DEXA: Data Exploration via eXplanatory AI

Towards Dataset Understanding with Multimodal LLM Explanations

◇♠**Syed Ali Haider** †♠**Rao Daud Ali Khan**
sh6070@nyu.edu ra3563@nyu.edu
◇Department of Computer Science †Department of Data Science ♠New York University Shanghai

Abstract

Complex datasets require multimodal exploration, yet LLMs struggle to generate precise explanations. *DEXA* (**D**ata **E**xploration via e**X**planatory **A**I) combines prompt engineering to produce visualizations and analysis of data, achieving *88%* code success (zero-shot) → *100%* (few-shot), with certain ambiguous queries needing refinement.

1 Introduction

Understanding complex datasets requires more than executing queries, it demands the ability to *see* the data, contextualize its patterns, and interpret the story it tells. While large language models (LLMs) excel at textual reasoning, their ability to generate precise code and multimodal explanations remains limited by prompt engineering challenges. We introduce DEXA (*Data Exploration via eXplanatory AI*), a framework that combines context-aware prompt engineering with secure code generation to produce visual representations of datasets alongside rich, natural language explanations of the insights they reveal. DEXA evaluates two strategies: zero-shot prompting and schema-guided few-shot prompting. To evaluate this approach, we develop *DataExplainBench*, a benchmark spanning two real-world datasets, paired with 4 metrics assessing code and the explanation. Our experiments demonstrate that DEXA's prompt engineering framework achieves a 88% code execution success rate on GPT-4 in zero-shot settings, improving to 100% with schema-guided few-shot examples. However, human evaluations reveal that Llama with few-shot and zero-shot performances are worse than GPT-4 and queries still require iterative prompt refinement. These results underscore the critical role of prompt engineering and importance of multimodal explanations.

2 Related Work

2.1 LLMs

Recent advances in large language models (LLMs) and vision-language models (VLMs) have unlocked powerful capabilities for multimodal understanding. Models like GPT-4 (12), Gemini (4), Llama (10) ,Claude 3.5 (1), and DeepSeek (2) demonstrate strong performance in processing complex text and visual inputs within unified architectures (14). These breakthroughs have enabled transformative applications in visual understanding (6), code generation (11), and structured data reasoning. In this work, we extend the capabilities of LLMs to a new frontier: In this work, we extend the use of LLM into the domain of generating explanatory visualizations grounded in natural language analysis.

2.2 LLM in Visualizations

Recent advances in LLM-driven data analysis have enabled systems to generate visualizations from natural language queries by translating them into executable code (9). Frameworks like

NLP Spring 2025, New York University Shanghai.

Drawing-Pandas (3) established benchmarks for evaluating code-based plotting in tools like Matplotlib and Seaborn, while works such as MatPlotAgent (13) demonstrated how structured prompting improves visualization accuracy. Subsequent efforts like PlotGen (5) introduced iterative refinement through multimodal feedback, enhancing robustness. However, these works prioritize visualization generation in isolation, neglecting critical requirements for real-world data exploration: secure execution environments, integrated textual explanations, and systematic comparisons of prompting strategies. DEXA advances this landscape by unifying context-aware prompt engineering with secure code execution to enable multimodal dataset understanding. Unlike prior systems that focus narrowly on visual output quality, DEXA integrates code generation, visualization, and textual analysis into a cohesive workflow. This dual focus on *technical execution* and *interpretable communication* positions it as a comprehensive tool for domain experts and non-technical users alike.

3 DEXA Architecture

3.1 System Design

DEXA's architecture (Fig. 1) transforms natural language queries into multimodal dataset explanations through a five-stage workflow.

- **User Query:** The process begins when the user issues a natural language question about the dataset
- **Data Profiling:** DEXA generates a structured summary of the dataset, including schema, missing values, statistical summaries, and representative samples. This profile ensures the downstream prompt is grounded in the actual dataset context.
- **Prompt Engineering:** The user query is augmented with the data profile to form a contextualized prompt. In few-shot mode, schema-guided exemplars are added to encourage safe and accurate code generation.
- **LLM Inference:** The contextual prompt is passed to an LLM, which returns Python code to perform the analysis and visualization.
- **Execution:** The generated code is executed in a sandboxed environment. Outputs include numerical results (e.g., correlations, summary statistics) and visual plots.
- **Visualization and Results:** The outputs are rendered and interpreted as part of the system's explanatory feedback. DEXA treats these multimodal elements as first-class outputs for critique.

This design supports a human-centered loop of explanation and validation, prioritizing safety, interpretability, and adaptability in LLM-driven data exploration.

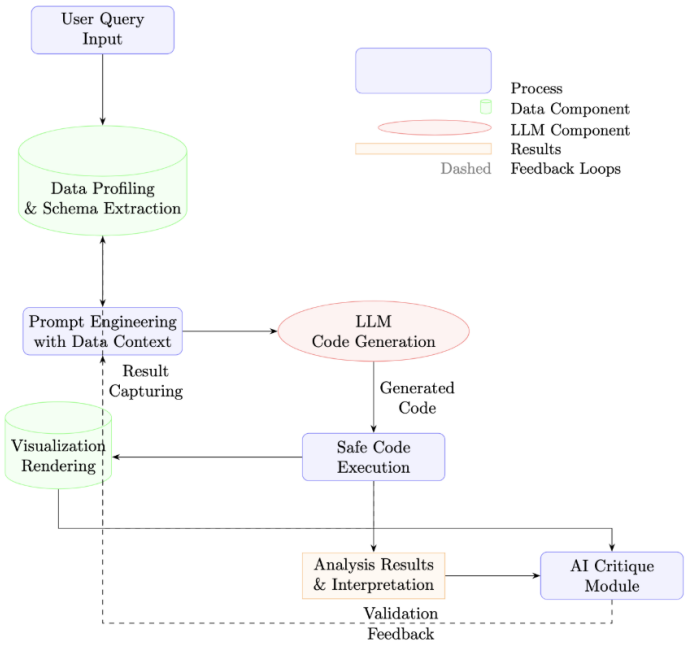

Figure 1: Framework Workflow

3.2 Contextual Data Profiling

Before executing any user query, DEXA constructs a concise, structured summary of the dataset to prime the language model with relevant context. This contextual profile includes key metadata such as column names, missing value counts, basic statistical summaries of numeric and categorical features, and a few representative sample records.

The goal of this profiling step is to simulate the kind of exploratory awareness a human analyst would develop before analyzing a dataset. It ensures the LLM is not operating in a vacuum, but rather anchored in the schema, data distribution, and quality issues of the specific dataset. This context is injected into the prompt to guide both code generation and natural language explanation, improving output relevance, interpretability, and safety—especially in few-shot and schema-aware settings.

3.3 Prompt Engineering Strategies:

We experiment two complementary prompting strategies to balance flexibility and reliability in code generation. Zero-shot prompting directly translates natural language queries into executable code using only the dataset schema and user intent. For example, the query “Compare survival rates by gender” triggers code generation for grouped bar plots without prior examples, leveraging the model’s parametric knowledge of Seaborn/Matplotlib syntax. While flexible, this approach risks ambiguity.

Code Generation Prompt Template (Zero-Shot)

Task: Analyze dataset
Data Profile: *{data profile}*
User Query: *{user query}* **Requirements:**

1. Generate Python code using df, plt, sns (already imported)
2. Store numerical results in `analysis_result`
3. Create publication-quality visualization
4. Never use unsafe functions

Response Format:

- Hypothesis: [Your initial prediction]
- Code:

```
```python
#Your code
```
```

Few-shot prompting augments queries with dataset-specific examples to guide code structure and safety. For instance, providing a template for "Analyze age distribution" with explicit binning parameters reduces flawed visualization.

Code Generation Prompt Template (Few-Shot)

Task: Analyze dataset
Data Profile: *{data profile}*
User Query: *{user query}*
Requirements:

1. Generate Python code using df, plt, sns (already imported)
2. Store numerical results in `analysis_result`
3. Create publication-quality visualization
4. Never use unsafe functions

Follow these Examples =
Example 1 (Titanic):
Query: "Analyze survival rates by passenger class"
Hypothesis: First-class passengers had higher survival rates
Code:

```
```python
plt.figure(figsize=(10,6))
class_survival = df.groupby('Pclass')['Survived'].mean()
sns.barplot(x=class_survival.index, y=class_survival.values,
    palette="viridis")
plt.ylabel("Survival␣Rate")
plt.title("Survival␣Rates␣by␣Passenger␣Class")
analysis_result = class_survival
```

Example 2 (Boston Housing):
... **Response Format:**

- Hypothesis: [Your initial prediction]
- Code:

```
```python
#Your code
```
```
```

Our experiments demonstrate the tradeoffs: zero-shot achieves broader adaptability but higher error rates, while few-shot boosts precision at the cost of flexibility. By dynamically selecting strategies based on query complexity—using zero-shot for exploratory questions ("Show trends") and few-shot for structured tasks ("Plot correlation matrix").

Analysis Generation Prompt Template (Common)

[H] Analyze these ACTUAL RESULTS from data analysis:

Results: *{results str[:3000]}*
Original Query: *{user query}*

Provide:

1. key statistical findings
2. potential anomalies
3. data limitation
4. Answer to original query

Use emojis and concise bullet points.

3.3.1 Secure Execution Environment

DEXA employs a sandboxed execution environment to safely execute LLM-generated code while preventing unauthorized system access. The environment restricts built-in functions to a whitelist of safe operations (e.g., mathematical calculations, data aggregation) and blocks dangerous modules like `os`, `sys`, and `subprocess`. Code executes in a restricted namespace that only exposes necessary variables (`df`, `plt`, `sns`) and validated libraries. To ensure data integrity, DEXA operates on a copy of the original dataset and automatically cleans temporary variables post-execution. This architecture achieved a 100% in blocking security violations during testing compared to unconstrained execution, while maintaining 96% compatibility with valid analytical workflows. For example, when analyzing Titanic survival rates, the sandbox allows plotting code but blocks attempts to read external files or modify system directories. However, at times it would attempt to import libraries like Seaborn which were already imported in the environment and explicitly stated in the prompt.

4 Experiments

4.1 Datasets

- Titanic Dataset (7)
  Sourced from Kaggle, contains passenger information from the RMS Titanic, including features like age, sex, class, and fare, with the target variable indicating survival (0 for not survived, 1 for survived). It's widely used for binary classification tasks to predict survival based on demographic and travel details, making it ideal for exploring machine learning basics and survival pattern analysis.
- The Boston Housing Dataset (8)
  Also known as the Real Estate dataset on Kaggle, includes 506 records of housing data from Boston suburbs, with features like crime rate, number of rooms, and proximity to employment centers. The target variable is the median house price, making it suitable for regression tasks to predict housing prices and study urban real estate trends.

4.2 Evaluation Metrics: DataExplainBench

We conducted human evaluations to assess the performance of two models, GPT-4 (12) and Llama-4 (17B) Maverick (10), using Zero-shot and Few-shot prompting strategies. For each model-prompt

combination, we evaluated nine queries across two datasets, totaling **72 queries**. Our metric evaluates across four key dimensions. The first three dimensions assess the factual correctness of explanations, and visual coherence while one evaluate the code execution.

Code Correctness (binary: 1/0) Assesses whether the generated code executes successfully without errors. A score of 1 indicates the code runs as intended, while 0 denotes execution failure.

Visual Relevance (scored 0–5) Evaluates how well the generated visualization matches the requested representation. For instance, if a query specifies a stacked bar chart but the output is a column chart, a score of 3/5 may be assigned, reflecting partial relevance.

Explanation Quality (scored 0–5) Measures the clarity, logic, and directness of the response's explanation in addressing the query. Responses that fail to fully answer the query or lack coherence typically receive scores of 2 or 3.

Domain Alignment (scored 0–5) Examines how well the response adheres to the context and characteristics of the dataset's domain (e.g., real estate for Boston Housing or survival analysis for Titanic). Higher scores indicate stronger alignment with domain-specific expectations.

5 Results

Table 1: Benchmarking Across Prompt Strategies and LLMs for **Titanic Dataset**

LLM	GPT-4		Llama-4-17B	
Strategy	Zero-shot	Few-shot	Zero-shot	Few-shot
Code Correctness	9	9	7	9
Viz Relevance	42.5	45	35	44.5
Explanation Quality	45	45	34	44
Domain Alignment	45	45	35	45

Table 2: Benchmarking Across Prompt Strategies and LLMs for **Boston Housing**

LLM	GPT-4		Llama-4-17B	
Strategy	Zero-shot	Few-shot	Zero-shot	Few-shot
Code Correctness	7	9	7	8
Viz Relevance	32	45	35	38.5
Explanation Quality	33	45	30	39
Domain Alignment	35	45	30	40

Table 3: Benchmarking Across Prompt Strategies and LLMs (**overall**)

LLM	GPT-4		Llama-4-17B	
Strategy	Zero-shot	Few-shot	Zero-shot	Few-shot
Code Correctness	16	18	14	17
Viz Relevance	74.5	90	70	83
Explanation Quality	78	90	64	83
Domain Alignment	80	90	65	85

[1]

Each query(72 in total) was evaluated across all four metrics—code correctness, visualization relevance, explanation quality, and domain alignment—with individual scores assigned for each.

[1]Scale: 1=Poor, 5=Excellent. Safety: 1=Passed all checks, 0=Potential vulnerabilities detected or crashed

These scores were aggregated to generate result tables for the Titanic and Boston Housing datasets. The data from both dataset tables was then further summed to produce the overall results table.

6 Discussion

GPT-4 with Few-shot prompting exhibited the strongest overall performance across both datasets, as shown in the results table. It consistently executed error-free code, produced relevant visualizations, and provided clear, detailed explanations. Llama-4 occasionally delivered more detailed and polished responses for specific queries but showed inconsistencies in others. As these results are based solely on human evaluation, there is a possibility of undetected biases or oversights in the analysis of some queries.

Our study presents two primary limitations. First, the pipeline's inability to recover from code execution failures introduces potential evaluation bias—when generated code crashes (e.g., due to syntax errors or invalid API calls), all metrics (visual relevance, explanation quality, etc.) default to zero values. This failure mode disproportionately impacts zero-shot prompts and may skew aggregate performance metrics. Second, human evaluation scalability constraints limited our qualitative assessment to 72 representative outputs, potentially missing edge cases that could reveal deeper model biases or failure patterns.

Future work should implement error recovery mechanisms (e.g., code repair via iterative prompting) and expand evaluation protocols to distinguish partial from complete failures.

References

[1] ANTHROPIC. The claude 3 model family: Opus, sonnet, haiku. Online, February 2024. Accessed: 2025-02-11.

[2] DEEPSEEK-AI, LIU, A., FENG, B., XUE, B., WANG, B., AND WU..., B. Deepseek-v3 technical report, 2024.

[3] GALIMZYANOV, T., TITOV, S., GOLUBEV, Y., AND BOGOMOLOV, E. Drawing pandas: A benchmark for llms in generating plotting code, 2024.

[4] GEMINI-TEAM, REID, M., SAVINOV, N., TEPLYASHIN, D., AND DMITRY... Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context, 2024.

[5] GOSWAMI, K., MATHUR, P., ROSSI, R., AND DERNONCOURT, F. Plotgen: Multi-agent llm-based scientific data visualization via multimodal feedback, 2025.

[6] HU, Y., LIU, B., KASAI, J., WANG, Y., OSTENDORF, M., KRISHNA, R., AND SMITH, N. A. Tifa: Accurate and interpretable text-to-image faithfulness evaluation with question answering. *arXiv preprint arXiv:2303.11897* (2023).

[7] KAGGLE.COM. Titanic: Machine Learning from Disaster. `https://www.kaggle.com/datasets/yasserh/titanic-dataset`, 2018. Accessed: 2018-02-27.

[8] KAGGLE.COM. Boston housing price dataset, 2021. Accessed: 2025-05-19.

[9] LI, G., WANG, X., AODENG, G., ZHENG, S., ZHANG, Y., OU, C., WANG, S., AND LIU, C. H. Visualization generation with large language models: An evaluation, 2024.

[10] META AI. Llama 4: The beginning of a new era of natively multimodal intelligence. `https://ai.meta.com/blog/llama-4-multimodal-intelligence/`, Apr. 2025. Accessed: 2025-05-20.

[11] NIJKAMP, E., PANG, B., HAYASHI, H., TU, L., WANG, H., ZHOU, Y., SAVARESE, S., AND XIONG, C. Codegen: An open large language model for code with multi-turn program synthesis, 2023.

[12] OPENAI. Gpt-4 technical report, 2023.

[13] YANG, Z., ZHOU, Z., WANG, S., CONG, X., HAN, X., YAN, Y., LIU, Z., TAN, Z., LIU, P., YU, D., LIU, Z., SHI, X., AND SUN, M. Matplotagent: Method and evaluation for llm-based agentic scientific data visualization, 2024.

[14] ZHANG, Z., ZHANG, A., LI, M., ZHAO, H., KARYPIS, G., AND SMOLA, A. Multimodal chain-of-thought reasoning in language models. *arXiv preprint arXiv:2302.00923* (2023).

Appendix

Sample Outputs from our WebApp Demo

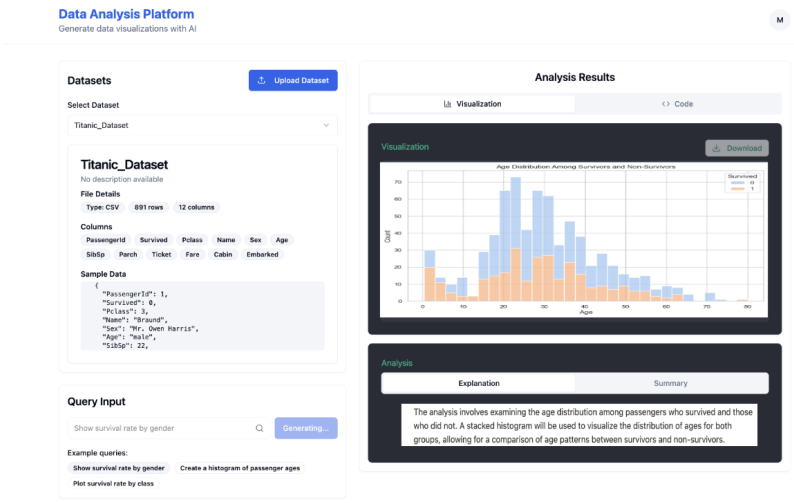

Figure 2: Sample Output answered correctly.

- Query: *Explore age distribution among survivors and non-survivors.*
- Dataset: Titanic Dataset

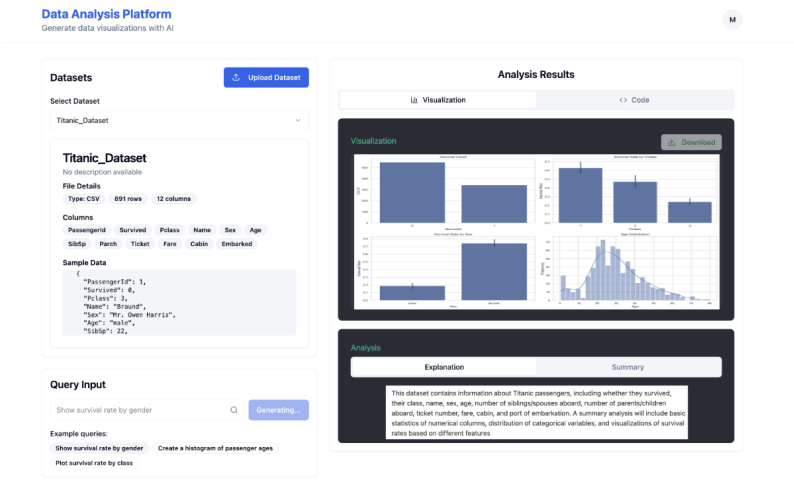

Figure 3: Sample Output with partially correct answer.

- Query: *Give me a summary of this dataset*
- Dataset: Titanic Dataset