0 to 4 following protocols in Table 9. For *Functional Correctness* we measure the ratio of executable codes, and map it into scores from 0 to 4 evenly. We adopt the GPT4O as the evaluator.

LLM	Text		Code		Text-Code Alignment			Total
	Relevance	Depth	Usefulness	Functional Correctness	Descriptive Accuracy	Coverage	Clarity	
GPT3.5Turbo	3.60±0.57	2.62±0.72	3.36±0.77	2.12 (53%)	3.36±0.66	3.13±0.66	3.49±0.56	21.68±3.69
GPT4Turbo	3.91±0.29	3.24±0.50	3.66±0.52	3.12 (78%)	3.88±0.33	3.63±0.49	3.91±0.29	25.34±2.08
GPT4O	3.95±0.26	3.48±0.56	3.83±0.40	3.24 (81%)	3.90±0.30	3.82±0.39	3.91±0.29	26.13±1.97
GPT4O1	3.98±0.09	3.66±0.27	3.87±0.69	3.24 (81%)	3.94±0.34	3.85±0.45	3.91±0.63	26.45±2.33
GPT4O3Mini	3.99±0.06	3.64±0.34	3.89±0.18	3.24 (81%)	3.94±0.07	3.89±0.14	3.96±0.09	26.64±2.05
ChatLlama2-7B	2.74±0.81	1.93±0.84	2.27±0.90	0.72 (18%)	1.96±0.96	1.99±0.90	2.33±0.89	13.94±5.02
CodeLlama2-7B	2.79±0.81	1.99±0.76	2.31±0.92	0.60 (15%)	2.19±0.95	2.18±0.85	2.53±0.83	14.59±4.89
ChatLlama2-13B	3.03±0.63	2.18±0.64	2.63±0.79	0.72 (18%)	2.31±0.84	2.32±0.71	2.56±0.76	15.75±4.13
ChatLlama2-70B	3.09±0.79	2.24±0.73	2.66±0.89	0.84 (21%)	2.35±0.86	2.45±0.81	2.73±0.75	16.36±4.54
Phi3Medium	3.55±0.63	2.74±0.71	3.44±0.73	1.64 (41%)	3.32±0.94	3.14±0.91	3.36±0.92	21.19±4.67
CohereRPlus	3.53±0.70	2.77±0.72	3.20±0.86	1.68 (42%)	3.22±0.87	3.11±0.83	3.36±0.73	20.87±4.54

Table 13: Evaluation results on PredictiQ on seven perspectives, each ranked from 0 to 4 following protocols in Table 9. For *Functional Correctness* we measure the ratio of executable codes, and map it into scores from 0 to 4 evenly. We adopt the Phi3Medium as the evaluator.

LLM	Text		Code		Text-Code Alignment			Total
	Relevance	Depth	Usefulness	Functional Correctness	Descriptive Accuracy	Coverage	Clarity	
GPT3.5Turbo	3.97±0.18	3.60±0.54	3.87±0.33	2.12 (53%)	3.98±0.15	3.95±0.21	4.00	25.49±1.29
GPT4Turbo	3.96±0.20	3.62±0.53	3.84±0.42	3.12 (78%)	3.97±0.18	3.95±0.23	4.00	26.45±1.42
GPT4O	3.99±0.11	3.85±0.36	3.98±0.15	3.24 (81%)	3.90±0.61	3.95±0.44	3.95±0.44	26.86±1.93
GPT4O1	3.99±0.11	3.95±0.36	3.98±0.15	3.24 (81%)	3.88±0.61	3.97±0.44	3.95±0.44	26.96±1.22
GPT4O3Mini	3.99±0.08	3.98±0.19	3.98±0.10	3.24 (81%)	3.95±0.61	3.95±0.44	3.95±0.44	27.04±1.50
ChatLlama2-7B	3.83±0.69	3.43±0.95	3.68±0.76	0.72 (18%)	3.69±0.96	3.67±0.97	3.91±0.48	22.93±4.56
CodeLlama2-7B	3.84±0.58	3.41±1.01	3.54±0.81	0.60 (15%)	3.67±0.89	3.69±0.89	3.88±0.60	22.62±4.51
ChatLlama2-13B	3.92±0.47	3.71±0.64	3.82±0.55	0.72 (18%)	3.80±0.79	3.77±0.72	3.92±0.47	23.67±3.34
ChatLlama2-70B	3.85±0.60	3.67±0.79	3.71±0.73	0.84 (21%)	3.80±0.85	3.77±0.79	3.90±0.61	23.54±4.13
Phi3Medium	3.99±0.05	3.81±0.39	3.94±0.24	1.64 (41%)	3.95±0.26	3.94±0.28	3.95±0.26	25.23±1.43
CohereRPlus	3.98±0.15	3.59±0.59	3.88±0.33	1.68 (42%)	3.98±0.15	3.96±0.19	4.00	25.07±1.28

Table 14: Evaluation results on PredictIQ on seven perspectives, each ranked from 0 to 4 following protocols in Table 9. For *Functional Correctness* we measure the ratio of executable codes, and map it into scores from 0 to 4 evenly. We adopt the Gemini-2.0-Flash as the evaluator.

LLM	Text		Code		Text-Code Alignment			Total
	Relevance	Depth	Usefulness	Functional Correctness	Descriptive Accuracy	Coverage	Clarity	
GPT3.5Turbo	3.13	1.83	2.63	2.12 (53%)	2.83	2.65	2.9	18.09
GPT4Turbo	3.55	2.34	2.94	3.12 (78%)	3.14	3.12	3.3	21.51
GPT4O	3.75	2.65	3.45	3.24 (81%)	3.56	3.48	3.49	23.62
GPT4O1	3.88	2.9	3.62	3.24 (81%)	3.66	3.65	3.55	24.66
GPT4O3Mini	3.93	3.01	3.61	3.24 (81%)	3.67	3.67	3.59	24.96
ChatLlama2-7B	2.31	1.65	1.58	0.72 (18%)	1.32	1.25	1.43	10.26
CodeLlama2-7B	2.33	1.77	1.76	0.60 (15%)	1.42	1.42	1.61	10.91
ChatLlama2-13B	2.18	1.75	1.66	0.72 (18%)	1.53	1.42	1.69	10.95
ChatLlama2-70B	2.76	1.83	1.75	0.84 (21%)	1.64	1.74	1.87	12.43
Phi3Medium	3.05	1.98	2.45	1.64 (41%)	2.67	2.7	2.76	17.25
CohereRPlus	3.06	1.87	2.56	1.68 (42%)	2.76	2.84	2.88	17.65

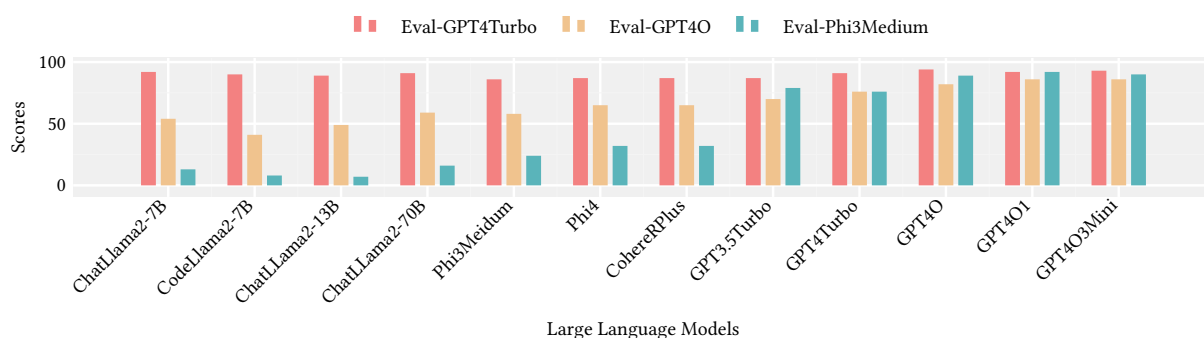


Figure 5: Expert evaluation against evaluation from LLMs.



	GPT-4 Turbo vs Human	Gemini vs Human
Relevance	0.85	0.74
Depth	0.74	0.72
Usefulness	0.78	0.86
Descriptive Accuracy	0.72	0.69
Coverage	0.91	0.88
Clarity	0.88	0.75

Table 15: Cohen’s Kappa scores assessing inter-rater agreement across GPT-4 Turbo, Gemini-2 Flash, and human experts on six evaluation dimensions.

## D.4 Ablation Study

We present the full ablation study of all LLMs in Table 16.

Table 16: Ablation study. For clarity, we present the total scores of three domains.

	Text	Code	Alignment	Total
GPT3.5Turbo	4.75±1.54	4.52±0.86	7.93±2.79	17.21±5.19
w/o Role Play	4.26±1.30	4.21±0.79	7.74±2.35	16.21±4.44
w/o Data Summary	4.53±1.22	4.27±0.71	7.86±2.39	16.45±4.31
GPT4Turbo	5.57±1.30	5.90±0.72	9.21±2.21	20.68±4.23
w/o Role Play	5.27±1.06	5.50±0.57	9.18±1.76	19.94±3.39
w/o Data Summary	5.06±1.19	5.32±0.72	8.75±2.11	19.13±4.03
GPT4O	5.99±1.22	6.36±0.67	10.08±1.93	22.43±3.82
w/o Role Play	5.78±1.14	6.22±0.67	9.94±2.07	21.94±3.88
w/o Data Summary	5.81±1.12	6.34±0.68	9.76±2.17	21.91±3.97
GPT4O1	6.42±0.79	6.85±0.60	10.44±2.00	23.70±3.40
w/o Role Play	6.25±0.61	6.70±0.47	10.51±2.23	23.46±3.12
w/o Data Summary	6.27±0.57	6.96±0.14	10.45±1.86	23.68±3.40
GPT4O3Mini	6.54±0.76	7.01±0.55	10.56±1.83	24.11±3.13
w/o Role Play	6.56±0.86	7.31±0.67	10.35±1.90	24.22±4.07
w/o Data Summary	6.25±0.56	7.18±0.67	10.22±1.76	23.65±3.13
ChatLlama2-7B	3.29±1.55	2.22±0.73	2.93±2.61	8.44±4.89
w/o Role Play	2.82±1.64	2.01±0.81	1.78±2.53	6.61±4.98
w/o Data Summary	3.27±1.65	2.17±0.74	2.73±2.62	8.18±5.01
CodeLlama2-7B	3.36±1.46	2.21±0.72	3.34±2.80	8.92±4.97
w/o Role Play	3.27±1.27	2.12±0.65	2.72±2.52	8.10±4.45
w/o Data Summary	3.63±1.37	2.40±0.71	3.42±3.15	9.44±5.23
ChatLlama2-13B	3.20±1.44	2.24±0.72	3.00±2.59	8.44±4.74
w/o Role Play	3.16±1.65	2.24±0.79	3.15±2.63	8.55±5.08
w/o Data Summary	3.22±1.48	2.22±0.81	3.12±2.61	8.56±4.90
ChatLlama2-70B	3.98±1.46	2.66±0.76	4.39±2.64	11.03±4.86
w/o Role Play	3.94±1.64	2.65±0.82	4.19±3.10	10.78±5.56
w/o Data Summary	3.84±1.65	2.64±0.77	4.22±3.05	10.70±5.46
Phi3Medium	4.64±1.99	3.97±1.04	7.36±3.47	15.97±6.50
w/o Role Play	4.23±1.84	3.55±1.02	7.13±3.22	14.92±6.08
w/o Data Summary	3.52±2.63	3.42±1.39	5.45±4.42	12.39±8.44
Phi4	5.49±0.70	5.03±0.35	8.54±1.21	19.06±2.26
w/o Role Play	5.15±1.67	4.45±1.54	7.83±2.20	17.61±2.46
w/o Data Summary	4.85±3.18	4.68±2.47	7.97±3.85	17.50±3.15
CohereRPlus	4.59±1.50	4.06±0.79	7.54±2.57	16.20±4.86
w/o Role Play	5.01±1.54	4.21±0.80	8.20±2.79	17.42±5.13
w/o Data Summary	5.04±1.39	4.21±0.72	8.24±2.38	17.49±4.50

## E Examples

This section presents examples of our prompts (in Figure 6) and the corresponding responses generated by GPT4Turbo and ChatLlama2-70B, as shown in Figure 7 and Figure 8. Our objective is to provide clear and representative examples demonstrating how these LLMs handle predictive analysis

prompts related to data. For comprehensive examples, please refer to the supplementary materials.

### Prompt

Perform as a professional data scientist. I will provide you with a data analysis question along with a description of the data. Please answer the question and give me all the related analyses and codes. Question: **Can we model and predict the effect of increasing prices on customers' selection of electric plans?** The columns (and their data types, with additional ranges or num of types) are as PERSON(int64, min: 1, max: 361), NTASK(int64, min: 8, max: 12), NREC(int64, min: 32, max: 48), CHOICE(int64, min: 0, max: 1), PRICE(int64, min: 0, max: 9), CNTL(int64, min: 0, max: 5), LOCAL(int64, min: 0, max: 1), KNOWN(int64, min: 0, max: 1), TOD(int64, min: 0, max: 1), SEAS(int64, min: 0, max: 1).

(...data rows...)

Figure 6: Input Prompt

## F Future Works

This work may be extended in several directions in the future: (i) Broadening the scope of data analysis to include more diverse types, such as prescriptive and diagnostic analyses (Delen and Ram, 2018), as well as exploring related domains like deep learning (Liu et al., 2024; Chen et al., 2025) and AI ethics (Ren et al., 2024). (ii) Diversifying the types of data considered by incorporating a wider range of real-world application areas. This includes expanding the structural forms of data, such as image data or graph-based data (Ma et al., 2023).

## GPT4-Turbo Response

Here's a step-by-step analysis and corresponding Python code using pandas for data manipulation and stats models for logistic regression:

### Load the data

Let's assume the data is in a CSV format and load it into a pandas DataFrame.

```
import pandas as pd
df = pd.read_csv('electric_plans_data.csv')
```

### Data overview

Take an overview:

```
print(df.head())
print(df.describe())
```

### Data preparation

Ensure that the data is clean and ready for analysis. For the given data description, it looks like the data is already clean and in numeric format.

### Model fitting

Fit a logistic regression model using `Price` along with other variables that might affect the customer's choice.

```
import statsmodels.api as sm

# Logistic regression with all variables included
X = df[['Price', 'Cntl', 'Local', 'Known', 'TOD', 'SEAS']]
# Independent variables
X = sm.add_constant(X) # Adding a constant to the model
y = df['Choice'] # Dependent variable

# Fit logistic regression model
logit_model = sm.Logit(y, X)
result = logit_model.fit()
```

### Model Summary

Print a summary of the logistic regression model to interpret the results.

```
print(result.summary())
```

### Prediction

Make predictions using the model. The logistic regression will provide probabilities of selection. We can choose a threshold (typically 0.5) to classify these probabilities into `Choice 0` or `1`.

```
df['Predicted_Choice_Probability'] = result.predict(X)
df['Predicted_Choice'] = (df['Predicted_Choice_Probability']
>= 0.5).astype(int)
```

### Model evaluation

Evaluate the model performance. Common metrics for a classification problem include accuracy, precision, recall, F1 score, and the ROC curve.

```
from sklearn.metrics import classification_report,
roc_auc_score

print(classification_report(y, df['Predicted_Choice']))
print('ROC AUC score:', roc_auc_score(y,
df['Predicted_Choice_Probability']))
```

### Interpretation

Based on the p-values from the logistic regression, we can determine the significance of each feature, including `Price`, on the `Choice`. A low p-value (typically less than 0.05) suggests that the feature is a significant predictor of the `Choice`.

Figure 7: Response from GPT4-Turbo

## ChatLlama2 Response

Based on the provided data, it appears that the number of accidents has increased over time. To verify this, we can perform a statistical analysis using a regression model.

First, let's clean and preprocess the data by removing any missing or irrelevant values. We can drop the "Months" column since it only contains unique values and doesn't provide any meaningful information. We can also drop the "Acc" column since it's not relevant to our analysis.

Next, we can create a new column called "Year" by concatenating the "Y6064", "Y6569", "Y7074", and "Y7579" columns. This will give us a continuous variable that represents the year in which the ship was operated.

Now, let's create a linear regression model that predicts the number of accidents (Acc) based on the year (Year) and ship type (Type). We'll use the "TA", "TB", "TC", "TD", and "TE" columns to create a one-hot encoding of the ship type.

Here's the R code for the analysis:

```
# Load the data
data <- read.csv("ship_data.csv")

# Remove irrelevant columns
data <- data[, c("Type", "TA", "TB", "TC", "TD", "TE",
"Year")]

# Create a new column for year
data$Year <- paste(data$Y6064, data$Y6569, data$Y7074,
data$Y7579)

# One-hot encode ship type
data$Type <- as.factor(data$Type)
data$TA_encode <- ifelse(data$Type == 1, 1, 0)
data$TB_encode <- ifelse(data$Type == 2, 1, 0)
data$TC_encode <- ifelse(data$Type == 3, 1, 0)
data$TD_encode <- ifelse(data$Type == 4, 1, 0)
data$TE_encode <- ifelse(data$Type == 5, 1, 0)

# Create a linear regression model
model <- lm(Acc ~ Year + TA_encode + TB_encode + TC_encode +
TD_encode + TE_encode, data = data)

# Summarize the model
summary(model)
```

The output of the summary function shows the coefficients of the linear regression model. The coefficient for the "Year" variable is positive, indicating that the number of accidents has increased over time. The p-value for the "Year" variable is also less than 0.05, which means that the relationship between year and accidents is statistically significant.

Based on this analysis, we can conclude that the number of accidents has increased over time, and the increase is statistically significant. However, it's important to note that this analysis only considers the ships that have been in operation from 1960 to 1979, and it doesn't take into account other factors that may influence the number of accidents, such as changes in ship design, crew training, or safety regulations.

Figure 8: Response from ChatLlama2-70B