Capture the Flag:
Uncovering Data Insights with Large Language Models

**Issam Laradji, Perouz Taslakian, Sai Rajeswar, Valentina Zantedeschi, Alexandre Lacoste,
Nicolas Chapados, David Vazquez, Christopher Pal, Alexandre Drouin**
ServiceNow Research
Montréal, Canada
`firstname.lastname@servicenow.com`

Abstract

The extraction of a small number of relevant insights from vast amounts of data is
a crucial component of data-driven decision-making. However, accomplishing this
task requires considerable technical skills, domain expertise, and human labor. This
study explores the potential of using Large Language Models (LLMs) to automate the
discovery of insights in data, leveraging recent advances in reasoning and code gener-
ation techniques. We propose a new evaluation methodology based on a "capture the
flag" principle, measuring the ability of such models to recognize meaningful and
pertinent information (flags) in a dataset. We further propose two proof-of-concept
agents, with different inner workings, and compare their ability to capture such flags
in a real-world sales dataset. While the work reported here is preliminary, our results
are sufficiently interesting to mandate future exploration by the community.

1 Introduction

Many organizations—businesses, government agencies, and academic research groups—accumulate
vast amounts of diverse data with the intent of using it for decision-making [McAfee and Brynjolfsson,
2012, Colson, 2019, Bean, 2022]. Accumulated historical data can empower organizations to acquire
valuable insights, make informed decisions, and be used to make predictions for future business
scenarios and their respective likelihoods [Colson, 2019]. Such insights can allow more effective
interventions in tasks that are typically conducted based on intuition [McAfee and Brynjolfsson,
2012]. However, the true value of this data only becomes apparent when individuals possess the
necessary resources, time, and expertise to fully leverage it [Bean, 2022]. Extracting meaningful
visualizations, generating insightful summaries, and identifying anomalies all require a significant
level of skill [McAfee and Brynjolfsson, 2012]. Moreover, interpreting these insights effectively
extends beyond data manipulation, calling for extensive domain expertise [Colson, 2019].

In this work, we envision autonomous data-science agents capable of extracting insights and interpreting
them within context; such agents would enable individuals with low data-science expertise (e.g., busi-
ness decision-makers) to make the most out of their data. We further seek agents capable of surfacing
insights of key interest, such as to limit the cognitive burden of decision-makers and mitigate the effect of
cognitive biases [Colson, 2019]. We argue that approaches that rely solely on statistical analysis, such as
the *Automatic Statistician* [Steinruecken et al., 2019], are of limited interest for this task since they lack
the background knowledge required to interpret their findings in context. Rather, we hypothesize that
Large-Language models (LLMs), such as ChatGPT [OpenAI, 2023b, Shen et al., 2023] and GPT-4 [Ope-
nAI, 2023a], may serve as a good basis for such agents in light of recent advances in reasoning [Wei et al.,
2022], code generation [Olausson et al., 2023], and given the breadth of their knowledge bases [Bubeck
et al., 2023]. This direction has been increasingly explored by the community, with some success, e.g.,
LiDA [Dibia, 2023], GPT4-Analyst [Cheng et al., 2023]), and Sheet-Copilot [Li et al., 2023a].
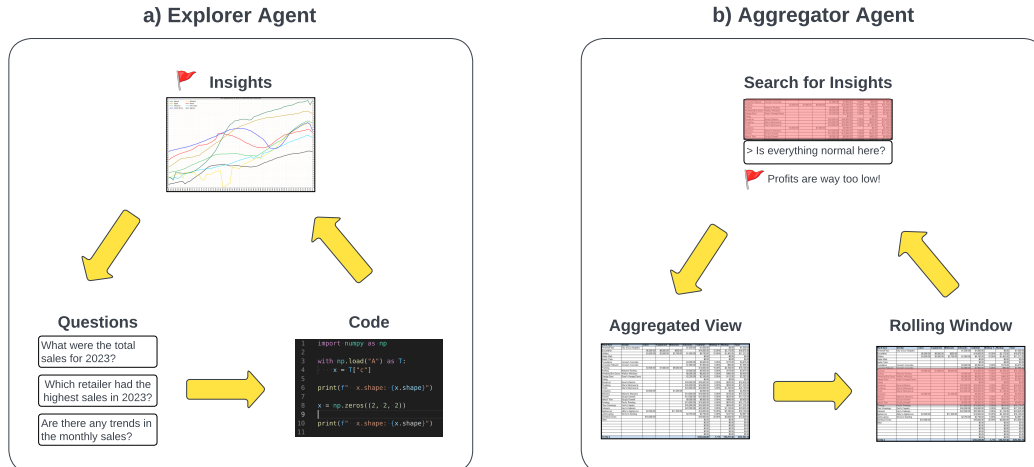
Figure 1: *Overview of our data science agents. a) The Explorer agent, which generates questions and writes code to answer them; b) The Aggregator agent, which produces various aggregations of the data and then looks at snippets of the aggregated data, pointing out anything it finds relevant. Both agents can perform multiple cycles, using the discovered insights to guide future exploration.*

To facilitate the development of such agents, we propose a new evaluation methodology, based on a *capture the flag* principle. In contrast with the common approach, which consists in evaluating the correctness of code generated by agents [Huang et al., 2022, Yin et al., 2023, Shinn et al., 2023], we focus on the end result and assess the ability of agents to recover key insights, regardless of how they were obtained. This results in a general approach that can be used to evaluate all kinds of agents, irrespective of their internal mechanisms. Our contributions are as follows.

Contributions:

- We propose a *capture the flag* approach to evaluating data science agents (Sec. 3)
- We elaborate two LLM-based proof-of-concept agents for this task (Sec. 4)
- We assess the ability of such agents to capture flags planted in real-world sales data (Sec. 5)

We wish to highlight that this work serves as a proof-of-concept, intended to inspire further developments in the community, rather than providing a comprehensive methodology and benchmark.

2 Related Work

Automatic Data Analysis: Considering the magnitude of skills and expertise needed in data science, it is essential to develop automated systems for this process [Aggarwal et al., 2019, Dhamdhere et al., 2017, Raedt et al., 2018, Dibia and Çagatay Demiralp, 2018]. In their work Steinruecken et al. [2019], Meduri et al. [2021] make an effort to streamline multiple aspects of data science, including the automated creation of models from data and the generation of reports with minimal human involvement. These reports not only incorporate fundamental plots and statistics, but also feature human-friendly explanations presented in natural language. Recent progress in AI suggests that LLM-based analysis assistants are a promising solution, enabling data analysts to execute and automate their analyses [Chandel et al., 2022, Yin et al., 2023]. Dibia [2023], Maddigan and Susnjak [2023] investigate GPT's capabilities in the realm of visualization tasks, primarily concentrating on its proficiency in code generation for data visualizations. Gu et al. [2023] provides design guidelines to empower end-user analysts interacting with AI-based analysis assistants.

Code Generation: LLMs can generate highly expressive programs in response to natural language prompts. These programs are proficient in addressing a wide range of tasks spanning competitive programming [Li et al., 2022], data science [Lai et al., 2022, Huang et al., 2022, Wang et al., 2022], all guided by high-level descriptions. Code generation models based on LLMs [Chen et al., 2021a, Khan et al., 2023, Fried et al., 2022, Agashe et al., 2019] alongside program synthesis techniques [Chaudhuri et al., 2021, Zhang et al., 2021] empower users to accomplish tasks without the need for traditional

programming approaches. Code generation for data analytics have seen a rise in research efforts, tackling important issues in automating the creation of data analysis pipelines. Lai et al. [2022], Huang et al. [2022] propose a benchmark in the Data Science domain that can be evaluated using human-written test cases of diversified Python libraries. Yin et al. [2023] investigate Python Jupyter Notebooks as the preferred platform for interactive coding, while limiting permissible actions to a restricted scope of code and libraries. Chen et al. [2021b] generate visualization code from natural language but limit options to a subset of Matplotlib. In this work, we deviate from the code-generation setting as we attribute more focus to the solution, rather than how it was obtained.

GPT for Data Analysis: GPT models have the potency to proficiently analyze, and extract valuable insights from extensive unstructured text datasets [Gravitas, 2023]. Chen et al. [2023] examine GPT models (GPT-3.5 and GPT-4) in the context of a data visualization course, shedding light on their advantages and drawbacks. Ma et al. [2023] introduce the InsightPilot system, an automated data exploration tool leveraging LLMs to streamline the data exploration process. It achieves this by automatically identifying suitable analysis objectives and generating corresponding targeted queries, known as intentional queries (IQueries). Cheng et al. [2023], Wang et al. [2023] evaluate GPT-4's data analysis abilities and introduces an end-to-end automation framework for data processing.

Anomaly Detection: The problem of anomaly detection has been an active area of study with important applications in many industries. Statistical approaches for finding outliers and, more recently, deep learning methods have shown great success in addressing the problem [Salehi et al., 2022, Ruff et al., 2021, Hendrycks and Gimpel, 2017, Yang et al., 2022]. However, anomaly detection remains a challenging task, as the definition of what constitutes an anomaly is vague: it often depends on the industry and requires specialized background to detect which of the outliers are in fact anomalous [Foorthuis, 2020]. However, the rarity of anomalies in real-world scenarios poses a challenge for machine learning training due to the lack of labeled data samples [Shen et al., 2020, Kumagai et al., 2019]. In this work, we leverage the background knowledge of LLMs to help identify anomalies.

3 Capturing the Flag as an Evaluation Methodology

We propose to evaluate data science agents based on their ability to recover salient facts in data. Our evaluation disregards the methodology utilized by the agents in discovering such facts, thereby enabling the assessment of agents with varying inner workings. Our approach is straightforward. We select an arbitrary dataset and manually introduce corruption to incorporate deviations from what could be expected in the context of the dataset. We term such deviations *flags*. We then evaluate the ability of agents to recover the planted flags. This approach is inspired by the one used in cybersecurity competitions [1], where participants are challenged to detect and recover flags representing vulnerabilities or malicious patterns in software systems.

What Makes a Good Flag? A good flag is an observation that could be useful for decision-making. Therefore, flags that correspond to insignificant data corruption, such as invalid dates like Feb. 31st, are not considered. Instead, we favor flags that: (i) have a plausible contextual explanation, and (ii) necessitate some background knowledge to identify. For example, in a clothing sales transactions dataset, a flag could be created by inflating swimwear sales in January beyond their June levels. This flag is plausible, as it could result from a marketing campaign, but domain knowledge tells us that most people purchase swimsuits during the summer, not the winter.

Adidas Sales Dataset: Throughout the rest of this work, we use the *Adidas Sales dataset* [Chaudhari, 2023] as a running example. This dataset contains 14 attributes and 9652 entries corresponding to daily information on the sales of various Adidas products, including the number of units sold, the total sales amounts, profit margins, etc. We manually incorporate the following flags in the data and assess the ability of our proposed LLM-based data science agents to recover them:

 🏴 1. The profit margins for all retailers in the state of Arizona are extremely low (0.1%)

 🏴 2. The state of Alaska has higher total sales than California despite its smaller population

 🏴 3. On one given day, one retailer in Los Angeles sold an enormous quantity of men's footwear

These flags are interesting because they require various levels of background knowledge, as well as various levels of data aggregation to be detectable. For instance, detecting Flag (1) requires knowing

---

[1]See example competition here: `https://infosec-conferences.com/event-series/def-con/`

that a store should normally make more than 0.1% profit; Flag (2) requires knowledge of the population of various states and can only be detected when aggregating sales data at the state level; and Flag (3) is a localized event that requires a fine-grained look at the specific data sample.

Towards a Benchmark: This preliminary work aims to assess the feasibility of creating a large-scale benchmark based on the capture-the-flag methodology. As such, we focus on the Adidas Sales dataset and defer a generalization to other datasets for future research. Additionally, we plan to address extensions, such as elaborating a methodology to account for flags that could naturally arise in such datasets in our future work.

4   Data Science Agents

As a proof of concept, we propose two data science agents, with different inner workings, and assess their ability to recover flags planted in data. The prompts used by our agents are resolved by OpenAI API calls to models such as GPT-3.5 [OpenAI, 2023b]. An overview of these agents is provided in Fig. 1 and their inner workings are detailed below.

4.1   Explorer Agent

This first agent relies on code generation. It follows a *top-down* approach: analyzing the schema of a dataset and generating a few questions, which it answers through code generation. Then, it generates more questions, digging into anything that it deems interesting or surprising, and the process is repeated. After multiple rounds of questions, it reports any noteworthy insights. A detailed pseudocode is available in Algorithm 1 and the prompts used at each step are given in App. A.1.

Limitations: The main limitation of Explorer Agent is that it must ask a series of high-level questions in order to identify traces of the flags and then, in subsequent rounds of questions, dig until it fully uncovers a flag. When flags exist at a high level, e.g., Flag (2) in Sec. 3, this is not a problem. However, flags corresponding to anomalous events that occur on one given day can easily be missed if the wrong questions are asked, e.g., Flag (3) in Sec. 3.

Algorithm 1 ExplorerAgent

Require: rawData, generalGoal, nRounds, dataContext
Ensure: Top insights about the dataset
 1: **Initialize:** questions ← {}, answers ← {}, insights ← {}
 2: **while** not Reached(nRounds) **do**
        ▷ Stage 1: Generate questions using the goal and the accumulated insights
 3:     questions ← AskLLM(rawData, generalGoal, insights, dataContext) as given by the prompt at App. A.1.1
        ▷ Stage 2: Generate code for each question
 4:     **for** each question in questions **do**
 5:         code ← GenerateCodeToAnswer(question, dataContext)
 6:         answer ← ExecuteCode(data, code)
 7:         answers.add(answer)
 8:     **end for**
        ▷ Stage 3: Extract insights from answers
 9:     insights.add(ExtractInsights(answers))
10: **end while**
11: **Return** TopInsights(insights) as given by the prompt at App. A.1.2

4.2   Aggregator Agent

The second agent uses a *bottom-up* approach that completely differs from the previous one. It begins by using code generation to produce various aggregations (views) of the data. Then, it scans each of these views using a sliding window, flagging anything that appears abnormal. Of particular interest, the flagging is done without relying on code generation and relies completely on the common-sense domain knowledge built into the model. In familiar terms, one could say that the agent *eyeballs* the data. Interestingly, in contrast to existing code-based evaluation methodologies, our capture-the-flag approach supports such agents. A detailed pseudocode for this agent is available in Algorithm 2 and the prompts used at each step are given in App. A.2.

Limitations: In contrast with the Explorer Agent, which is limited to asking specific questions, a key strength of the Aggregator Agent is that it looks at the data in a more holistic way. However, its main limitation is that its conclusions are not grounded in code. Rather, the data is added to the prompt and

the agent extracts any insight it finds relevant. Proceeding in such a way makes the agent vulnerable to hallucinating facts. Further, the reliance on a sliding window approach may prevent the agent from detecting patterns that occur over multiple such windows (e.g., trends that arise over several months).

Algorithm 2 AggregatorAgent

Require: rawData, nAggregations, generalGoal, dataColumns, dataStats, slidingWindow
Ensure: Top insights about the dataset
 1: **Initialize:** aggregationList ← {}, insightCount ← 0, insightList ← {}
    ▷ Stage 1: Generate Aggregations of the Data
 2: **while** not Reached(nAggregations) **do**
 3:     aggregation ← AskLLM(rawData, generalGoal, dataColumns, dataStats) as given by the prompt
    in App. A.2.1
 4:     aggregatedData ← AggregateData(rawData, aggregation)
 5:     aggregationList.add(aggregatedData)
 6: **end while**
    ▷ Stage 2: Extract Insights from each aggregation
 7: **for** aggregatedData in aggregationList **do**
 8:     **for** i in range(0, length(aggregatedData), slidingWindow) **do**
 9:         aggregatedDataWindow ← aggregatedData[i:i+slidingWindow]
10:         insights ← ExtractInsights(aggregatedDataWindow) as in App. A.2.2
11:         insightList.add(insights)
12:     **end for**
13: **end for**
14: **Return** TopInsights(insights) using the prompt in App. A.2.3

---

5 Experiments

In this section, we assess the ability of our agents to capture flags planted in the Adidas Sales dataset. We start by reviewing the experimental protocol and then discuss the results separately for each agent.

5.1 Protocol

Our experiments focus on the three flags presented in Sec. 3. For each flag, we create a separate copy of the original Adidas Sales dataset and apply the corruption that corresponds to planting the flag. Given the potential cost associated with OpenAI API calls, we have opted to limit our datasets to 1000 rows, by selecting 100 rows for each of the following states: New York, Texas, California, Illinois, Arizona, Alaska, Colorado, Washington, Florida, and Minnesota. We then run each agent on these datasets and instruct them to rank their insights according to relevance using the prompts shown in App. A.

The extracted insights are then evaluated in terms of whether i) they correspond to planted flags, ii) the data is factual (not hallucinated), and iii) the insight is relevant. In what follows, we review and discuss the results for each agent.

5.2 Aggregator Agent Results

Implementation Details.   We employed the GPT-3.5 model with a sliding window of size 50 to process the input data, generating 20 aggregations for the data and extracting 5 insights from each window. Subsequently, we utilized the GPT-4 model to rank and refine the final set of insights based on how interesting they are to a data analyst as shown in Algorithm 2. See App. A.2 for an explanation of the prompts and the variables used in the Algorithm.

Relevance.   Table 1 presents insights generated by the Aggregator agent, with each insight accompanied by an explanation of its significance. Interestingly, all flags were identified among the top 5 generated insights. For example, for Flag 2, the agent mentions that it was reported since it is surprising that Alaska has the highest sales despite its small population in comparison to California and New York. It is worth noting that the agent also returns insights that are not part of the planted flags, but that appear to be relevant. For example, it reports that the sales amount of retailer West Gear nearly match those of Amazon, which is surprising given that the former is lesser-known. Finally, we observe that the reported insights were extracted from both aggregated data and the original data, supporting the relevance of considering both kinds of data in the analysis.

Consistency.   We have learned that when we prompt the agent to be specific about which columns, values, and row numbers it bases its insights on (as demonstrated in the Prompts section in the Appendix A.2), the results can be easily verified after they are generated. We can automatically

Table 1: *Insights extracted by the Aggregator agent. The table shows all flag-related insights that were discovered as well as a representative sample of other, non-flag-related, insights. Each insight is accompanied by a description of the aggregated view in which it was found, the numerical value of the quantity in question, a generated explanation of its relevance, and how the data was aggregated in terms of which column it was grouped by and the column it was aggregated over.*

Insight	Aggregation	Value	Explanation
⚑ Flag 1			
Arizona has an extremely low Operating Margin	Grouped by: **State** on **Operating Margin**	0.1 %	This insight is captivating because it demonstrates an extremely low operating margin, indicating very low profitability for this sale.
⚑ Flag 2			
Alaska has the highest sales.	Grouped by: **State** on **Total Sales**	$ 49 473 404	Despite its small population, Alaska has the highest sales, which is surprising considering the larger populations and economies of states like California and New York.
⚑ Flag 3			
Kohl's has unusually high units sold.	None	8 000 000	This is the highest number of units sold, which is surprising given the average units sold in other rows.
Other			
West Gear, a lesser-known retailer, has high sales.	Grouped by: **Retailer** on **Units Sold**	715 900	It's surprising that a lesser-known retailer like West Gear has such high total sales, nearly matching Amazon.
Kohl's has highest Units Sold	None	7 500	This insight is the most interesting as it highlights the highest number of units sold, indicating a very popular product.
Men's Street Footwear has the highest Total Sales	None	732 000 $	This insight is intriguing because it represents the highest total sales value, indicating a very successful sale.
Foot Locker has the highest Operating Margin seen	None	65 %	This insight is interesting as it reveals the highest operating margin in the data, suggesting a very profitable sale.

cross-check the results against our dataset programmatically by asserting that the values at the cited row and columns match. As a result, the insights we get (as shown in Table 1) are always correct in terms of the reported data values. However, some non-numerical insights, like "Kohl's has the highest number of sales" are more difficult to validate and may not hold true in the data. Therefore, we plan to investigate LLM-based methods that can create code that specifically verifies if these claims are true in future work.

Limitations. There are a couple of important limitations to this agent. First, using the window approach with large amounts of data leads to many LLM queries, which can be quite expensive. Hence, we opted for using GPT-3.5 instead of the better, but more expensive, GPT-4. This means that our current agent may not be using this suite of models to its full potential. Second, our approach tends to provide insights that are local in view, making it difficult to see big trends or broader patterns in the data. In other words, it is challenging to get a high-level view with our bottom-up method which can potentially be addressed by the Explorer Agent.

Future Work. While our research has shown promising initial results in extracting useful insights from data, there is room for improvement in achieving deeper insights. We intend to experiment with other language models like GPT-4 [OpenAI, 2023a] and Llama-2 [Touvron et al., 2023], as well as explore different ways to prompt the agent to perform data aggregation. These explorations could lead to methods that are crucial in extracting more interesting insights. We currently prompt the agent to use first- and second-order dataset statistics for aggregating the data, but, in some cases, higher statistics could be necessary to identify interesting insights. Another important aspect is to investigate other

Table 2: *Insights extracted by the Explorer agent. Each insight is accompanied by the question that led to its discovery and a generated explanation of its relevance. Insights corresponding to planted flags are shown first, followed by other non-flag-related insights.*

Question	Insight	Explanation
⚑ Flag 1		
What is the relationship between operating margin and sales method for each retailer in 2021?	Kohl's had a very low average operating margin for online sales (0.001%), but a higher average operating margin for outlet sales (35.63%).	The almost non-existent operating margin for Kohl's online sales, when juxtaposed with its outlet sales, is surprising. Retailers typically achieve significant margins through online sales due to reduced overheads compared to physical stores.
⚑ Flag 2		
What are the top 5 cities in terms of sales revenue for 2021?	Anchorage is the top city in terms of sales revenue for 2021.	Anchorage being at the top in sales revenue is unexpected, given that it's a less populated city when compared to other major U.S. cities. One would expect cities with higher populations and larger economies, like San Francisco or Chicago, to lead in sales revenue.
⚑ Flag 3		
Agent failed to capture the flag		
Other		
What is the average units sold per transaction for each sales method (Outlet, Online, In-store) in 2021?	In-store and Outlet sales methods have a significantly higher average units sold per transaction compared to Online.	In today's digital age, with the rise of e-commerce platforms, one might expect online sales to have higher transactions. However, the data suggests that physical outlets still have robust sales.
Are there any seasonal trends in sales for specific product categories in 2021?	All product categories have the highest sales in January.	While some January sales might be expected due to post-holiday clearances, it's surprising to see all categories peak during this month, suggesting a broad trend or seasonality.
How did the sales of each retailer evolve month by month in 2021?	Sports Direct had no sales data available for any month in 2021.	While not shocking, it is noteworthy that there's no data for a retailer for the entire year. This could indicate a data issue or that the retailer was not operational.
What is the relationship between operating margin and total sales for each product category in 2021?	There is no strong correlation between operating margin and total sales for product categories in 2021.	A business might aim for a balance between sales volume and profit margins. The absence of a strong correlation indicates that some categories might be focusing on volume, while others prioritize profitability.

forms of aggregations like computing correlations between columns instead of grouping them based on different first-level statistics. Further, there are many hyperparameters that need to be explored, like how many aggregations we need, the sliding window size, and the number of insights to be extracted from each portion of the data. We intend to address these limitations in future work; however, our current work serves as a thought-provoking proof of concept, offering valuable ideas and discussion points.

5.3 Explorer Agent Results

Implementation Details. We employed the GPT-3.5 model for generating the outputs for our prompts and we have set the number of rounds of question refinement to be 3, defined as nRounds in Algorithm 1. We defined the high level goal generalGoal to be "I want a general overview of the sales for 2021." and the dataContext to be "This is a dataset of sales transactions" which we both set in the prompt to help in generating the questions and the code (prompt can be found here A.1). We ask the model to generate 10 questions for the data in each round. The dataSchema is the description of each column in the rawData which is a csv file where each row represents a record corresponding to the data.

Relevance. Table 2 presents insights extracted by the Explorer agent, categorized into flagged and non-flagged insights. These insights are highly relevant to the user as they address specific questions, such as the relationship between operating margin and sales method, top-performing cities in terms of sales revenue, and seasonal trends in sales for product categories in 2021. Flagged insights (Flag 1 and Flag 2) highlight unexpected findings, such as Kohl's low operating margin for online sales and Anchorage (Alaska) leading to high sales revenue despite its smaller population. The non-flagged insights also provide valuable information, such as the higher average units sold per transaction in physical stores and outlets compared to online sales and the lack of sales data for Sports Direct throughout 2021. Additionally, the absence of a strong correlation between operating margin and total sales for product categories suggests varying business strategies among retailers. These insights collectively offer valuable information for informed decision-making and analysis.

Consistency. All values extracted by the Explorer Agent in Table 2 have been derived directly from the output of the code generated to address each specific question, ensuring their factual basis. However, it is important to note that, similar to the Aggregator Agent, the justifications themselves may require further verification by a critic (which could be human or an efficient LLM method). These justifications are judgments made by the model and should be subject to scrutiny and validation to ensure their accuracy and reliability.

Limitations. The Explorer Agent has certain limitations in its approach. It begins by asking high-level questions to identify patterns in the data, which may inadvertently overlook crucial flags that would be interesting to the data analyst. For example, Flag 3, which consists of an event that happened in a very short period, is overlooked by this agent. This likely occurs due to its tendency to ask high-level questions, rather than digging into low-level details. Furthermore, it does not consider the local perspectives of the data, potentially missing deviations from expected norms, an aspect that the Aggregator Agent could potentially address. Another limitation is that the Explorer Agent might generate code that is not executable (due to bugs for example). Addressing this limitation requires additional OpenAI API calls showing the original prompt along with the execution error in order to re-generate the code with the bugs fixed. However, generating executable code is not guaranteed, forcing the agent to skip the question altogether. These limitations highlight the need for a more comprehensive and context-aware approach to data analysis and code generation.

Future Work. We plan to conduct more experiments related to the Explorer Agent to investigate how it behaves when using different large language models, like GPT 4, StarCoder Li et al. [2023b] and Llama 2 Touvron et al. [2023], and when generating the questions and code based on different prompts. We plan to evaluate how changing hyperparameters like the number of questions, and the number of times we refine the questions impacts the result. We also plan to compare between different context and high level goal that we set in the prompts with respect to the quality of the generated insights.

6 Conclusion

In conclusion, we explore two distinct approaches for automating data analysis and extracting useful insights from a "capture the flag" benchmark that we have introduced: the top-down approach of the Explorer Agent and the holistic, bottom-up, perspective of the Aggregator Agent. The Explorer Agent excels in generating context-aware high-level questions and code, but may overlook specific flags, miss local deviations, and can potentially generate non-functional code. Meanwhile, the Aggregator Agent offers a comprehensive data view, but could potentially struggle with capturing long-term trends, and can be expensive with large datasets. Given these observations, it becomes evident that a hybrid approach, combining the high-level question generation capabilities of the Explorer Agent with the holistic data perspective of the Aggregator Agent, holds great promise for future research and development in automating data analysis effectively. This work represents a preliminary step towards effective data analysis automation using large language models, emphasizing the need for continued research in this avenue.

References

Rajas Agashe, Srini Iyer, and Luke Zettlemoyer. Juice: A large scale distantly supervised dataset for open domain context-based code generation. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2019.

Charu C. Aggarwal, Djallel Bouneffouf, Horst Samulowitz, Beat Buesser, Thanh Hoang, Udayan Khurana, Sijia Liu, Tejaswini Pedapati, Parikshit Ram, Ambrish Rawat, Martin Wistuba, and Alexander G. Gray. How can AI automate end-to-end data science? *ArXiv*, abs/1910.14436, 2019.

Randy Bean. Why becoming a data-driven organization is so hard. *Harvard Business Review*, feb 2022.

Sébastien Bubeck, Varun Chandrasekaran, Ronen Eldan, Johannes Gehrke, Eric Horvitz, Ece Kamar, Peter Lee, Yin Tat Lee, Yuanzhi Li, Scott Lundberg, Harsha Nori, Hamid Palangi, Marco Tulio Ribeiro, and Yi Zhang. Sparks of artificial general intelligence: Early experiments with GPT-4, 2023.

Shubham Chandel, Colin B. Clement, Guillermo Serrato, and Neel Sundaresan. Training and evaluating a Jupyter notebook data science assistant. *ArXiv*, abs/2201.12901, 2022.

Heemali Chaudhari. Adidas sales dataset. `https://www.kaggle.com/datasets/heemalichaudhari/adidas-sales-dataset`, 2023. Accessed: September 29, 2023.

Swarat Chaudhuri, Kevin Ellis, Oleksandr Polozov, Rishabh Singh, Armando Solar-Lezama, and Yisong Yue. Neurosymbolic programming. *Found. Trends Program. Lang.*, 7:158–243, 2021.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde, Jared Kaplan, Harrison Edwards, Yura Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, David W. Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William H. Guss, Alex Nichol, Igor Babuschkin, S. Arun Balaji, Shantanu Jain, Andrew Carr, Jan Leike, Joshua Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew M. Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. Evaluating large language models trained on code. *ArXiv*, abs/2107.03374, 2021a.

Xinyun Chen, Linyuan Gong, Alvin Cheung, and Dawn Song. PlotCoder: Hierarchical decoding for synthesizing visualization code in programmatic context. In *Association for Computational Linguistics (ACL)*, 2021b.

Zhutian Chen, Chenyang Zhang, Qianwen Wang, Jakob Troidl, Simon Warchol, Johanna Beyer, Nils Gehlenborg, and Hanspeter Pfister. Beyond generating code: Evaluating GPT on a data visualization course, 2023.

Liying Cheng, Xingxuan Li, and Lidong Bing. Is GPT-4 a good data analyst? *arXiv*, abs/2305.15038, 2023.

Eric Colson. What ai-driven decision making looks like. *Harvard Business Review*, jul 2019.

Kedar Dhamdhere, Kevin McCurley, Mukund Sundararajan, Qiqi Yan, and Ralfi Nahmias. Analyza: Exploring data with conversation. In *Intelligent User Interfaces*, 2017.

Victor Dibia. LIDA: A tool for automatic generation of grammar-agnostic visualizations and infographics using large language models. In *Association for Computational Linguistics (ACL)*, 2023.

Victor C. Dibia and Çagatay Demiralp. Data2vis: Automatic generation of data visualizations using sequence-to-sequence recurrent neural networks. *IEEE Computer Graphics and Applications*, 39: 33–46, 2018.

Ralph Foorthuis. On the nature and types of anomalies: a review of deviations in data. *International Journal of Data Science and Analytics*, 12, 2020.

Daniel Fried, Armen Aghajanyan, Jessy Lin, Sida I. Wang, Eric Wallace, Freda Shi, Ruiqi Zhong, Wen tau Yih, Luke Zettlemoyer, and Mike Lewis. Incoder: A generative model for code infilling and synthesis. *ArXiv*, abs/2204.05999, 2022.

Significant Gravitas. AutoGPT, 2023. URL `https://github.com/Significant-Gravitas/Auto-GPT`.

Ken Gu, Ruoxi Shang, Tim Althoff, Chenglong Wang, and Steven M. Drucker. How do analysts understand and verify ai-assisted data analyses? *arXiv*, abs/2309.10947, 2023.

Dan Hendrycks and Kevin Gimpel. A baseline for detecting misclassified and out-of-distribution examples in neural networks. In *International Conference on Learning Representations*, 2017.

Junjie Huang, Chenglong Wang, Jipeng Zhang, Cong Yan, Haotian Cui, Jeevana Priya Inala, Colin Clement, and Nan Duan. Execution-based evaluation for data science code generation models. In *ACL Workshop on Data Science with Human-in-the-Loop (Language Advances)*, 2022.

Mohammad Abdullah Matin Khan, M Saiful Bari, Xuan Long Do, Weishi Wang, Md. Rizwan Parvez, and Shafiq R. Joty. xCodeEval: A large scale multilingual multitask benchmark for code understanding, generation, translation and retrieval. *arXiv*, abs/2303.03004, 2023.

Atsutoshi Kumagai, Tomoharu Iwata, and Yasuhiro Fujiwara. Transfer anomaly detection by inferring latent domain representations. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*. Curran Associates, Inc., 2019.

Yuhang Lai, Chengxi Li, Yiming Wang, Tianyi Zhang, Ruiqi Zhong, Luke Zettlemoyer, Scott Wen tau Yih, Daniel Fried, Sida Wang, and Tao Yu. DS-1000: A natural and reliable benchmark for data science code generation. *ArXiv*, abs/2211.11501, 2022.

Hongxin Li, Jingran Su, Yuntao Chen, Qing Li, and Zhaoxiang Zhang. SheetCopilot: Bringing software productivity to the next level through large language models. *arXiv*, abs/2305.19308, 2023a.

Raymond Li, Loubna Ben Allal, Yangtian Zi, Niklas Muennighoff, Denis Kocetkov, Chenghao Mou, Marc Marone, Christopher Akiki, Jia Li, Jenny Chim, et al. Starcoder: may the source be with you! *arXiv preprint arXiv:2305.06161*, 2023b.

Yujia Li, David H. Choi, Junyoung Chung, Nate Kushman, Julian Schrittwieser, Rémi Leblond, Tom, Eccles, James Keeling, Felix Gimeno, Agustin Dal Lago, Thomas Hubert, Peter Choy, Cyprien de, Masson d'Autume, Igor Babuschkin, Xinyun Chen, Po-Sen Huang, Johannes Welbl, Sven Gowal, Alexey, Cherepanov, James Molloy, Daniel Jaymin Mankowitz, Esme Sutherland Robson, Pushmeet Kohli, Nando de, Freitas, Koray Kavukcuoglu, and Oriol Vinyals. Competition-level code generation with AlphaCode. *Science*, 378:1092 – 1097, 2022.

Pingchuan Ma, Rui Ding, Shuai Wang, Shi Han, and Dongmei Zhang. Demonstration of InsightPilot: An LLM-empowered automated data exploration system. *CoRR*, abs/2304.00477, 2023.

Paula Maddigan and Teo Susnjak. Chat2vis: Fine-tuning data visualisations using multilingual natural language text and pre-trained large language models. *arXiv*, abs/2303.14292, 2023.

Andrew McAfee and Erik Brynjolfsson. Big data: The management revolution. *Harvard Business Review*, oct 2012. From the Magazine (October 2012).

Venkata Vamsikrishna Meduri, Abdul Quamar, Chuan Lei, Vasilis Efthymiou, and Fatma Ozcan. BI-REC: Guided data analysis for conversational business intelligence. *arXiv*, abs/2105.00467, 2021.

Theo X. Olausson, Jeevana Priya Inala, Chenglong Wang, Jianfeng Gao, and Armando Solar-Lezama. Demystifying GPT self-repair for code generation, 2023.

OpenAI. GPT-4 technical report. *ArXiv*, abs/2303.08774, 2023a.

OpenAI. ChatGPT. https://chat.openai.com/chat, 2023b.

Luc De Raedt, Hendrik Blockeel, Samuel Kolb, Stefano Teso, and Gust Verbruggen. Elements of an automatic data scientist. In *International Symposium on Intelligent Data Analysis*, 2018.

Lukas Ruff, Jacob R. Kauffmann, Robert A. Vandermeulen, Grégoire Montavon, Wojciech Samek, Marius Kloft, Thomas G. Dietterich, and Klaus-Robert Müller. A unifying review of deep and shallow anomaly detection. *Proc. IEEE*, 2021.

Mohammadreza Salehi, Hossein Mirzaei, Dan Hendrycks, Yixuan Li, Mohammad Hossein Rohban, and Mohammad Sabokrou. A unified survey on anomaly, novelty, open-set, and out of-distribution detection: Solutions and future challenges. *Transactions on Machine Learning Research*, 2022. ISSN 2835-8856.

Lifeng Shen, Zhuocong Li, and James Kwok. Timeseries anomaly detection using temporal hierarchical one-class network. In *Advances in Neural Information Processing Systems*, 2020.

Yongliang Shen, Kaitao Song, Xu Tan, Dongsheng Li, Weiming Lu, and Yueting Zhuang. HuggingGPT: Solving AI tasks with ChatGPT and its friends in Hugging Face. *arXiv*, abs/2303.17580, 2023.

Noah Shinn, Federico Cassano, Beck Labash, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning, 2023.

Christian Steinruecken, Emma Smith, David Janz, James Lloyd, and Zoubin Ghahramani. *The Automatic Statistician*, page 161–173. Springer International Publishing, 2019.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.

Lei Wang, Wanyu Xu, Yihuai Lan, Zhiqiang Hu, Yunshi Lan, Roy Ka-Wei Lee, and Ee-Peng Lim. Plan-and-solve prompting: Improving zero-shot chain-of-thought reasoning by large language models. *arXiv*, abs/2305.04091, 2023.

Zhiruo Wang, Shuyan Zhou, Daniel Fried, and Graham Neubig. Execution-based evaluation for open-domain code generation. *ArXiv*, abs/2212.10481, 2022.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, brian ichter, Fei Xia, Ed H. Chi, Quoc V Le, and Denny Zhou. Chain of thought prompting elicits reasoning in large language models. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho, editors, *Advances in Neural Information Processing Systems*, 2022.

Jingkang Yang, Pengyun Wang, Dejian Zou, Zitang Zhou, Kunyuan Ding, WENXUAN PENG, Haoqi Wang, Guangyao Chen, Bo Li, Yiyou Sun, Xuefeng Du, Kaiyang Zhou, Wayne Zhang, Dan Hendrycks, Yixuan Li, and Ziwei Liu. OpenOOD: Benchmarking generalized out-of-distribution detection. In *Thirty-sixth Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2022. URL `https://openreview.net/forum?id=gT6j4_tskUt`.

Pengcheng Yin, Wen-Ding Li, Kefan Xiao, Abhishek Rao, Yeming Wen, Kensen Shi, Joshua Howland, Paige Bailey, Michele Catasta, Henryk Michalewski, Oleksandr Polozov, and Charles Sutton. Natural language to code generation in interactive data science notebooks. In *Association for Computational Linguistics (ACL)*, 2023.

Tianyi Zhang, Zhiyang Chen, Yuanli Zhu, Priyan Vaithilingam, Xinyu Wang, and Elena L. Glassman. Interpretable program synthesis. *CHI Conference on Human Factors in Computing Systems*, 2021.

A Supplementary Materials

A.1 Explorer Agent

A.1.1 Generating the Questions

```
 Hi, I require the services of your team to help me reach my goal.

        <context>{dataContext}</context>

        <goal>{generalGoal}</goal>

        <schema>{dataSchema}</schema>

        <insights>{insights}</insights>

        Instructions:
        * Produce a list of questions to be solved by the data scientists in your
        team to explore my data and reach my goal.
        * Explore diverse aspects of the data, and ask questions that are relevant to
        my goal.
        * You must ask the right questions to surface anything interesting (trends,
        anomalies, etc.)
        * Make sure these can realistically be answered based on the data schema.
        * The insights that your team will extract will be used to generate a report.
        * Each question that you produce must be enclosed in <question></question> tags.
        * Do not number the questions.
        * You can produce at most {max_questions} questions.
```

A.1.2 Ranking the Insights Prompt

```
Rank the answers and justification from the sales csv below based on the order of
how surprising each is. Start with the most surprising. Explain why these insights
deviate from what is expected.

Write the row number, the insight, the values, and an explanation

Put it in this format.

Row:
Insight:
Explanation:

{insights}
```

where insightList looks like the raw CSV version of Table 2.

A.2 Aggregator Agent

A.2.1 Data Aggregation Prompt

Below is the prompt used for identifying different aggregations of the data where the values under "CSV Columns" and "Stats" are data dependent.

```
{generalGoal}
Below is an instruction that describes a task. Write a response that appropriately
completes the request.

### Instruction:

Given these csv columns and their stats, what are 20 useful aggregations to the data
that groups on one column and aggregates values on another column?
Write them in this format
```

```
Groupby:
Target column:
Aggregation function:


CSV Columns:
=======
{dataColumns}


Stats
=====
{dataStats}
```

where for the sales dataset, {generalGoal} is

```
You are a sales expert analyst who is interested in understanding the operations of
the store sales across the USA.
```

{dataColumns} is

```
Retailer,Retailer ID,Invoice Date,Region,State,City,Product,Price per Unit,
Units Sold,Total Sales,Operating Profit,Operating Margin,Sales Method
```

{dataStats} is

```
Retailer ID,Price per Unit,Units Sold,Total Sales,Operating Profit,Operating Margin
count,1000.0,1000.0,1000.0,1000.0,1000.0,1000.0
mean,1163877.567,53.096,13604.612,654966.201,85163.373,35.83
std,29580.41380228183,12.881331946375449,284477.0482445489,12800683.818093507,\
71638.05466051835,9.748237180543368
min,1128299.0,20.0,7.0,224.0,112.0,10.0
25%,1128299.0,45.0,2000.0,90000.0,29718.75,30.0
50%,1185732.0,50.0,4500.0,236250.0,75000.0,35.0
75%,1185732.0,60.0,7250.0,382500.0,126000.0,42.25
max,1197831.0,110.0,9000000.0,405000000.0,390000.0,76.0
```

A.2.2 Extracting Insights Prompt

Below is the prompt used for extracting insights from a chunk of the data represented under "CSV Data."

```
{generalGoal}
Below is an instruction that describes a task. Write a response that appropriately
completes the request.

### Instruction:

Find 5 surprising, interesting insights from the csv below in 8 words max in bullet
points. For each cite the row number, explain why, provide the relevant value as
(column, value), and give a score 1-5 about how surprising it is and why did you
give it that score.

        Row:
        Insight:
        Values:
        Score:
        Explanation:

CSV Data
=======
{aggregatedDataWindow}
```

```
### Response:
```

where an example aggregated data is {aggregatedDataWindow} is

```
,Retailer,Total Sales (sum)
0,Amazon,13158552
1,Foot Locker,64051537
2,Kohl's,417223750
3,Sports Direct,22582500
4,Walmart,38552250
5,West Gear,99397612
```

A.2.3   Ranking the Insights Prompt

```
Rank the insights from the csv below based on order of how interesting each
is. Start with the most interesting.

Write the row number, the insight, the values,  and an explanation

Put it in this format.

Row:
Insight:
Explanation:

{CSV Content}
```

where CSV Content looks like the raw CSV version of Table 3.

Table 3: *Insights extracted by the Aggregator Agent*

Insight	Values	Score	Explanation
Amazon's total sales are surprisingly low.	(Retailer, Amazon), (Total Sales (sum), 45,020,834)	4	Given Amazon's dominance in the retail market, one would expect their total sales to be higher than other retailers.
Foot Locker has the highest total sales.	(Retailer, Foot Locker), (Total Sales (sum), 49,888,450)	5	It's surprising that a specialized retailer like Foot Locker has the highest total sales, surpassing even Amazon.
Kohl's total sales are significantly lower than others.	(Retailer, Kohl's), (Total Sales (sum), 11,888,750)	3	Considering Kohl's is a well-known retailer, it's surprising their total sales are so much lower than other retailers.