University of Edinburgh

GAIL Seed Funded Project Report

Large Language Models: Are They Ready for Data Science?

PI: Ozan Evkaya
Collaborators: Miguel de Carvalho, Wenqi Ni

Long version Report

July 31, 2025

Abstract

As a result of recent advancements in generative AI (Gen-AI), the field of Data Science (DS) is prone to various changes. Data Analysis (DA) capabilities of the ChatGPT and its competitors are getting more powerful based on the recent developments in the field of Gen-AI. While DA provides researchers and practitioners with unprecedented analytical capabilities, it is far from being perfect, and it is important to recognize and address its limitations. The random nature of LLMs can mislead practitioners for certain parts of the Data Analysis components, especially the statistics-based interpretations and final decisions.

A key goal of this project is to evaluate the performance of the OpenAI Advanced Data Analytics (ADA) tool on DS tasks (e.g., data wrangling, exploratory analyses, fitting models) For that purpose, OpenAI Code Interpreter assistant will be used for the specific tasks repeatedly by mimicking the idea of Monte Carlo experiments borrowed from the field of Statistics. By incorporating the Code Interpreter within GPT assistant, using zero-shot style prompts, the project aims to measure the performance of this tool numerically alongside the automation of DS discussions. To give a solid measure of the performance, certain aspects of the generated outcome (i.e. correctness, text similarity, verbosity) will be mainly focused on.

In the literature, experiments done so far are mainly limited to a certain number of queries mostly and not resembling the idea of repeated sampling, necessary for the uncertainty measurement on top of the generated responses from GPT's ADA extension. To overcome this issue, by following different strategies (different data set file upload, specific Data Science related prompts and more advanced statistical concepts explanation), the project will capture a comprehensive overview of the certain behavior of the DA capabilities of GPT models. For the tool development, OpenAI platform instructions have been considered to create a new DA-capable GPT within a personal execution environment. Using the curated tasks from the specific parts of the four DS quadrants mentioned in the work of De Bie et al. (2022), the project aims to contribute the discussions about the automation of the Data Science field with empirical Gen-AI based results over curated questions.

Keywords: Large Language Models (LLMs), Data Science (DS), Code Interpreter, Data Analysis, Performance Metrics

Contents

Chapter 1

Introduction

In this project, we aim to create a Code-Interpreter based GPT assistant via OpenAI API usage to examine certain data science related tasks. Experiments done so far mainly limited to a certain number of queries and do not resemble the idea of repeated sampling, necessary for the uncertainty measurement on top of the generated responses from GPT's ADA extension. To overcome this issue, by following different strategies (different data set file upload, specific Data Science related prompts and more advanced statistical concepts explanation), the project captures a comprehensive overview of the certain behavior of the DA capabilities of GPT via Code Interpreter. Besides, over various data science tasks, the results examine the impact of various experimental setting such as the base-model or model parameter (mainly temperature) change.

1.1 Background

The OpenAI's ADA extension (as tool to call) of ChatGPT, also known as the Code Interpreter, has proven to be a transformative tool in Data Science, facilitating tasks from data cleaning and analysis to complex statistical modeling. Studies show that ADA enables a new level of accessibility, allowing users, even those with limited coding skills, to interact with data through natural language prompts. This tool may serve for various reasons including simulation of the data or optimizing data analysis pipelines but not limited to these.

Large Language Models (LLMs) have demonstrated impressive capabilities across a range of scientific tasks including mathematics, physics, and chemistry. Despite their successes, the effectiveness of LLMs in handling complex statistical tasks remains systematically under-explored. Briefly, the model's dependency on prompt specificity can lead to errors if users are not skilled at framing detailed prompts. Additionally, while GPT's ADA can often match or even outperform human analysts in basic tasks, it may struggle with contextually complex or nuanced analyses (Ping et al., 2020). It has been claimed recently that ADA is unable to make careful assumptions and claims when providing insights, unlike professional human data analysts (Bayne, 2015). Besides, inherited randomness of these LLMs can add certain level of uncertainty to the created responses even if the same prompt is asked repeatedly (Evkaya and de Carvalho, 2024).

1.2 Aims and objectives

We systematically experiment with representative LLMs using a set of questions from various topics in the statistics and data science fields. For that purpose, we introduced DS-QA as a small new benchmark, designed for the project purposes, covering various concepts at different difficulty levels. Alongside the repeated experimentation, we aimed to explore the followings within the original goal of the project;

- RQ1: To what extent can GPT based LLMs produce consistent responses over a certain number of repeated prompts, how to quantify the differences numerically?
- RQ2: In which task, which common grounds are important to be part of the evalation process, including main numerical characteristics and properties?
- RQ3: How the based model or model parameter (namely change on the temperature) can effect the created responses, in terms of the corresponding metrics?

1.3 Source of Questions

While creating the DS-QA source, we benefited from the previously considered data set oriented questions and answers from different undergraduate level modules, including

- Introduction to Data Science (IDS) course weekly lab exercises, homework assignments varying on different data sets in certain formats and data sets from some R packages directly
- Some statistical-related questions from our Statistics Year 2 (StatY2) from weekly labs/quiz exercises from previous years
- Some questions, such as Principal Component Analysis (PCA), Clustering, or Bayesian modelling, have different complexity from specific modeling landscape.
- IDS quiz examples including interpretational type of questions without any specific data file

1.3.1 Selected Task types

By considering the available sources and the certain gaps in the available studies, we focused primarily on the tasks related to the four data science quadrants mentioned earlier by Tjil De Bie et al. (2022) to cover question-answer pairs from the dimensions called as (i) Data Engineering, (ii) Data Exploration, (iii) Model building and (iv) Exploitation. Among those, our curated questions can be classified under the following main keywords as their concept names:

- Data Undertanding and Wrangling
- Data Cleaning, Pre-processing and Transformations
- Data Summary Statistics and Interpretations
- Data Visualization - EDA
- Statistical Models
    - Supervised Learning (Linear Regression modeling, Variations of regression (ie. Lasso or Ridge), Classification models such as Logistic Regression model, Tree based model, Model comparison type questions, Bayesian Regression)

  - Unsupervised Learning (ie. Principal Component Analysis (PCA), Clustering methods)

- Statistical hypothesis testing-related questions

Overall the created questions may range on the above given list in terms of the considered keywords, as in Table 1.1. In general, most of the curated questions targeted different types of statistical modeling, ranging over supervised or unsupervised learning tools. Besides, Data Summary statistics and data visualization-related questions are on the second rank, with limited items on data cleaning or hypothesis testing concepts.

Category	**Frequency**
Statistical Models	37
Data Summary Statistics and Interpretations	21
Data Visualization - EDA	18
Data Understanding and Wrangling	13
Data Cleaning, Pre-processing and Transformations	6
Statistical Hypothesis Testing	3
Total	100

Table 1.1: Summary of Question Concepts by Category

1.3.2 Difficulty levels

Similar to the available literature, the created DS-QA benchmark items are manually tagged under three difficulty levels by considering their clarity and complexity. For that purpose, our question-answer pairs are labeled under (i) easy, (ii) medium or (iii) hard level of difficulties to use as additional information. This labeling process has been completed via the concept information, and content of the question. Overall, the task and difficulty categories of the curated DS-QA can be summarized as in Table 1.2, across different data sets that are used in this project.

Level	Count	Percentage
easy	22	22.0%
medium	39	39.0%
hard	39	39.0%
Total tasks	100	100%

Table 1.2: Distribution of tasks by difficulty level

More detailed question summary across different data sets and difficulty levels can be summarized as follows in Table 1.3. For the corresponding question list, details can be followed in the main project GitHub repository `https://github.com/oevkaya/GAIL-DS-project`.

As shown in Table 1.3, 15 different data sets are considered across different questions and difficulty levels in total. In one case, the nature of the created questions required two merged data sets (edibnb.csv, council_assessments.csv) but other questions relied on one specific data set having specific formats such as .csv, .xlsx, or .txt.

Dataset	Question IDs	Difficulty Level
diamonds.xlsx	15.1, 16.0, 16.1, 17.0, 15.0, 18.0, 19.0, 20.0	'hard': 4, 'medium': 4
Edinburgh_rainfall.csv	69, 69.1, 69.2, 69.3, 69.4	'hard': 4, 'medium': 1
gss16_advfront.csv	37.0, 38.1, 39.1, 38.0, 39.0	'medium': 1, 'hard': 3
Gene_Data.xlsx	66, 67, 67.1, 67.2	'medium': 2, 'hard': 2
edibnb.csv, council_assessments.csv	31.0, 32.0, 33.0	'easy': 2, 'hard': 1
cherryblossom_run17.xlsx	34.0, 34.1, 35.0, 36.0	'medium': 1, 'hard': 3
mouse.txt	53.0, 54.0	'medium': 1, 'easy': 1
laptop_data_cleaned.csv	46.0, 47.1, 48.0, 48.1, 47.0, 49.0, 50, 51, 70, 72, 73	'easy': 1, 'medium': 6, 'hard': 4
instructional-staff.csv	14.2, 14.0, 14.1	'easy': 1, 'medium': 2
weatherAUS.csv	27, 28, 29, 30, 30.1, 30.2, 71	'easy': 1, 'medium': 3, 'hard': 3
aeroplane.txt	56.0, 57.0, 58.0	'medium': 2, 'hard': 1
evals.csv	21.0, 22.0, 23.0, 23.1, 23.2, 24.0, 26, 26.1	'medium': 6, 'hard': 3
joined_plastic_data_all.csv	10.0, 11.0, 12.0, 13.0	'easy': 2, 'medium': 2
duke_forest.xlsx	59, 60, 61, 61.1, 61.2, 62, 63, 64, 65, 65.1, 65.2	'easy': 2, 'medium': 2, 'hard': 7
UK-visitor-numbers.csv	0.0, 1.0, 2.0, 3.0, 4.0, 5.0, 6.0, 7.0, 8.0, 9.0	'easy': 4, 'medium': 4, 'hard': 2

Table 1.3: Summary of datasets, associated question IDs, and difficulty distribution

1.4 Summary of contributions and achievements

Throughout this project, we contributed to the recent studies around the LLM investigation for the purpose of solving certain data science / statistics related tasks. Mainly, the project contributions can be classified under these themes;

1. DS-QA benchmark set (100 questions over 15 different data sets) creation from different sources to create a rich question-answer pairs to consider for the experimentation

2. Systematic evaluation of the specific LLM setting responses over repeated prompting (100 iterations)

3. Listing the proposed evaluation metrics under three main components; (i) General Properties, (ii) Course-Grained Metrics and (iii) Task-Specific Metrics.

Primarily, different tasks are repeatedly experimented from the selection under DS-QA benchmark for our evaluation. The created computational worflow allowed to separate the text, code and image details for this semi-automated performance evaluation. Especially, the proposed three pillars for the evaluation metrics, includes the task-specific investigation to evaluate the performance of the model numerically.

Chapter 2

Literature Review

Generative AI—typified by large language models (LLMs) and other content-creating algorithms – is rapidly transforming data science workflows. In the last five years, advances in generative AI have enabled tools that can automatically clean data, generate analyses or code, build models, and even produce natural language summaries of findings. This literature review examines how generative AI is being applied to automate various data science tasks, the impact on traditional workflows (from data wrangling and exploratory analysis to modeling and decision-making), as well as the challenges regarding accuracy, interpretability, and ethical use.

The recent literature in the domain of AI-driven data analysis focused on more complicated and advanced data analysis tasks. DataSciBench Zhang et al. (2025), Text2Analysis He et al. (2023), Infiagent-dabench Hu et al. (2024), DA-Code Huang et al. (2023) and StatQA Zhu et al. (2024) are five main recent examples (not limited to these) discussed below for the testing performance of LLMs for data science-related tasks.

2.1 DataSciBench: An LLM Agent Benchmark for Data Science

Zhang et al. (2025) provides a benchmark to evaluate LLM capabilities in data science, DataSciBench. Compared to the previous benchmarks on single tasks and straightforward evaluation metrics, the DataSciBench benchmark focuses on a more comprehensive collection of natural and challenging prompts for uncertain ground truth and evaluation metrics. To generate ground truth (GT) and validate evaluation metrics, the authors also propose a semi-automated pipeline, which uses an LLM-based self-consistency and human verification strategy to produce GTs. Moreover, to assess each code execution outcome, the authors propose a Task – Function – Code (TFC) framework with precisely defined metrics and programmatic rules.

The overall framework of DataSciBench comprises of three key procedures, (i) prompt definition and collection, (ii) response integration and validation, and (iii) LLM evaluation. Firstly, the authors identified 6 data science tasks, including (i) Data Cleaning & Preprocessing, (ii) Data Exploration & Statistics Understand, (iii) Data Visualization, (iv) Predictive Modelling, (v) Productive Modelling, (vi) Data Mining & Pattern Recognition and (vii) Interpretability & Report Generation. The considered prompts were sourced from CodeGeeX, BigCodeBench, LLM-synthesis and human experts. To ensure high quality, the authors implemented question filtering and expert review on the collected prompts. Additionally, the paper introduced a TFC framework, which contains 25 aggregated functions and programmatic rules, to generate 519 test cases. With a dataset of 222 prompts with the proposed 519 GTs, authors examined

23 LLMs, including (i) 6 API-based models (mainly GPT-oriented ones), (ii) 8 open-source general models and (iii) 9 open-source Code Generation models

To generate GTs, the authors produced outputs of each prompt by sampling LLMs several times and execute the generated code to obtain the final output. For questions from Big-CodeBench, the answers were validated by performing all test cases and if all pass, then the answers were re-checked by humans. For other prompts, the authors applied a self-consistency strategy to obtain the output, which samples multiple reasoning paths for a question and use majority voting to pick the final result, and then the answers were verified by 6 authors of the paper. For evaluation selection, the authors used GPT-4o-mini to select valuable task types, return evaluation functions, and generate the evaluation code for each prompt. Additionally, two style of evaluation metrics they used to create a final score for each task. Regarding the main results, they demonstrate that API-based models outperform open-sourced models on all metrics and Deepseek-Coder-33B-Instruct has the best performance among open-source models. For further details on the procedure, evaluation and results, interested reader referred to the results and appendix section of the paper (Zhang et al., 2025).

2.2 Text2Analysis: A Benchmark of Table Question Answering with Advanced Data Analysis and Unclear Queries

He et al. (2023) proposed a Text2Analysis benchmark for tabular data analysis to conduct advanced analysis tasks that go beyond the SQL-compatible operations and require more in-depth analysis. This paper introduces 5 annotation methods to improve data quality as well as quantities and create unclear queries. The authors use in total 2249 input-output pairs with 347 tables and evaluate 5 models (GPT-4 models, StarChat-α, StarChat-β, CodeGen2.5, and TAPEX).

The authors argue that the existing tools such as Text2SQL and TableQA focus on elementary or rudimentary tasks, which are descriptive and insufficient diagnostic analysis to identify the patterns and insights within tabular data, and the operations can be executed with SQL. Text2SQL and TableQA can implement descriptive analysis, while TableQA is less effective to identify the causes of visible trends compared to Text2SQL. These tools pay insufficient attention to advanced analysis tasks that build on basic insights, integrating predictive and prescriptive techniques with reporting and visualization for deeper data analysis. To address this gap, the Text2Analysis benchmark [2] is developed to work on 4 data science tasks, focusing primarily on (i) Rudimentary Operations (whose main functions include group by, aggregation, filter, and sort), (ii) Basic Insights incorporates value-based ranking, group attribution, time-series trends, monotonicity, outlier detection, and unimodality, (iii) Forecasting and (iv) Chart Generation

The task distribution of all queries is 355 Rudimentary Operations, 689 Basic Insights, 273 Forecasting, and 1109 Chart Generation tasks. The task distribution of unclear queries is 56 Rudimentary Operations, 215 Basic Insights, 182 Forecasting, and 964 Chart Generation tasks. From the prompting perspective, this paper uses unclear queries, which may lack the essential information required to perform tasks. The authors state that unclear queries or intents are very common in real-life situations, and it can examine the analysis and recommendation capabilities of LLMs.

A Text2Analysis problem has the input (a table and a user query) and the output (Python code snippet(s) and the corresponding results). The Text2Analysis dataset contains 2249 pairs of (table, query, code, result) with 347 distinct tables. To collect the dataset, the authors established a tabular data annotation website on Label Studio and used the following annotation methods: forward annotation ((table, query) -> (code, result)), reverse generation from code snippets ((table, code, result) -> (query)), reverse generation from results ((table, result) -> (code, query)), expansion with new tables ((table) -> (query, code, result)), expansion of unclear queries ((clear query) -> (unclear query)).

In terms of the tools that they are testing, they focused on GPT family models (GPT-4 models (OpenAI 2023), with the setting of GPT-4-32k, a temperature of 0, and the maximum length of 4096) and different code-generation models such as StarChat-α/β and CodeGen2.5. For the evulation of the created results, they mainly focuse on the charateristics like Executable Code Ratio (ECR), Pass Rate and Regression Score. The authors performed iterative annotation and human evaluation to ensure the quality of the LLM's generation. To ensure the credibility of LLM's generation, they verified at least 100 samples. They compared the baseline performance of these tools over the created prompts, concluded that GPT-4 outperforms other models, while StarChat-β performs well in Chart Generation with regard to ECR. For further details on results, one can see the original paper in depth.

2.3 InfiAgent-DABench: Evaluating Agents on Data Analysis Tasks

Hu et al. (2024) introduced an InfiAgent-DABench benchmark designed for data science tasks, which requires agents to interact with an execution environment to end-to-end solve complex tasks. The benchmark InfiAgent-DABench contains the DAEval dataset, which is an evaluation dataset of 257 questions derived from 52 CSV files, and an agent framework with LLMs-based agents served as implementation and evaluation. The authors also use a format-prompting technique to convert questions into a closed-form format for automatic evaluation. This paper evaluates 34 LLMs and main experiment results show that current LLMs face many challenges in data science; and most powerful open-source LLMs achieve comparable performance with GPT-3.5. DAAgent is also developed to perform data science tasks, which surpasses GPT-3.5 by 3.9% on the InfiAgent-DABench benchmark.

The authors highlight the importance of closed-form questions and explain they help to bypass the issues caused by the unsatisfactory performance of LLMs on auto-evaluation and such as, GPT-4 models could only achieve 67% consistency with human experts. Human evaluation is applied to ensure the quality of the DAEval dataset and the evaluation metrics include suitableness, reasonableness, value, restrictiveness, alignment and correctness. Mainly, DAEval dataset's validation set is public and contains 257 questions with 52 csv files, 461 subquestions, and the maximum subquestions of 8 per question. The authors identified 7 data science tasks over these questions, including (i) Comprehensive Data Preprocessing (12.0%), (ii) Summary Statistics (24.0%), (iii) Feature Engineering (13.3%), (iv) Correlation Analysis (19.2%), (v) Machine Learning (5.1%), (vi) Distribution Analysis (17.1%) and (vii) Outlier Detection (9.3%).

For the classification, the authors also used GPT-4 to classify questions into easy, medium and hard levels and evaluated the different 34 benchmarks; (i) Proprietary models such as OpenAI GPT-4 (OpenAI, 2023b), (ii) Open-source general LLMs such as Vicuna, (iii) Open-source code LLMs such as Code Llama and (iv) Agent frameworks such as XAgent. The main experiment results summarized by the authors include data analysis tasks are challenging for current LLMs; most powerful open-source LLMs achieve comparable performance with GPT-3.5; and DAAgent surpasses GPT-3.5 by 3.9%. They presented that GPT-4 exhibits excellent performance in Summary Statistics, Correlation Analysis, and Outlier Detection, and satisfactory performance in Data Preprocessing, Feature Engineering, and Machine Learning.

2.4 DA-Code: Agent Data Science Code Generation Benchmark for Large Language Models

The DA-Code benchmark is designed by Huang et al. (2023) to assess LLMs on challenging, real, and diverse data science tasks. The authors study 9 LLMs and main experiment results include GPT-4 only achieves a 30.5% score and as the difficulty increases from Easy to High, the score sharply cuts; Reference Plan can help to improve model performance and for instance, it leads to an increase of 8.2% in the overall score of DA-Code on DA-Code-100. The paper further explores that agents are expected to grasp the task instruction and solve the task in the early steps, otherwise additional steps do not necessarily improve the performance; and most developed LLMs follow an Exploration-Execution-Evaluation-Adjustment pattern.

The overall annotation pipeline of DA-Code consists of 5 key procedures, Manually Selecting Data Source, Rewrite Task or Define New Task, Task Implementation, Evaluation Setup, and Cross Validation and Redteam Test. The authors extracted from Kaggle, Github, and Other Web Sources, to construct a dataset with real-world relevance, complexity, timelines, and code intensity. For task definition, the paper mainly rewrote the tasks by transforming explicit code instructions into abstract agent task descriptions. For implementation, the tasks were set up in a Sandbox environment and agents were required to extract useful information from a noisy environment of multiple files to complete tasks.

The DA-Code benchmark contains 500 instances and requires about 6 files per task over 6 data science tasks, including (i) Data Wrangling, (ii) Machine Learning, (iii) Data Manipulation, (iv) Data Insights, (v) Visualization and (vi) Statistical Analysis. In DA-Code, Exploratory Data Analysis (covering the last 4 data science tasks) account for up to 60%. Each of Data Wrangling and Machine Learning has 100 questions. The medium-level tasks account for 57%. 69% of the files are in the form of .csv and other file types are accepted, including .md, .yaml, .sql, .txt, .sqlite, and .xlsx. This paper examines 9 LLMs, including (i) Open-source general LLMs such as Mixtral-8x22B (Jiang et al., 2024), (ii) Open-source code LLMs such as DeepseekCoder-V2.5 (Zhu et al., 2024)., (iii) Closed-source or Proprietary models such as Claude-3-Opus (Anthropic, 2024) and GPT-families and (iv) Agent frameworks such as OpenDevin (OpenDevin Team, 2024). Their main evaluation metrics include items called Total Match Score, Chart Match Score and ML Normalized Score.

In terms of their main findings, the most advanced model GPT-4 only achieves a 30.5% score. It exhibits better performance in ML than DW and EDA and as the difficulty increases from Easy to Medium, the score almost cuts by half. GPT-4, GPT-4o and Claude-3-Opus models have high completion rates and satisfactory ratios of executable code. The three models perform obviously better than the other models, but there is a common trend that the score declines when the difficulty increases. Qwen2.5-72B is the only model which has a better performance in hard tasks compared to medium-level tasks. Deepseek-Coder-33B has the lowest score of 10.8%.

This paper also explores task completion efficiency, Exploration-Execution-Evaluation-Adjustment pattern (EEEA), and error analysis. The success rates substantially grow within the initial 4 and 10 steps before reaching a plateau. After 10 steps, even with the continuous reduction of incomplete rates, success rates do not keep increasing at the same speed. This highlights that agents are expected to grasp the task instruction and solve the task in the early steps, otherwise additional steps after 10 steps do not necessarily lead to better results.

2.5 StatQA: Are Large Language Models Good Statisticians?

Zhu et al. (2024) introduced StatQA, a new benchmark designed for statistical analysis tasks. StatQA comprises 11,623 examples tailored to evaluate LLMs' proficiency in specialized statistical tasks and their applicability assessment capabilities, particularly for hypothesis testing methods. They systematically experiment with representative LLMs using various prompting strategies and show that even state-of-the-art models such as GPT-4o achieve a best performance of only 64.83%, indicating significant room for improvement. Notably, while open-source LLMs (e.g. LLaMA-3) show limited capability, those fine-tuned ones exhibit marked improvements, outperforming all in-context learning-based methods (e.g. GPT-4o). Moreover, their comparative human experiments highlight a striking contrast in error types between LLMs and humans: LLMs primarily make applicability errors, whereas humans mostly make statistical task confusion errors.

For the purpose of systematic evaluation, authors proposed a set of statistics related questions called as e StatQA, a new benchmark for statistical analysis tasks, particularly focusing on the applicability assessment of statistical methods (Zhu et al., 2024). They created refined versions of questions via LLM across different difficulty level and topics that requires the certain data set for the execution of the task. They mainly divided the tasks into descriptive statistics and inferential statistics, with inferential statistics primarily including regression analysis and hypothesis testing. Overall, five typical categories of statistical tasks are created (StatQA contains 11,623 examples and mini-StatQA contains 1,163 examples), ranging over (i) Correlation Analysis, (ii) Contingency Table Test, (iii) Distribution Compliance Test, (iv) Variance Test and (v) Descriptive Statistics (please see Table 3 of Zhu et al., 2024 for further details for each).

This research, relying on the created StatQA, examines the performance of certain LLMs, including (i) Non-fine-tuned LLMs such as LLaMA-3 models or ChatGPT (gpt-3.5-turbo), GPT-4, and (ii) Fine-tuned LLMs such as fine-tuned versions of the models: LLaMA-2 (LLaMA-2-7b-chat-hf) and LLaMA-3 (Meta-Llama-3-8B, Meta-Llama-3-8B-Instruct). Additionally, as alternative strategy, few-shot learning and Chain-of-Thought (CoT) are used as prompting strategies within these models.

They found that different LLM models display significant performance discrepancies on our benchmark, indicating sufficient discriminative ability. The LLaMA-2 models show weak performance, suggesting their statistics inability. The newer LLaMA-3 shows remarkable improvement over the previous LLaMA-2, with the LLaMA-3-8b achieving an accuracy of up to 36.1%, far surpassing all tested LLaMA-2 models, close to the performance of 0-shot GPT-3.5-Turbo (37.4%) whereas it is still noticeably weaker than GPT families like GPT-4 and GPT-4o. On the other side, by fine-tuning, LLaMA-2-7b, LLaMA-3-8b, and LLaMA-3-8b Instruct models achieve marked enhancement, with the fine-tuned LLaMA-3-8b model showing the best overall performance, but considerable room still remains for further improvement. As a part of their study, they found that (i) few-shot learning and the inclusion of domain knowledge are helpful for LLMs in this task, whereas CoT is more likely to result in slight performance degradation in smaller models, (ii) LLMs with prompt-based approaches remain behind people in statistics and (iii) humans and most LLMs are adept at descriptive statistics tasks but struggle with contingency tables and variance tests.

2.5.1 Overall summary

Some of the project-goal related papers in the literature (beyond the underlined papers above) can be briefly summarized as follows in Table 2.1

Table 2.1: Brief overview of Code-Generation Benchmark-style works

Benchmark	**Prompt Source**	**Evaluation Metrics**
Analyst-Inspector approach (Zeng et al., 2025)	DiscoveryBench & QRData & StatQA	Reproducibility-oriented
DataSciBench (Zhang et al., 2025)	CodeGeeX & BCB & Human	Aggregate Metrics and Programmatic Rules
NaturalCodeBench (Zhang et al., 2024)	CodeGeeX	Test Cases + Pass Rate
LiveCodeBench (Jain et al., 2024)	LeetCode & AtCoder & CodeForces	Test Cases + Pass Rate
StatQA (Zhu et al., 2024)	LLM & Human	LLM & Human
BigCodeBench (Zhuo et al., 2024)	StackOverflow	Test Cases + Pass Rate
InfiAgent-DABench (Hu et al., 2024)	LLM	Accuracy
DS-1000 (Lai et al., 2022)	StackOverflow	Test Cases + Surface-Form Constraints
MLAgentBench (Huang et al., 2023)	Kaggle	Acc. + Success Rate, Human
Text2Analysis (He et al., 2023)	Human & LLM	Executable Code Ratio, Acc., Regression Scores

For further details of the each work, the reference list mentioned at the end can be useful to explore.

2.6 Novelty of our project

Our study primarily focuses on the impact of the repeated prompting over data science / statistics tasks, created via questions written by human experts. By mimicking the idea of Monte Carlo simulation, we have investigated the deviations across the same prompting for a specific LLM tool. In addition to the novel repeated prompting idea, this work results in a curated question-answer pair style of data sets covering different topics of data science / statistics at varying difficulty levels. For that purpose, a set of DS-QA (100 questions over 15 different data sets) is curated in a certain format.

For the experimental setting, since GPT models are primarily the ones that are performing better than other competitors and since we are interested in the impact of code-interpreter tool, we narrowed our focus into the non-fine-tuned GPT versions attached to code-interpreter external tool. Differently from the available listed works, as it was highlighted in the work of Zeng et al. (2025), we targeted to create specific evaluation metrics for different types of questions beyond the certain performance related metrics, such as completion ratio or similarity of the created response. Additionally, rather than focusing on one single metric or score for the overall response quality to measure, we breakdown the response nature into three main metric categories; (i) General Properties, (ii) Course-Grained and (iii) Task-Specific Metrics.

For the systematic evaluation over repeated promptings, each prompt for the selected questions is initiated 100 times consistently via API protocol. In addition to the repeated prompting impact, the possible impact of the change of base-model, such as changing from GPT-4o to GPT-4.1 (under the recent available model list), and the model temperature parameter discussed briefly alongside the main project findings. Besides, the certain programming related constraints within the questions are simply highlighted whenever the nature of prompt includes specific instruction of R language related constraints.

Chapter 3

Methodology

We mentioned in Chapter 1 that a project experimentation follows a certain modeling settings. For the collection of results, various connected python scripts are created that are working in a modular style. For the numerical exploration, specific-formatted input data (question-answer) pairs are curated.

3.1 Format of Question-Answer pairs

For each of the created questions in our DS-QA, we applied the same structure for the input prompt including certain question characteristics as they mentioned earlier such as concept name or difficulty level. All these questions are created on a jsonl file format to use them repeatedly and easily over various settings, as exemplified below;

```
{"id": 0,
"question": "How many tourist attractions are there in the data set?",
"concepts": ["Data understanding"],
"format": "@count_visitor_value[count_visitor_value]
where \"count_visitor_value\" is given in integer format.",
"file_name": "UK-visitor-numbers.csv",  "level": "easy"}
```

In a similar manner, especially for the closed-form type of questions that we curated, the response list is created to match with the given id number easily, exemplified below.

```
{"id": 0,
"question": "How many tourist attractions are there in the data set?",
"concepts": ["Data understanding"],
"format": "@count_visitor_value[count_visitor_value] where \"count_visitor_value\"
is given in integer format.",
"file_name": "UK-visitor-numbers.csv", "level": "easy"
"common_answers": [["count_visitor_value", "348"]]}
```

With this matching, we aimed to implement specific automated evaluations for certain characteristics, practically over the generated multiple responses. For sure, when the questions include multiple steps and not only a unique numerical response, the structure of the "common_answers" is getting more complex in principle. It may include more than one unique numerical value, or both numerical value and its interpretation, the former illustrated as below;

```
{"id": 5, "question": "What are the mean and median visitor numbers
in 2022 across all attractions?",
"concepts": ["Data summary"],
"format": "@visitor_summary[visitor_summary] where \"visitor_summary\" is given
as a numeric vector or tibble formated values.",
"file_name": "UK-visitor-numbers.csv", "level": "medium",
 "common_answers": [
   {
     "visitor_summary": {
       "mean": 351942,
       "median": 184640
     }
   }
 ]
}
```


3.2 LLM interaction framework

The main communication with the selected LLM setting is implemented via API key usage for the multiple times. The main goal is to examine the performance of the DA (code interpreter) plugin with certain GPT-4 versions to see their impact. Additionally, over a set of selected questions, certain scenarios are created.

In our original system prompt instructions, we avoided well-defined details for the testing purposes of almost zero-shot learning performance of the created settings. One specific user prompt component that we injected is about saving the necessary Python code components with the help of adding the specific phrase "Provide a complete Python snippet ready to run".

Table 3.1: Experimental setting summary

Model setting	Parameter	Constraints
"gpt-4o" + DA		
"gpt-4o-mini" + DA		
"gpt-4o-nano" + DA		
"gpt-4o-mini"		
	Temperature	(0.2, 0.7, 1.0, 1.2)

3.2.1 Experimental setting

For the impact of the selected model, by using the selected questions, different GPT model versions are tested under the repeated prompting mechanism as summarized in Table 3.1. Beyond this testing, by fixing the prompt and the model, we use different temperature values from the set of options (0.2, 0.7, 1.0-default, 1.2) to examine the importance of LLM parameters that are mostly invisible for the end users.

3.2.2 Coding Design

The code design was intended to be relatively modular so that new data science questions can be trained or implemented by the created LLM settings easily. We provide a brief overview of each of the folders/files, presented in the project Github resource;

- `params`: Contains the functions
  - `load_params`: Reads questions from jsonl files and keep in SimpeNamespace.
  - `input_dataset`: Split into different datasets, specific to ids, questions, concepts and difficulty levels.
  - `data_name_mapping`: Maps the usually used dataset name to the corresponding filename
  - `fileid_name_mapping`: Maps the usually used dataset name to the corresponding file id
- `scripts`: Contains the scripts to run all simulation experiments, collect results and make evaluation.
  - `automation`: Covers two functions of multi-round experiments, one for running each of a selected set of questions specific to one dataset, the other for running selected sequential questions specific to one dataset.
  - `collect_results`: Extracts text and Python code parts from the experiment outcomes, and stores them separately. Combines the following details into one jsonl file for a dataset: each question id, round, thread id, status (completed or failed), runtime, number of words for text, number of tokens for text, reasoning part, and outcome. For sequential questions, one additional step is to separate the outcomes per question. The features include {'id', 'round', 'thread_id','words','tokens','status','runtime','reasoning'}
  - `evaluation`: Compute the accuracy of outcomes compared with GTs.
    * Number of words or tokens for the question input.
    * Number of tokens corresponding to GPT-4o, GPT-4.1-mini, GPT-4.1-nano. As the token embedding for "gpt-4.1-nano" and "gpt-4.1-mini" doesn't exist by Juliy 2025, we use an encoding of 'cl100k_base' instead.

    Then further computes the following metrics:
    * Verbosity Ratio (Input/Output) regarding words and tokens.
    * Jaccard Index as a similarity measure of the reasoning text content, with the generated responses at the initial round as the comparison benchmark.
    * Completion Ratio, which equals to 1 if status is 'completed' and 0 otherwise.
    * Accuracy of outcomes compared with GTs for questions with direct answers. Accuracy for questions with equation outcomes is checked with regression equations. Accuracy for the other mainly text-based questions leaves for human evaluation.
    * Image-related metrics: require_image = 1 if a question asks for visualization images and/or the generated responses contain images.
    * Code Executability: code_execute = 0 for code missing, 1 for code with issues, 2 for successfully code executability.

To run multi-round experiments on a selected set of questions specific to one dataset, we provide a single file (`runner.py`) that can accommodate both independent questions and sequential questions. Within this modular framework, if all the questions are independent from each other, we can use the function `multi_round_assistant`. If the questions are sequentially related, the user can execute the alternative `sequential_question_assistant`. Overall, all the coding-related details are described in the project Github repository, given in https://github.com/oevkaya/GAIL-DS-project.

3.3 Metrics for Response Evaluation

Differently from the other works, we aimed to classify certain performance related results under three main themes. Specifically, as one of the recent gaps, the novel component of the targeted list includes task related metric items to evaluate the quality of the response beyond the completion rate or just checking the match with ground truth.

The main metrics for the created responses can be listed under three subsections mainly.

3.3.1 General Properties

- **Verbosity**: General length for the number of tokens or words to count the verbosity of the generated response in general except the coding component. This will focus on only the text related part!
- **Run time**: For the response creation for each iteration, the total runtime of the response creation is calculated.
- **Verbosity ratio (input / output)**: It is the ratio of number of input tokens (words) / number of output tokens (words).

3.3.2 Course-Grained Metrics

This section include the main metrics that are applied for all types of tasks in general. One of the given items below (Response Accuracy (RA)) will be combined later with the task specific cases to cover the nuances in generated responses accross multiple promptings.

- **Completion Ratio (CR)**: This can be two different values in general (i) 0: if the executed thread is failed/not completed, (ii) 1: if the executed thread completed with an acceptable outcome (either matching with ground truth or not). This can be measured for each prompt out of 100 trials based on the stored list of responses as a ratio.
- **Response Accuracy (RA) or Accuracy of Response (AoR)**: Whether the reported result is matching with the ground truth or not (can be numeric or string, or vector etc.), it can take either 0 or 1 again. If the output is decimal value, or if the match appears partially it can be controlled during the comparison for numerical values!
- **Text Similarity**: Similarity measures such as Jaccard Index for the text part of the generated response to compare with the first response as reference or ground truth for some tasks, when it is applicable.
- **Code Executability**: Whether the generated code is directly executable in a different environment or not, either taking 2 (it is executable and solving the problem successfully), taking 1 (it is available but not executable directly) or 0 (not saved executable code).

3.3.3 Task-Specific Metrics

For each of the concepts we are considering, varying set of criterions can appear for the output evaluation in general. These task-related ones are following the common grounds of each solution while evaluating their performance. Equivalently, the task-specific metric items for specific questions are created not only focusing on the single numerical answer.

By mimicking the idea of rubric creation of an exam question, we believed that certain tasks need specific item checking for the numerical evaluation beyond the accuracy. For the sake of simplicity, out of 5, each response can be evaluated based on the created metrics individually and repeatedly using binary marking. With this strategy in mind, we focused on certain tasks and related evaluation metrics. Although there are a variety of questions we had, in a thematic way we primarily focused on the following tasks and its corresponding task-specific metrics;

Data Summary stats and interpretations

- **Ratio of Accuracy** (will be calculated automatically via comparing GT value)
- **Data Context recognition:** Clearly define each variable, its units of measurement, and data source to ground the summary in context
- **Appropriateness of Stats Chosen:** Were the most relevant statistics reported for the data type within the context of question (e.g., median for skewed data, mode for categorical)?
- **Accuracy of Interpretation:** Do the interpretations logically follow from the statistics? (e.g., not saying data is "normally distributed" based on wrong evidence)
- **Clarity of Interpretation:** Are the interpretations understandable, concise, and easy to follow, or is there any confusing word selection or terminology usage.

Data Visualization and EDA

- **Data Viz Completeness:** If the requested data visualization is generated or not, it can take 1 (if they exists) or 0 (if not) otherwise. If this takes 0, nothing to compute indeed so this item is just the starting point so not included in the overall
- **Aesthetic Mapping:** Each data variable is unambiguously mapped to an appropriate aesthetic (e.g., a continuous variable to position, a categorical variable to color), and the mappings are clearly documented in a legend or caption.
- **Geometric Object (Geom Layer):** The chosen geom (e.g., geom_bar() for counts, geom_point() for scatter) matches the data structure and analytical intent given by the question, with no misuse or unsuitablee geom selection
- **Scales & Coordinate System (Scale/Coord Layers):** Axes use appropriate, non-truncated scales with meaningful breaks; any coordinate transform (e.g., log scale, coord_flip()) is applied suitably, and all components are fully readable.
- **Visualization-Interpretation matching:** The correct use of wordings and abbreviations in the created visual and the related interpretations.

Linear Regression

- **Ratio of Accuracy** (will be calculated automatically via comparing GT value)
- **Correct model coefficient interpretation:** Understanding the magnitude and direction of the relationship between independent and dependent variables

- **p-value recognition:** Recognizing statistical significance on the model coefficients by reporting of p-values for each coefficient against a pre-specified -level (commonly 0.05).
- **Evaluation Metric Reported and Interpreted:** Assessing the model's goodness of fit and the proportion of variance explained by the model via R^2, and/or supplemented by predictive error metrics such as RMSE or others
- **Interpretation of Results:** Interpreting the results within the context of the specific research question and the problem domain.

Logistic Regression

- **Ratio of Accuracy** (will be calculated automatically via comparing GT value)
- **Correct model coefficient interpretation:** Understanding the magnitude and direction of the relationship between independent and dependent variables
- **p-value recognition:** Recognizing statistical significance on the model coefficients by reporting of p-values for each coefficient against a pre-specified -level (commonly 0.05).
- **Confusion Matrix recognition:** The confusion matrix details and derive class-specific error rates
- **Interpretation of Results:** Interpreting the results within the context of the specific research question and the problem domain.

Principal Component Analysis (PCA)

- **Suitable Variable Selection:** Identifying the suitable type of variables in the data by selecting.
- **Number of Components Justified:** Is the decision on how many PCs to keep justified correctly?
- **Standardization of Input Data:** Was scaling recognised and applied where needed before PCA?
- **Correct Variance/Cum. Variance Explanation:** Are explained variance / cumulative variance ratios reported and understood?
- **Visual evidences:** Is the decision on the number of PCs supported by certain visuals like loading, biplot or scree plot.

Clustering

- **Number of clusters justified:** Is the final number of obtained clusters justified correctly?
- **The methods consistency:** Is the requested method applied properly with the specification of certain parameters
- **Cluster characteristics:** Are the obtained cluster characteristics such as cluster size, cluster profile mentioned clearly

- **Validation metrics:** Are the certain report metrics like silhouette scores, Davies-Bouldin index, or other relevant statistics are recognised
- **Visual evidences:** Is the decision on the number of clusters supported by certain visuals like scatter plots, dendrograms.

Hypotesis testing

- **Correct Test/Test name Chosen:** Recognition of the correct statistical testing / test name
- **Null and Alternative Hypotheses Clearly Stated:** Clearly and correctly stated hypothesis test components with correct notation
- **Accuracy on test statistics/p-value and/or Critical region** Having the correct final decision based on the testing procedure
- **Correct Interpretation of test statistics/p-value and/or Critical region** Having the correct final decision based on the testing procedure
- **Contextual Conclusion:** Interpreting the final decision within the context of the specific research question

3.4 Main Workflow

Figure 3.1 is a rought summary of the main workflow that we followed, after sending the specific task related prompts to the model;

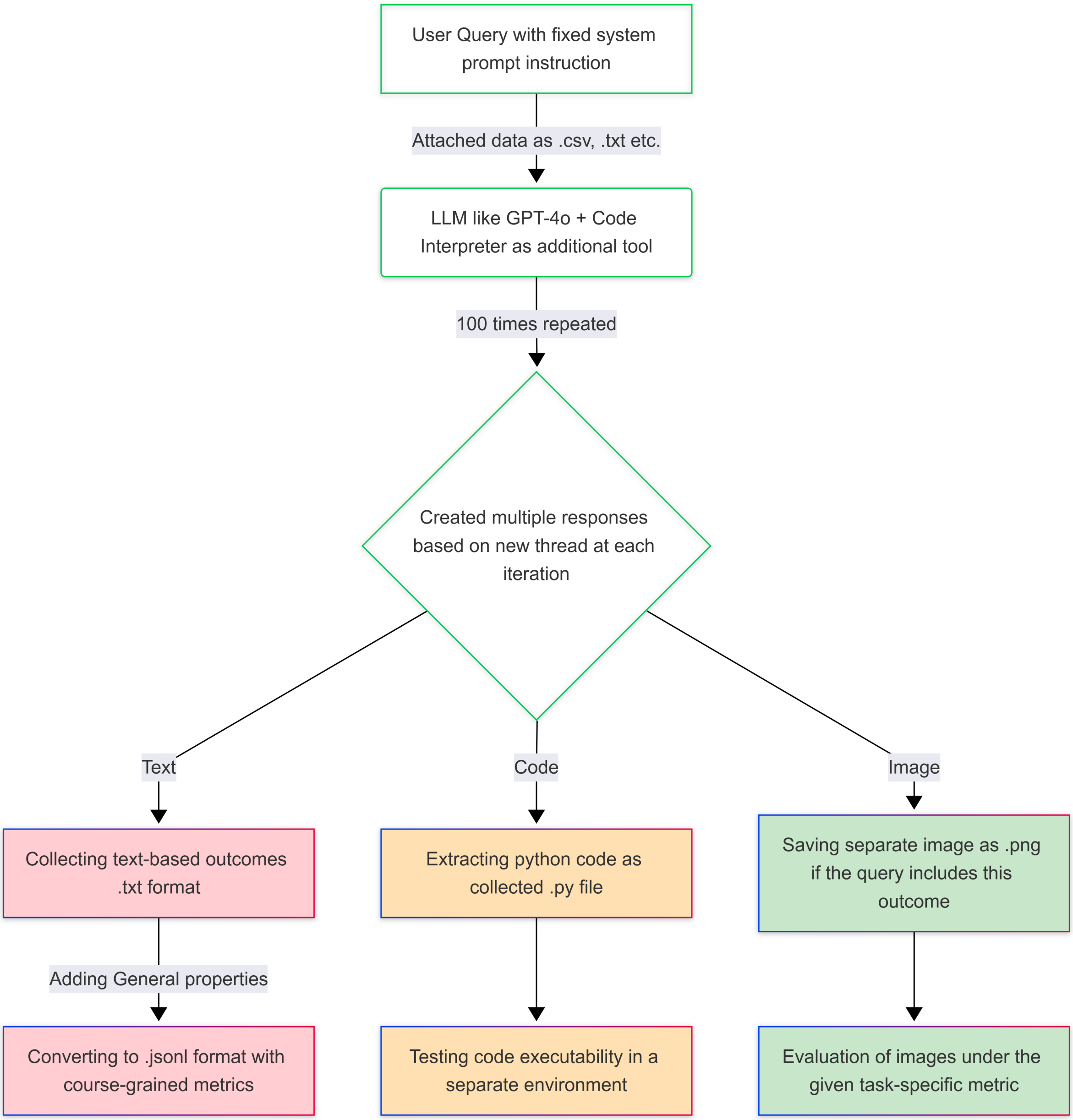


Figure 3.1: Main workflow for the repeated execuation via API usage

Chapter 4

Results

The results chapter summarizes the main findings of the research project after giving the main highlights below;

- The API-based LLM interaction allowed systematic prompt sending and response collection, whereas there are some inconsistencies on the response format such as missing code component.
- Data reading and parsing can be a consistent issue that we observed over different data sets, whenever it appears at the beginning, it may result in incomplete response or very long response including the solution of data reading issue in an unanticipated way.
- The automated calculation of response characteristics (general properties) and the course-grained metrics such as similarity index is easy to complete, but the code executability tested separately. Some results on the specific data sets and questions on the executability of the generated code is given in Section 4.2
- Across different tasks and different concepts, the overall average performance (out of 5) based on our metric creation and the individual metric items can be varying, as it is summarized in Section 4.3 with different tasks.
- Adding the code-interpreter as an additional tool or not makes a real difference as we observed in the Table 4.8 in Section 4.4 since it may not create the numerical results in the zero-shot style prompting.
- The impact of the temperate is crucial also, similar to the selection of the base-model, some observations are highlighted over two different tasks over the evals data is summarized in Section 4.5 verbally.

Some initial examples are given on the topic called "Data Understanding" but later certain tasks are primarily focused on using a subset of questions from the original list. For the response extraction, all the repeatedly asked prompt results (via "round" number) are collected for each question (via "id" number) covering reasoning, outcome, and general properties with some course-grained metric results, calculated automatically.

In general, we considered the combination of model="gpt-4o" and code interpreter for the whole set of questions that we curated. For some questions, different models are discussed as presented, but limited to GPT-4.1 mini scenario with and wout code interpreter.

4.1 Output format with metrics

To illustrate, the following representation covers the first round of the generated responses in a structured way for the question with "$id = 0$" relying on the `UK-visitors.csv` data set file;


```
{"id": 0, "round": 0, "thread_id": "thread_kzmVH4BH7hiop26SSebamTiH",
"image_id": null, "words": 288, "tokens": 411, "reasoning": "First, let's examine
the contents of the uploaded file to determine the structure and data it contains.
After that, we'll identify
how many tourist attractions are listed in the dataset.
Let's start by loading and inspecting the file.
It seems there's an issue with reading the file.
This could be due to several reasons, including an incorrect file format.
Let's check to ensure the file is indeed an
Excel file and determine the correct way to read it.
I'll first attempt to identify the file type and
then try reading it again. It seems that the file
type can't be determined using mimetypes.
To proceed, I'll attempt to read it as a CSV file, as it's a common format
that can sometimes be mislabeled. If this fails,
I'll check for any specific patterns in the file content. Let's give it a try.
The file was successfully read as a CSV.
Here is a preview of the dataset:-
**attraction**: Name of the tourist attraction-
**n_2021**: Number of visitors in 2021-
**n_2022**: Number of visitors in 2022-
**admission**: Type of admission-
**setting**: The setting of the attraction-
**region**: Region where the attraction is located
To determine the number of tourist attractions, we simply need to count
the unique entries in the \"attraction\" column.
Let's count them. There are 348 unique tourist
attractions in the dataset. Here is the Python
code used to determine this:
If you have any more questions or need further analysis, feel free to ask!",
"status": "completed", "runtime": 38.14174938201904, "input_words": 55,
"input_tokens_gpt_4o": 11,
"input_tokens_gpt_4.1_mini": 11, "input_tokens_gpt_4.1_nano": 11,
"common_answers": [["count_visitor_value", "348"]],
"file_name": "UK-visitor-numbers.csv",
"concepts": ["Data Understanding"],
"level": "easy",
"verbosity_ratio_words": 0.1909722222222222,
"verbosity_ratio_tokens": 0.0267639902676399, "complete_ratio": 1,
"accuracy": 1, "auto_image": 0, "require_image": 0,
"Similarity_Jaccard": 1.0,
"code_execute": 2, "code_error": ""}
```


As one can see, "Role:" details are stored as the reasoning component to be evaluated across various tasks, whereas the corresponding code component is stored separately for code executability testing, exemplified below. Like mentioned before, "id" and "round" values are representing each question with multiple responses in a similar format. Each unique value given by "thread_id" represent the initiated thread for the given task based on the created LLM session separately.

```
#Question 0, Round 0 with threat_id: thread_kzmVH4BH7hiop26SSebamTiH
import pandas as pd

# Load the CSV file
df = pd.read_csv('/mnt/data/file-DhUsT7Dn6RBmVs7HT6yCZa')

# Count the number of unique tourist attractions
number_of_attractions = df['attraction'].nunique()
print(number_of_attractions)
```

For the general properties, by using the text stored within the "reasoning" item, both the words and tokens of the output and the verbosity ratio values are calculated as different items. Additionally, as the numerical part of course-grained metrics, the values of "CR", "Similarity_Jaccard" and "out" components are created to represent the completion ratio (where 1 means the executed thread completed with an acceptable outcome), the Jaccard similarity index (where 1 means it is solely compared by itself because of being the first response) and numerical response accuracy (where 1 means the obtained answer matches with the original Ground truth- GT). For the illustration, the numerical response to this question is calculated as GT $= 348$, where the created response includes the correct calculation both within the "reasoning" item and covers the matched answer within the "outcome": "are 348 unique" item.

To see how the general properties and numerical course-grained metrics are changing across different tasks and difficulty levels, we explored the calculated values across different "id" values to create their summary statistics. Over 100 rounds that we obtained from different LLM session responses, it is possible to see how these characteristics are changing. For instance, again for "id=0" question, we can see these changes over 100 repeated prompts as in Figure 4.1

As in Figure 4.1, one can see the distributional properties of the general characteristics and text similarity information visually.

Across different style of questions, with varying topic or difficulty level, we can see the changes of certain characteristics across the created question-answer pairs. To illustrate, for the `UK-visitors.csv` data set again, we can see how certain characteristics (general properties) are changing over different questions ($id = 0, \cdots 9$) as in Figures 4.2 and 4.3.

In these images, one can see the differences between the calculated response verbosity or the corresponding versbosity ratio as the characteristics of the outputs created by an LLM. For certain tasks, runtime might have some extreme values but overall the variability on the runtime for all questions of the data set of UK-visitors.csv are almost the same (upper panel of Figure 4.2). For the output tokens, variability seems higher across different types of questions and depend on the simplicity of the input in general. For question $id = 6$, we had the largest value (lower panel of Figure 4.2).

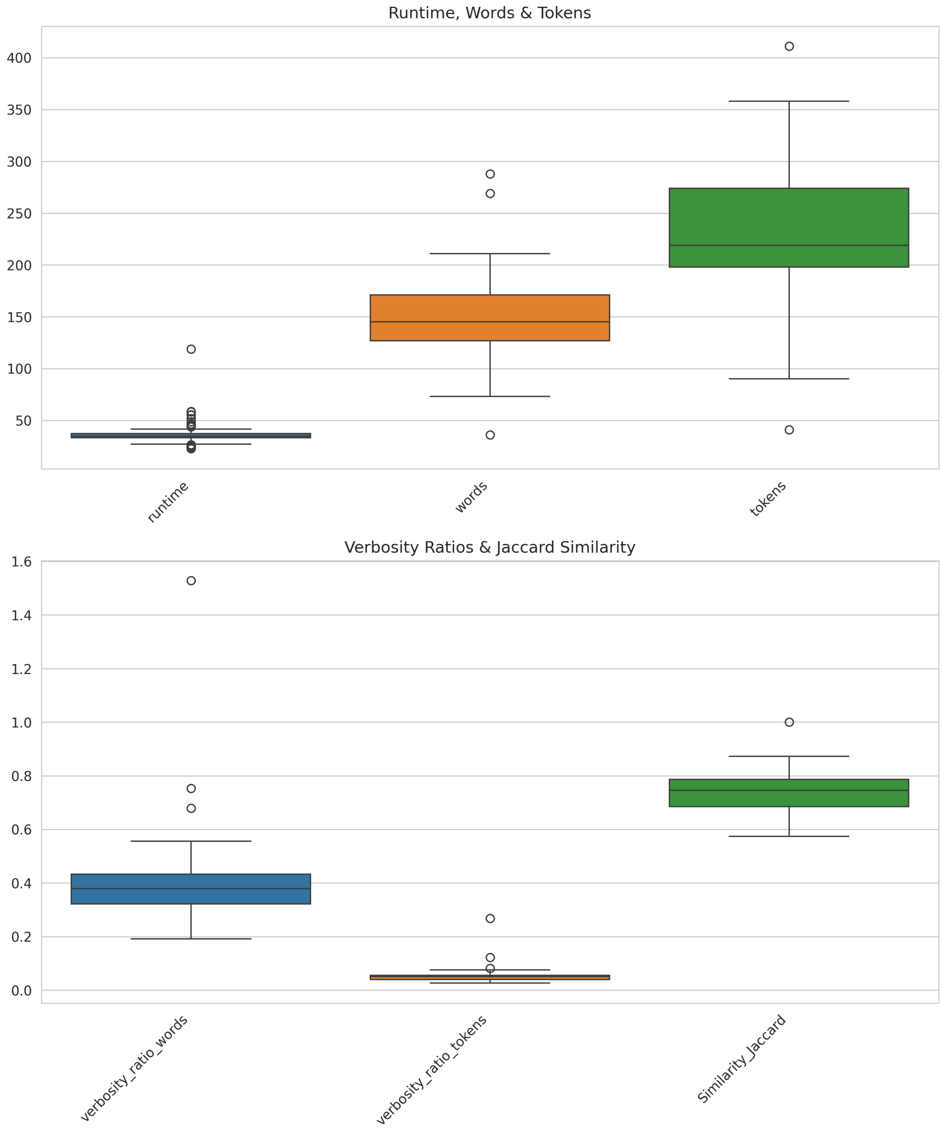


Figure 4.1: Summary of response characteristics for "id=0"

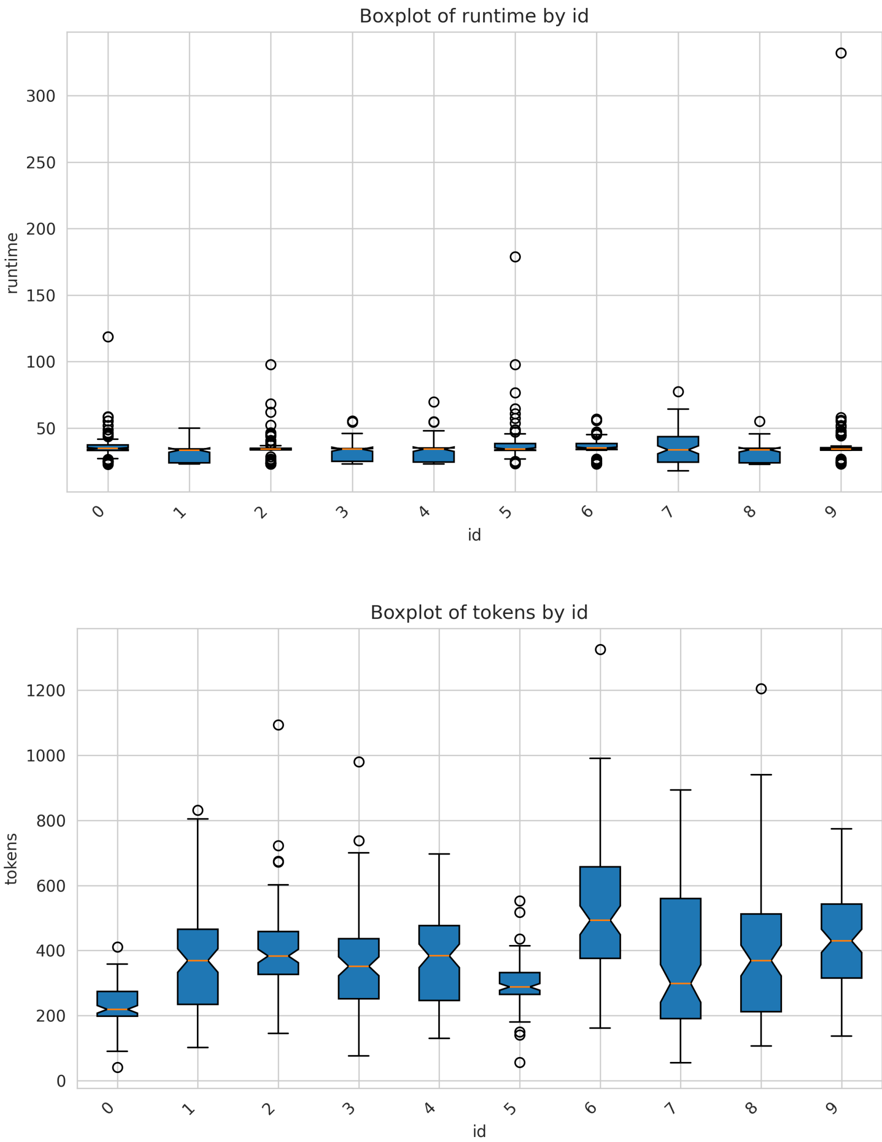
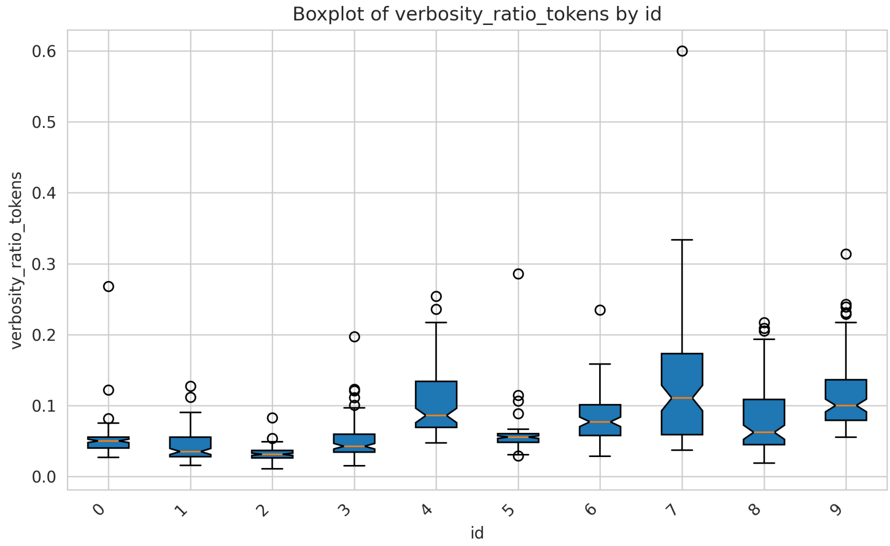


Figure 4.2: Runtime and output tokens summary of response characteristics for different question IDs.



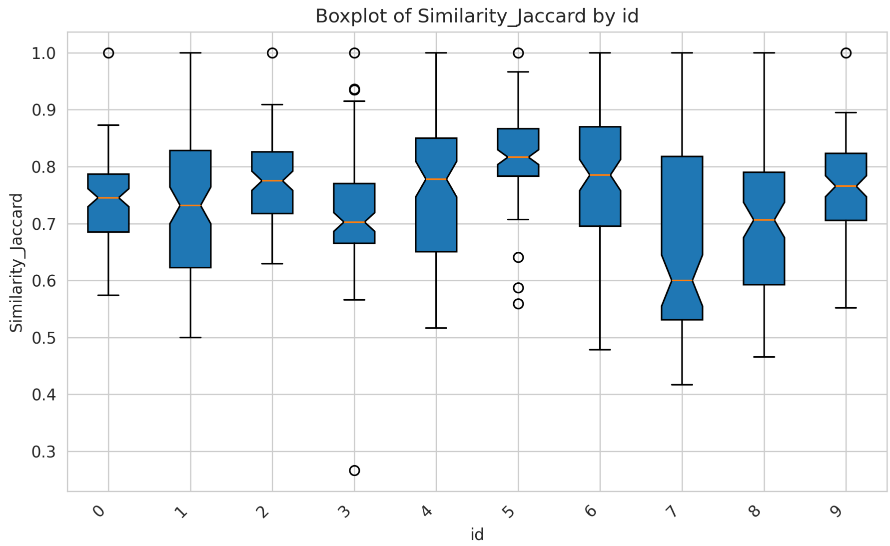


Figure 4.3: Verbosity ratio for tokens and Jaccard Similarity summary of response characteristics for different question IDs.

4.2 Code Executability

For this metric component, as the part of course-grained metric, it is possible to extract the created python code for the task and test about their executability in a different session. For that purpose, triplet system is used over the extracted code components, available for some set of questions under code_record folder, similar to the exemplified illustration earlier.

As a small illustration, over set of questions, we counted the specific cases out of 100 trials to point out the possible deviation from the correct and executable code snippet related to task computation.

Model	Temperature	Q ID	Item	Score
gpt-4o-mini	1.0	21	2	55
		21	1	12
		21	0	34
gpt-4o-mini	1.0	23.0	2	78
		23.0	1	17
		23.0	0	5
		23.1	2	53
		23.1	1	18
		23.1	0	29
		23.2	2	37
		23.2	1	32
		23.2	0	31
gpt-4o-mini	1.0	26	2	42
		26	1	46
		26	0	62
		26.1	2	10
		26.1	1	17
		26.1	0	73
gpt-4o-mini wot Code Interpreter	1.0	26	2	85
		26	1	10
		26	0	5
		26.1	2	86
		26.1	1	8
		26.1	0	6

Table 4.1: Summary of evaluation results for evals data set for a specific question with / without code interpreter

As shown in Table 4.1, mostly the created code snippets are executable, whereas less cases observed as "code with issues" - 18% and "code missing" - 11% in total for the sequantial question that we explored.

Model	Temperature	Q ID	Item	Score
gpt-4o-mini	1.0	0	2	95
		0	0	6
		1	2	69
		1	1	31
		2	2	76
		2	1	22
		2	0	2
		3	2	63
		3	1	22
		3	0	15
		4	2	65
		4	1	26
		4	0	9
		5	2	91
		5	1	3
		5	0	7
		6	2	60
		6	1	22
		6	0	18
		7	2	37
		7	1	32
		7	0	31
		8	2	32
		8	1	38
		8	0	30
		9	2	57
		9	1	25
		9	0	18

Table 4.2: Summary of evaluation results for UK-visitor-number data set for a specific question without code interpreter

Model	Temperature	Q ID	Item	Score
gpt-4o-mini	1.0	56	2	5
		56	1	41
		56	0	54
		57	2	8
		57	1	41
		57	0	51
		58	2	13
		58	1	29
		58	0	58
gpt-4o-mini wot Code Interpreter	1.0	56	2	89
		56	1	11

Table 4.3: Summary of evaluation results for aeroplane data set for a specific question without code interpreter

Model	Temperature	Q ID	Item	Score
gpt-4o-mini	1.0	14	2	71
		14	1	2
		14	0	7
gpt-4o-mini	1.0	14.0	2	71
		14.0	1	22
		14.0	0	7
		14.1	2	39
		14.1	1	22
		14.1	0	39
		14.2	2	47
		14.2	1	26
		14.2	0	27

Table 4.4: Summary of evaluation results for instructional-staff data set for a specific question without code interpreter

Model	Temperature	Q ID	Item	Score
gpt-4o-mini wot Code Interpreter	1.0	73	2	38
		73	1	54
		73	0	8

Table 4.5: Summary of evaluation results for laptop-data-cleaned data set for a specific question without code interpreter

Model	Temperature	Q ID	Item	Score
gpt-4o-mini	1.0	27	2	11
		27	1	30
		27	0	59
gpt-4o-mini	1.0	28	2	19
		28	1	55
		28	0	26
gpt-4o-mini	1.0	29	2	23
		29	1	53
		29	0	24
gpt-4o-mini	1.0	30	2	76
		30	1	22
		30	0	2
gpt-4o-mini	1.0	30.0	2	76
		30.0	1	22
		30.0	0	2
		30.1	2	68
		30.1	1	29
		30.1	0	3
		30.2	2	67
		30.2	1	30
		30.2	0	3

Table 4.6: Summary of evaluation results for weatherAUS data set for a specific question without code interpreter

Model	Temperature	Q ID	Item	Score
gpt-4o-mini	1.0	59	2	66
		59	1	33
		59	0	1
gpt-4o-mini	1.0	60	2	36
		60	1	27
		60	0	37
gpt-4o-mini	1.0	63	2	17
		63	1	63
		63	0	20
gpt-4o-mini	1.0	65.0	2	68
		65.0	1	28
		65.0	0	4
		65.1	2	61
		65.1	1	34
		65.1	0	5

Table 4.7: Summary of evaluation results for duke-forest data set for a specific question without code interpreter

4.3 Task-specific exploration

For the purpose of demonstrating the task-specific measures on certain data science tasks, some questions are targeted alongside the earlier-created metrics. Primarily, detailed evaluations are implemented for the given set of questions below;

```
{"id": 5, "question": "What are the mean and median visitor numbers in 2022
across all attractions?",
"concepts": ["Data Summary"],
"format": "@visitor_summary[visitor_summary] where \"visitor_summary\" is given
as a numeric vector or tibble formated values.",
"file_name": "UK-visitor-numbers.csv", "level": "medium"}

{"id": 21, "question": "Visualize the distribution of score in the dataframe evals.
Is the distribution skewed? What does that tell you about how students rate courses?
Is this what you expected to see?
Why, or why not?", "concepts": ["Data Viz And EDA"],
"format": "@dist_score_evals_viz[dist_score_evals_viz] where \"dist_score_evals_viz\"
is a created data visualization with additional interepretations.",
"file_name": "evals.csv", "level": "medium"}

{"id": 23, "question": "Fit a linear model called score_bty_fit to predict
average professor evaluation score from average beauty rating (bty_avg).
Print the regression output using tidy(). Based on the regression output,
write down the linear model.", "concepts": ["Regression Modeling"],
"file_name": "evals.csv", "level": "medium"}

{"id": 56, "question": "For now consider only the data stored in the vector first.
We assume that the data are independentobservations from N(µ, 2) and wish to conduct
the following hypothesis test: H0 : µ equals 10 vs H1 : µ not equal 10
Define a suitable test statistic and state its distribution, assuming that the
null hypothesis is true. Calculate the associated observed test statistic
Conduct the hypothesis test and clearly state your conclusions.",
"concepts": ["Hypothesis Testing"],
"format": "@aeroplane_one_sample_test[aeroplane_one_sample_test] where
\"aeroplane_one_sample_test\" is the one sample test implementation",
"file_name": "aeroplane.txt", "level": "medium"}

{"id": 65, "question": "Run PCA for this data, with all suitable variables.
Discuss whether it is appropriate to run scaled
or unscaled PCA for this data. ",
"concepts": ["PCA"], "format": "@duke_forest_PCA[duke_forest_PCA]
where \"duke_forest_PCA\" is the fitted PCA and its interpretations",
"file_name": "duke_forest.xlsx", "level": "hard"}
```

For each case, beyond the general characteristics and course gained information, task-specific performances are exemplified to resemble the main idea of the project

4.3.1 Data Summary example

For the `UK-visitors.csv` data, specific question is explored under the created task-specific metric. Since there is no "Accuracy of Interpretation" part, it is evaluated out of 4 only for this example here. The overall average performance out of 100 trials calculated as 3.65 out of 4, in terms of the main metric items. Generally, for each specific item the created response is well performed except the item called "Clarity of Calculation/Interpretation" because of some misleading wordings.

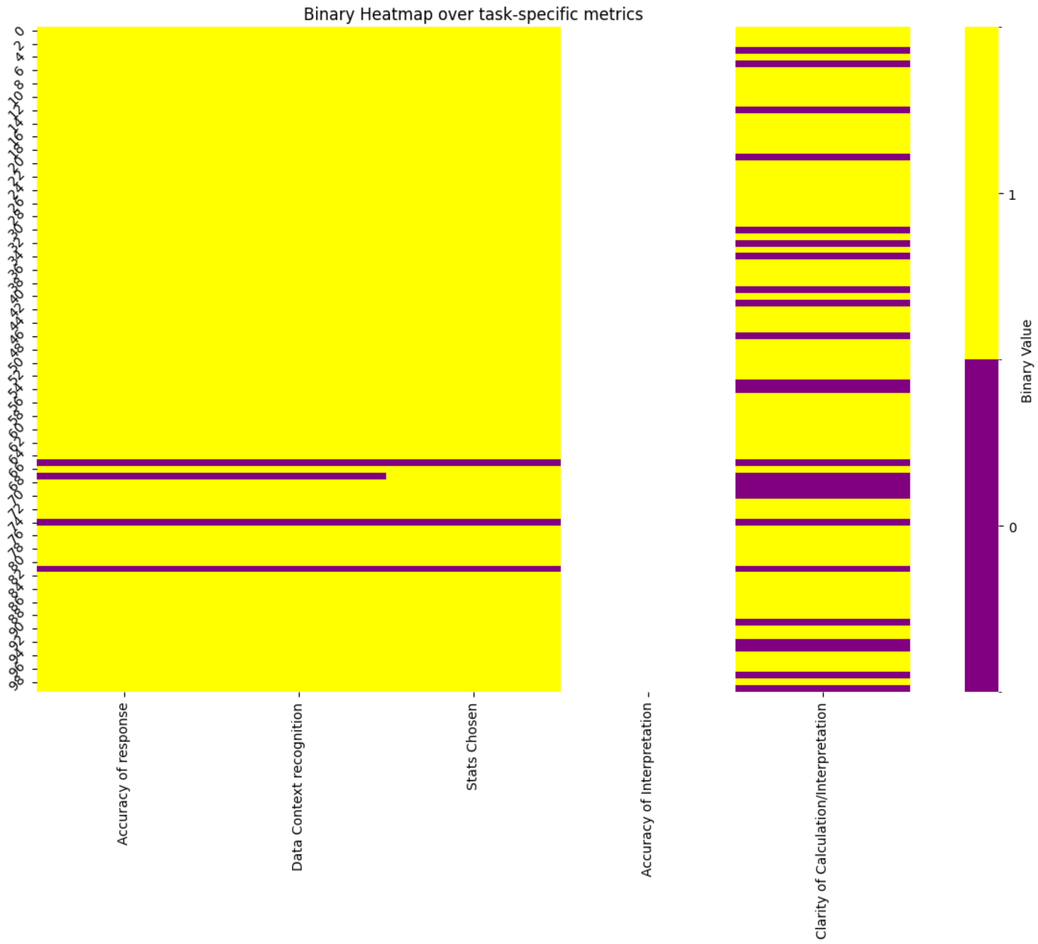


Figure 4.4: Summary of task-specific evaluation for Question "id=5" for `UK-visitors` data

The individual task-specific pattern can be visualized in Figure 4.4. As visually shown, the overall percentage based succes for each item can be summarized as (i) Accuracy of response - 96% , (ii) Data Context recognition - 96%, (iii) Stats Chosen - 97% and (iv) Clarity of Calculation/Interpretation - 76% out of 100 prompt investigations. Overall, it performs well enough on such a task regarding the individual items, and overall summary measure.

4.3.2 Data Viz example

As one of the trickiest cases, the created images for data viz example and related interpretations might fail from time to time. As it was mentioned earlier in Figure 3.1, based on the created thread with the tool, when the image is generated, it was saved as in .png format for further evaluation. Besides, if the question includes any interpretation, it is possible to see these details in the reasoning part of the structurally saved jsonl format output. For this specific question ID, over all performance is measured as 3.27 out of 5 in general.

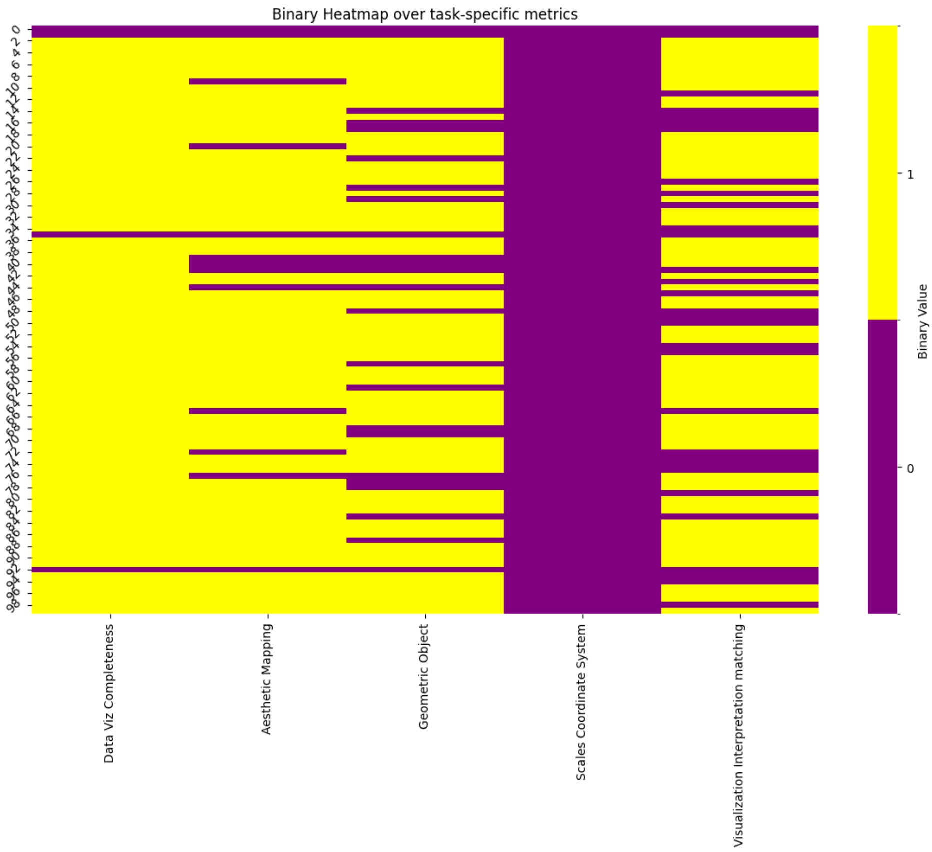
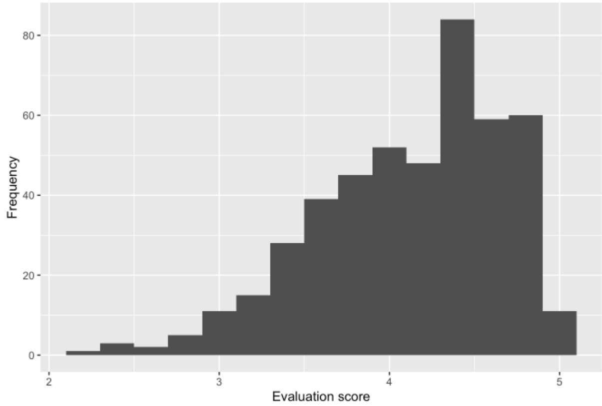
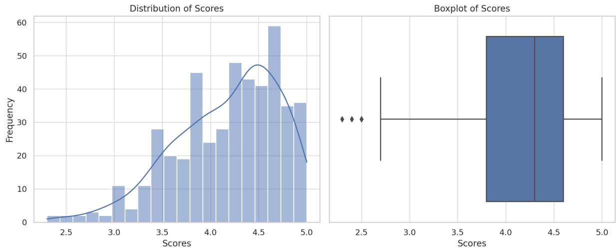


Figure 4.5: Summary of task-specific evaluation for Question "id=21" `evals` data

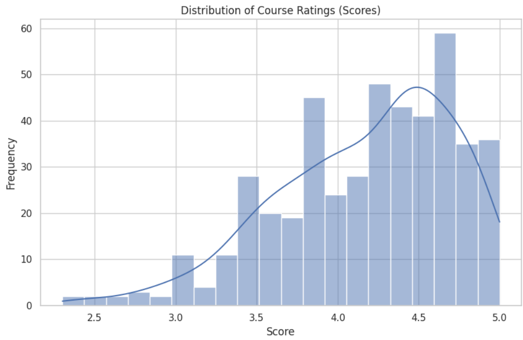
By zooming into the individual items of the metric, we can see some differences and nuances regarding the performance. Most of the time, image is created indeed (as the item called "Data Viz Completeness" shows 96% over 100 trials). However, the other individual items are not performing well compared to the data summary case now. For Aesthetic Mapping we observed 87% since sometimes frequency information is not used for y-axis as anticipated. Additionally, for Geometric Object and Visualization-Interpretation matching we observed some inconsistencies or incorrect usage so they got 76% and 68%, respectively. For some cases, boxplot is added or solely created as different style or there were some not natural gaps in the created histograms. For the interpretation part, the most crucial issue is that, based on the result the tool argue that the shape is right-skewed whereas the original response is lewft-skewed. But the most important issue is that, the item Scales Coordinate System has got literally zero success because of trimming the values somehow in the created histogram so that it never matches with the ground truth image in our comparison. A set of illustrations from the generated images with the original response image is given in Figure 4.6.



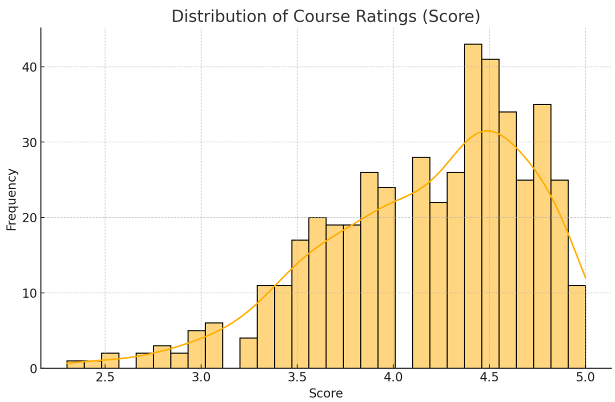
((a)) Original Plot



((b)) Visual at round 22



((c)) Visual at round 51



((d)) Visual at round 78

Figure 4.6: A set of illustrations from the generated images with the original ground truth visualization for question id=21 over evals data set

4.3.3 Regression example

For the simple linear regression we explored in question "id = 23" is also resulted in interesting shape. Generally it was good at various sub-items we used, except a very important nuance in the fitted model.

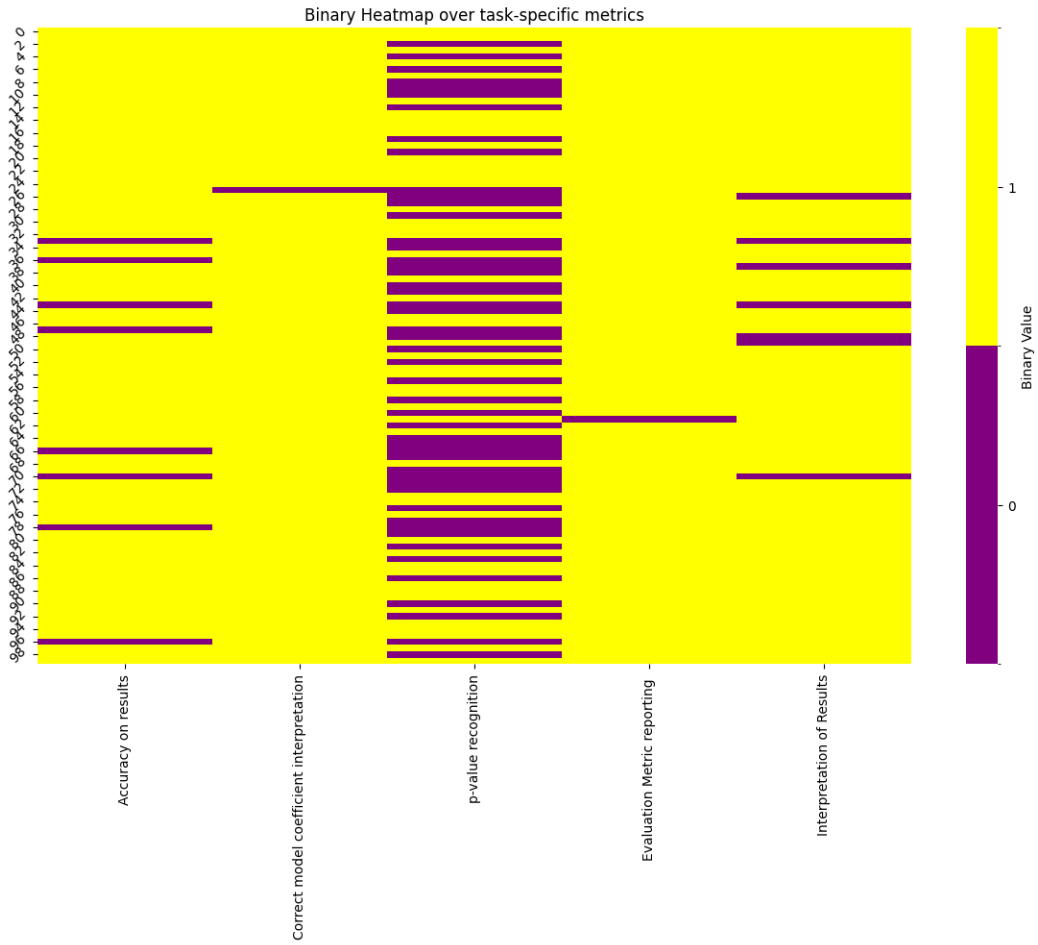


Figure 4.7: Summary of task-specific evaluation for Question "id=23" `evals` data

As one can see in Figure 4.7, there are mostly yellow colors to say large success over the sub-items of Accuracy on results - 92%, Correct model coefficient interpretation - 99%, Evaluation Metric reporting - 99% and Interpretation of Results - 93% beyond looking at accuracy only. However, while presenting the model fit results, the recognition of the p-value is missing almost half of the cases 51% of 100 cases. This was a very important missing nuance regarding the fitted model parameters significance investigation. Overall, the performance can be summarized as 4.34 out of 5 scores.

4.3.4 Hypothesis Testing example

As a specific task type, we explored the performance on the testing procedure on a widely applied test setting, aka one sample t-test example, having certain deviations on specific metric items. Overall, the performance is measured as 4.03 out of 5 scale again.

Figure 4.8: Summary of task-specific evaluation for Question "id=56" `aeroplane` data

As clearly illustrated in 4.8, no issues appeared for the test naming, so "Correct Test/Test name Chosen" has 100% success, whereas there is less consistency on the items such as "H0 and H1 are clearly stated", "Accuracy on Test Stat / p-value / Critical Reg" and "Contextual Conclusion" having values of 73%, 72% and 76%, respectively. Even if for some cases the test numerical results are not matching with the GT, the final interpretation seems reasonable so we had "Correct Interpretation of Test Stat / p-value / Critical Reg" as 82% of the time over 100 times repeated prompting. There were some cases that the solution is incomplete (asking to user to continue), no specific numerical outcome or using the sample mean in the interpretation rather than the population mean, as one of the challenging misuse for learners in general.

4.3.5 PCA example

As the example of increased difficulty, we had the most unsatisfactory outcomes for the PCA implementation over a specific data. The example given here belongs to the `duke_forest` data set with certain numerical and categorical variables.

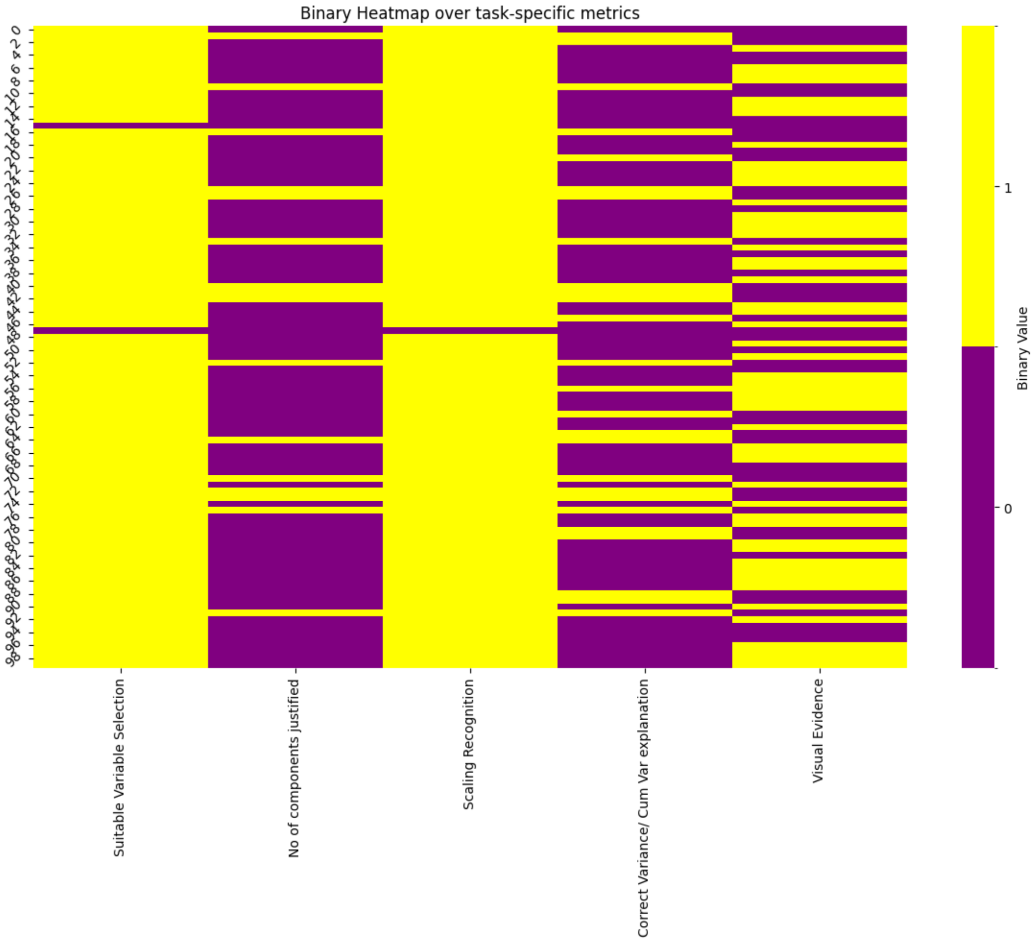


Figure 4.9: Summary of task-specific evaluation for Question "id=65" duke$_forestdata$

The overall performance value is 2.92 out of 5 over 100 trials. Specifically, the created response rarely includes the justification of the selected number of components or the numerical explanation of the explained variance or cumulative variance. Even if the code structure seems following the correct recipe mostly, the generated text response is mostly limited. To illustrate, it is observed that "No of components justified" - 16% and "Correct Variance/ Cum Var explanation" - 26% lower values compared to other items such as detecting the reasonable subset of numerical variables to apply PCA ("Suitable Variable Selection" - 98%), necessary scaling for the variables have different ranges ("Scaling Recognition" - 99%) and even the specific supportive visualization in half of the cases ("Visual Evidence" - 53%).

4.4 Impact of model Setting

Specifically, we implemented different model setting for some selected questions in a limited manner for practical reasons. The tested questions are mainly coming from different concepts and difficulty levels again, namely the question IDs with 56, 26, 26.1, 73 for the concepts of hypothesis testing, regression and correlation related tasks. To highlight the impact of the importance of additional "code interpreter" tool, another setting covers only the base-model "GPT 4o-mini" without the tool itself, implemented a bit different compared to the previous settings. The one simple comparative illustration is given in Table 4.8 for the main idea of this comparison. For the sake of simplicity, this is represented by one question only, but applicable to others and raw results are available in the project GitHub repository.

Table 4.8: Comparison Example of Question id=56 for the data set aeroplane.txt (Item1: Correct Test/Test name Chosen, Item2: H0 and H1 are clearly stated, Item3: Accuracy on Test Stat / p-value / Critical Reg, Item4: Correct Interpretation of Test Stat / p-value / Critical Reg and Item5: Contextual Conclusion)

	Task Specific metrics (%)					
Model	Item1	Item2	Item3	Item4	Item5	Mean Score (out of 5)
GPT-4o with Code Interpreter	100	73	72	82	76	4.03
GPT-4.1 mini Code Interpreter	97	95	45	46	29	3.12
GPT-4o mini wout Code Interpreter	67	55	0	0	0	1.22

As it shown in Table 4.8, the addition of the code interpreter as a tool or not really matters regarding the performance of the created responses. Compared to the case of "GPT-4o with code interpreter", the results of the case "GPT-4.1 mini with code interpreter" is better at capturing certain nuances but interestingly as a result of multiple time data reading/parsing issues, various threads are incomplete. One interesting issue is that, at the data reading/parsing stage, small detail in the .txt format file resulted in multiple incomplete cases that requires human intervention and additional prompt to guarantee the correct outcomes. Because of that reason numerical part and interpretation is comparatively lower than the "GPT-4o with code interpreter" setting but overall, the results are way better than the case of NO code-interpreter definitely. Although some threads are incomplete, generally the model asks user to either run manually or order to run for the model, including well written and executable code snippets with details. However, the case of "GPT-4.1 mini wout code interpreter" performed very poor regarding the quality of the created responses in terms of the calculations and interpretations. Most of the time, it created only the general steps and suggestions witout clearly presenting the numerical results since the calculation could not executed specifically. Additionally, regarding the items of the task specific metrics, sometimes the recognition of the test is not correct (37 out of 100 failing since it picks the standard normal as the testing rather than the t-test.) Similarly, a specific nuance of stating the null and alternative is sometimes missing, poorer than the "GPT-4o with code interpreter".

The raw results for the model setting "GPT-4.1 nano" is available in the GitHub repository but not added here for the sake of simplicity. Overall, this simple comparison points out the importance of adding code interpreter as a tool with the presented empirical results.

4.5 Impact of Temperature parameter

To discuss the impact of the temperature, we started with the initial large value exploration. For the value of "temperature = 1.5", for a simple question like "How many tourist attractions are there in the data set?", it immediately started to create non-sense content both text and code wise, nothing reasonable to evaluate. This resembles the fact that certain model parameter is very crucial in general, where we decided to explore the impact over this small set of values very simply, $(0.2, 0.7, 1.0, 1.2)$.

For this impact, we looked at another regression related question covering both the model fitting and the explanation (question id=26, 26.1) simply without presenting numerical performance. The considered sequential question is given as follows;

```
{"id": 26, "question": "Fit a multiple linear regression model, predicting
average professor evaluation score based on average beauty rating and gender.
Name the model score_bty_gender_fit. Interpret the intercept and the slopes
of beauty rating and gender.Make a scatterplot
(using jitter) of score by bty_avg and color the points by gender.",
"concepts": ["Regression Modeling"],
"format": "@multiple_linear_model[multiple_linear_model]
where \"multiple_linear_model\" is a
fitted multiple linear regression with specific data visualization",
"file_name": "evals.csv", "level": "hard"}

{"id": 26.1, "question": "What percent of the variability in score is explained
by the model score_bty_gender_fit.
What is the equation of the line corresponding to
just male professors? How does the relationship between beauty and evaluation
score vary between male and female professors?",
"concepts": ["Regression Modeling Interpretation"],
"format": "@multiple_linear_model_explain[multiple_linear_model_explain] where
\"multiple_linear_model_explain\" is a list of explanations based
on the fitted multiple linear regression",
"file_name": "evals.csv", "level": "hard"}
```

Below is a list of raw representation, for a specific rounds for the illustrative purposes. The base model considered under the hood is "GPT-4o-mini" with different temperature values. Specifically, the rare results are highlighted below after carefully examining the created responses for each case.

For the case of question id=26 with model "GPT-4o-mini"

General notes: There is a misconnection between the parts of sequential questions and generally the code is created with suggestions but could not answer directly. So even if the response included the model fitting via `sm.OLS()` function in python, it can not retain the numerical results for the second part of the question, possibly this might be a memory issue appeared when the multiple prompting on the sequential-style question, differently from the question id=23, where there are numerical answers there.

One coding example for the round 5 (for the case of "temperature = 0.2") for the first part of the question is given below:

```
‘‘‘python
import pandas as pd
import statsmodels.api as sm
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np

# Assuming the data is in a DataFrame named df with columns:
# ’score’ for average professor evaluation score,
# ’bty_avg’ for average beauty rating,
# ’gender’ for gender (categorical, e.g., ’Male’/’Female’)

# Example data loading (replace with actual data loading)
# df = pd.read_csv(’your_data.csv’)

# Convert gender to a numeric dummy variable (e.g., Female=0, Male=1)
df[’gender_num’] = df[’gender’].map({’Female’: 0, ’Male’: 1})

# Define predictors and response
X = df[[’bty_avg’, ’gender_num’]]
X = sm.add_constant(X)  # Adds intercept term
y = df[’score’]

# Fit the multiple linear regression model
score_bty_gender_fit = sm.OLS(y, X).fit()

# Print the summary to interpret coefficients
print(score_bty_gender_fit.summary())

# Interpretation:
# Intercept: Expected average score when bty_avg=0 and gender=Female
(reference group)
# Slope for bty_avg: Change in average score for one unit increase in beauty rating,
holding gender constant
# Slope for gender_num: Difference in average score between Male and Female,
holding beauty rating constant

# Scatterplot with jitter
plt.figure(figsize=(10, 6))
sns.scatterplot(
    x=df[’bty_avg’] + np.random.uniform(-0.1, 0.1, size=len(df)),  # jitter on x
    y=df[’score’] + np.random.uniform(-0.1, 0.1, size=len(df)),    # jitter on y
    hue=df[’gender’],
    palette=’Set1’,
    alpha=0.7
)
plt.xlabel(’Average Beauty Rating (with jitter)’)
plt.ylabel(’Average Professor Evaluation Score (with jitter)’)
plt.title(’Scatterplot of Score by Beauty Rating Colored by Gender’)
```

```
plt.legend(title='Gender')
plt.show()
```

There is an attempt to create the image also with jittering above, beyond the fitting. The code itself seems reasonable for the use of `sm.OLS()` function, whereas it is not directly executable since the code snippet part

```
{df['gender_binary'] = df['gender'].map({'Female': 0, 'Male': 1})
```

created a column with NaN values and this avoids the model fitting and further visualizations in the given code. In the second part also, the suggested code snippet includes different variable names so did not result in numerical answer. This miscommunication issue between two parts of the question appeared repeatedly.

For some rounds, instead of creating extra issue with the line of

```
df['gender_binary'] = df['gender'].map({'Female': 0, 'Male': 1})
```

, it correctly converts the gender into categorical variable with

```
df['gender_num'] = pd.Categorical(df['gender']).codes
```

and this results in some numerical answer indeed (when it is executed in a different session). On the other thand, for the second part of the question, it can not retain these results still and giving just a verbal suggestion like this

```
I could not find direct textual results about the model "score_bty_gender_fit"
in the uploaded files. However, I can provide a general Python snippet that:

- Calculates the percent of variability explained by the model (R-squared).
- Extracts the equation of the regression line for male professors.
- Explains how the relationship between beauty and evaluation score varies
by gender using interaction terms.

If you provide the model's coefficients or the dataset,
I can tailor the snippet exactly.
```

For the round of 14, even if the created code is not directly executable because of similar variable transformation issue as it mentioned above, the second part of the question includes some hallucinations now;

```
I found the relevant information about the model `score_bty_gender_fit`
in the uploaded files. Here are the key points:

1. **Percent of variability explained by the model (R-squared):**
   The R-squared value of the model is approximately 0.13 (13%),
   meaning the model explains about 13% of the variability
   in the evaluation score.
```

whereas the original value is totally different from the above mentioned R-squared value indeed. Similar patterns of issues appeared in general, including some cases of creation a sample dataframe for the second part of the questions because of this retaining problem. Differently from the simple linear regression case (id=23), under the model of "GPT-4o-mini" with "temperature = 0.2", the performance was really failing. Similar problem persisted for the other temperature values in general, under the same model setting.

For the case of question id=21 with model "GPT-4o-mini"

As an extra illustration for a question with data visualization, we explored briefly the raw results of the question id = 21, that is previously defined under the task-specific performance section.

For the case of "temperature = 0.2", generally the response includes only the code creation without any numerical result or comment on the skewnees. Code snippet generally includes the calculation of skewness like this

```
# Check skewness
skewness = evals['score'].skew()
print(f"Skewness of score distribution: {skewness}")
```

but it creates just the code snippet, without having the actual calculation. Only for some limited rounds (such as round 44), there are generic comments about the interpretation of the skewness for the given data,

```
print("""
Interpretation:
- If the distribution is right-skewed, it indicates that most students
give relatively high scores,
  but there are a few low scores pulling the distribution tail to the left.
- If left-skewed, most scores are low with a few high scores.
- This can tell us about the general tendency of students to rate courses,
whether they are generally positive, critical, or balanced.

Is this expected?
- Often, course evaluations tend to be skewed towards higher scores because
students may be reluctant to give very low ratings.
- However, this can vary depending on the institution and course difficulty.
""")
```

For the case of "temperature = 0.7", now there are some rounds include both the result and the interpretation. However, one of the characteristics of the responses is that the tool itself needs further confirmation by the user to proceed and give the results, rather then presenting them all directly. Generally, similar to the case of "temperature = 0.2", it creates the code without result or explanation. For some rounds like round 13, there is solution in the created response but it happens rarely

```
...
{
  "outcome": "The distribution of the score is visualized with a histogram
  and KDE curve. The distribution appears to be left-skewed (negatively skewed)
  with a skewness value of approximately -0.70. This suggests that most students
  tend to give relatively high scores to courses, with fewer low scores.
  This is somewhat expected as students might generally rate courses positively,
  possibly due to grading leniency, course enjoyment, or instructor effectiveness.
  However, the left skew indicates that there are some courses or instructors
  receiving notably lower scores, pulling the distribution slightly to the left."
}
...
```

However, in other rounds, it is either asking to user to proceed or giving general suggestions about the code execution to collect the outcomes

```
Round 14 with thread_id: thread_uPKyLxFS6DBXRMuwzjEyqZMr
...
Would you like me to run this code and provide the visualization and interpretation?
...
Round 16 with thread_id: thread_egxEi64hwVpUmipkUswhdvp0
...
This code loads your data, plots the distribution of the 'score' column,
and prints the skewness of the distribution to help assess if it is skewed.
```

Roughly counting, only the 14 out of 100 trials include the real result and corresponding interpretation it seems, without the created/saved image, only giving the code snippet to execute manually. Among those outcomes, even if the ground truth is left-skewed for the distribution shape, 1 out of 14 created outcomes it interpreted as "right-skewed" still.

For the case of "temperature = 1.0"- originally the default parameter setting in the base calculations, similar "code only without results" persists interestingly and differently from the "GPT-4o" model results. Surprisingly, the results include real outcome only 2 out of 100 trials, wheres the rest requires further communication with user to get the numerical results again only including the code snippet without any images. Similar problems continued when the temperature is increased to "temperature = 1.2", but now outcomes include some results 11 times including both skewness value and comments about the type. However, there are some cases indicating the distribut, sometimes only python code still. ion is right-skewed, real failure in terms of the ground truth and the interpretation of results regarding the considered task type. These simple observations are brief empirical comments on the importance of the selected based-model and the model parameters, compared to the results of "GPT-4o" with code interpreter discussion under the task specific results in Section 4.3 for the same question.

Chapter 5

Discussion and Analysis

This systematic experiment framework is implemented for the purpose of testing the performance of certain LLMs over the set of created data science related tasks, aka, certain questions that are valid in the field. Based on the curated questions, the repeated prompting impact is explored over various question input-output pairs to investigate how LLMs with code-interpreter tool is successful, how it is changing across different concept or difficulty level questions.

5.1 Proposed Metrics

Differently from the available literature, rather than simply looking at aggregate metrics or programmatic rules, we investigated performances under three main components, including the task-specific metric we explored on certain tasks ranging from data summary to unsupervised learning (PCA) method.

The study utilized API-based interactions with large language models (LLMs) to systematically send prompts and collect responses, though inconsistencies like missing code components were noted. General response characteristics and coarse-grained metrics such as similarity indices were easily computed, while code executability was tested separately, with results detailed in Section 4.2. Performance varied across tasks and concepts, as summarized in Section 4.3. The inclusion of a code interpreter significantly impacted outcomes, especially in zero-shot scenarios, as shown in Table 4.8 (Section 4.4). Additionally, both temperature settings and base model selection played crucial roles, with comparative observations across two tasks discussed in Section 4.5.

5.2 Significance of the findings

Considering the completion rate or certain course-grained characteristics of the LLM responses can be limited but adding task-specific metric evaluation can enrich the discussion on the LLM performance on the data science tasks. Besides, how they are varying across the level of difficulty, concept change or model impact can be beneficial for the further studies on the data science automation with LLMs and attached certain tools such as code-interpreter. Last but not least, the curated specific questions from the field of data science / statistics primarily rely on the human expert writings, rather than the LLM creation or Human-LLM collaboration.

Overall, the considered models are very promising on the certain tasks, but a bit limited in terms of the data visualization and the interpretation of the created responses. Whenever the complexity of the task is increased and certain nuances are appeared, the created responses might not be capturing these aspects under the tested scenarios, such as PCA example.

Beyond the considered tasks, the injection of code interpreter as an additional tool plays a crucial role otherwise the created response might be just the instructions and the code snippets to run under the zero-shot learning prompting environment. Additionally, the clarity level of the prompt plays an important role to cover certain nuances in the generated responses otherwise, they might be missing randomly such as the p-value recognition for the fitted regression model as we exemplified. In that respect, it is possible to improve the problem-solving ability of LLMs through adding detailed instructions specific to each concept, that brings us the either prompt with specific details or few-shot style prompting necessity to increase the performance.

5.3 Limitations

In terms of the created set of questions, there is a certain space to improve the diversity regarding the concepts and the difficulty level we explored. From that point of view, the curated data set is open to be enhanced by adding additional question-answer pairs to increased the quality of data. Rather than adding new human-written input data, it is possible to merge certain open data sets for enriching the curation to implement further experimentation.

Beyond the size of data set, this study mainly focused on the certain GPT models with and without code interpreter intentionally. However, as a result of the time constraints, all the model setting alternatives could not applied to the whole data set. For this reason, one possible limitation is that, the findings are belonging to certain tasks and models, wheres the importance of repeated prompting is highlighted still as the main goal of the project. To add more flexibility, the evaluation process of task-specific metrics can be partly automated with the help of LLM but for this project, it was not preferred because of the main objection.

Another limitation, as a result of the time and human resources constraint, the proposed task-specific metrics are limited to certain tasks, even if there are proposed more in the project. For some tasks, like logistic regression, there is a suggested framework but the evaluation results are not available so that part can be enhanced naturally for the further publication purposes.

5.4 Further Possibilities

From the data automation perspective, definitely the zero-shot learning with certain models have to be improved via other prompt engineering, and/or the combination of the different LLMs by calculation and checking in a sequential manner. Regarding the obtained systematic and raw findings that we are presenting, for certain tasks, their initial response might not be enough and also the interaction with the user either should be designed in advance or human in the loop approach must be the part of the process. In that respect, by using the curated data set by combining new ones or adding open-source alternatives, it is reasonable to explore the agent-based system to create a sophisticated environment for tackling data science tasks efficiently, aligning with some recent literature and performance limitation on our results.

Regarding the further improvement, one other possibility is first fine-tuning the best performing model and then testing its performance over the curated data sets. With that approach, fine tuned models over the statistics/data science task related data sets can be more powerful over new questions and can capture certain nuances of the field. As a less-cost alternative of this, certain prompt engineering strategies can be implemented so that the performance of the model setting and the quality of the generated response can be increased without fine-tuning process.

Another analogy can be made, from a teaching perspective, to consider giving draft marking and feedback for multiple responses of students. Rather than considering the multiple responses from LLM, assume that the specific question has 100 responses from students under a similar setting. By crafting the multi-agent style environments for the purpose of both evaluating results by comparing to the ground truth (solution key) and looking at certain rubric items (like task-specific items) to give draft marks, LLM empowered local marking helpers can be imagined. Additionally, if these tools are created in local environment without any data privacy issue, for specific questions can be analyzed locally by instructors based on their solutions and created individual rubric items. For sure, LLMs nature do not guarantee correct outcomes all the time as we explored, this mechanism should be designed by thinking the human-in-the-loop approach.

References

Bayne, S. (2015), 'Teacherbot: interventions in automated teaching', *Teaching in Higher Education* **20**(4), 455–467.
URL: *http://dx.doi.org/10.1080/13562517.2015.1020783*

De Bie, T., De Raedt, L., Hernández-Orallo, J., Hoos, H. H., Smyth, P. and Williams, C. K. I. (2022), 'Automating data science', *Communications of the ACM* **65**(3), 76–87.

Evkaya, O. and de Carvalho, M. (2024), 'Decoding ai: The inside story of data analysis in chatgpt'.

He, X., Zhou, M., Xu, X., Ma, X., Ding, R., Du, L., Gao, Y., Jia, R., Chen, X., Han, S., Yuan, Z. and Zhang, D. (2023), 'Text2analysis: A benchmark of table question answering with advanced data analysis and unclear queries'.

Hu, X., Zhao, Z., Wei, S., Chai, Z., Ma, Q., Wang, G., Wang, X., Su, J., Xu, J., Zhu, M., Cheng, Y., Yuan, J., Li, J., Kuang, K., Yang, Y., Yang, H. and Wu, F. (2024), 'Infiagent-dabench: Evaluating agents on data analysis tasks'.

Huang, Q., Vora, J., Liang, P. and Leskovec, J. (2023), 'Mlagentbench: Evaluating language agents on machine learning experimentation'.

Jain, N., Han, K., Gu, A., Li, W.-D., Yan, F., Zhang, T., Wang, S., Solar-Lezama, A., Sen, K. and Stoica, I. (2024), 'Livecodebench: Holistic and contamination free evaluation of large language models for code'.

Lai, Y., Li, C., Wang, Y., Zhang, T., Zhong, R., Zettlemoyer, L., Yih, S. W.-t., Fried, D., Wang, S. and Yu, T. (2022), 'Ds-1000: A natural and reliable benchmark for data science code generation'.

Ping, Q., Niu, F., Thattai, G., Chengottusseriyil, J., Gao, Q., Reganti, A., Rajagopal, P., Tur, G., Hakkani-Tur, D. and Nataraja, P. (2020), 'Interactive teaching for conversational ai'.
URL: *https://arxiv.org/abs/2012.00958*

Zeng, Q., Jin, C., Wang, X., Zheng, Y. and Li, Q. (2025), 'An analyst-inspector framework for evaluating reproducibility of llms in data science'.

Zhang, D., Zhoubian, S., Cai, M., Li, F., Yang, L., Wang, W., Dong, T., Hu, Z., Tang, J. and Yue, Y. (2025), 'Datascibench: An llm agent benchmark for data science'.

Zhang, S., Zhao, H., Liu, X., Zheng, Q., Qi, Z., Gu, X., Zhang, X., Dong, Y. and Tang, J. (2024), 'Naturalcodebench: Examining coding performance mismatch on humaneval and natural user prompts'.

Zhu, Y., Du, S., Li, B., Luo, Y. and Tang, N. (2024), 'Are large language models good statisticians?'.

Zhuo, T. Y., Vu, M. C., Chim, J., Hu, H., Yu, W., Widyasari, R., Yusuf, I. N. B., Zhan, H., He, J., Paul, I., Brunner, S., Gong, C., Hoang, T., Zebaze, A. R., Hong, X., Li, W.-D., Kaddour, J., Xu, M., Zhang, Z., Yadav, P., Jain, N., Gu, A., Cheng, Z., Liu, J., Liu, Q., Wang, Z., Hui, B., Muennighoff, N., Lo, D., Fried, D., Du, X., de Vries, H. and Von Werra, L. (2024), 'Bigcodebench: Benchmarking code generation with diverse function calls and complex instructions'.

Appendix A

Data Set and Code Availability

All the curated questions (input-output) pairs with their specific characteristics are available on the project github repository. For some of the questions having a direct answer, the list of ground truth is created within the same repo but some questions do not have specific answers because of their open-ended style construction.

For the experimentation, list of inter-communicated python files, the question related data set sources, and main results based on the repeated prompting experiments are available via https://github.com/oevkaya/GAIL-DS-project for the interested reader to check, and share their critics and suggestions.