SLEUTHBENCH: BENCHMARKING STATISTICAL LLM EVALUATION USING TABULAR HIDDEN SIGNALS

Anonymous authors
Paper under double-blind review

ABSTRACT

Evaluating statistical discovery by large language model (LLM) agents requires verifiable analytical ground truth. Establishing such ground truth for real-world datasets is costly, and prior knowledge of public datasets can influence agent responses. We introduce SLEUTHBENCH, a benchmark that addresses both problems by injecting controlled data-quality problems and feature effects into public tabular datasets: the injected pattern determines the answer, so reference answers are computed automatically and memorized knowledge of the original table is insufficient, while the table keeps its background structure. The injected patterns are modeled on phenomena reported in real data analyses. The benchmark defines 17 question templates in two families: data-quality questions and feature-contribution questions. We evaluate six state-of-the-art LLMs that analyze the data using a Python coding tool, on data-science and business phrasings of 70 validated dataset–template combinations, yielding 1680 graded responses in total. The models detect data-quality problems reliably (83.8% accuracy) but recover feature contributions poorly (41.9%). Finding how features shape the target requires searching over both candidate variables and analytical procedures. To address this issue, we propose the *Empirical Layer*, a set of precomputed statistical artifacts comprising summaries, fitted feature and interaction effects, and dataset descriptions, which exposes candidate patterns for direct inspection. Access to these artifacts raises feature-contribution accuracy from 41.9% to 68.0%.

1 INTRODUCTION

Large language model (LLM) agents increasingly perform data analysis by inspecting tables, writing code, identifying statistical relationships, and explaining their findings. Evaluating these agents requires determining whether their answers are supported by the supplied data. Public datasets complicate this assessment: prior work shows that LLMs can memorize popular tabular datasets and perform better on datasets encountered during pretraining (Bordt et al., 2024). Consequently, accuracy on a static public dataset can reflect both prior dataset knowledge and the ability to recover and interpret evidence from the table.

Early table-reasoning benchmarks primarily evaluate retrieval and operations over table values, such as selecting rows, comparing or aggregating entries, and verifying whether a statement is supported by a table (Pasupat & Liang, 2015; Chen et al., 2020). More recent benchmarks broaden this scope to include statistical and data-analysis questions (Wu et al., 2024). However, they do not systematically test whether an agent can recover a controlled statistical phenomenon distributed across observations. SLEUTHBENCH targets this distinction by evaluating patterns—such as nonlinear effects, feature interactions, and data-quality artifacts—that are implicit in the empirical distribution rather than stated in any individual cell.

Establishing reliable reference answers for statistical questions over real tables can also require substantial expert effort, making such benchmarks costly to build and refresh. SLEUTHBENCH addresses this challenge by introducing a prescribed statistical phenomenon into a public table and generating a corresponding question. The intervention can introduce a single-feature effect, a pairwise interaction, or a data-quality anomaly while retaining the surrounding table structure. A task-specific validator checks that the intended phenomenon is detectable in the modified table and that competing patterns do not make the answer ambiguous.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

SLEUTHBENCH addresses public-data memorization and costly ground-truth curation by injecting controlled statistical phenomena into source tables and automatically computing reference answers for validated instances. These interventions make knowledge of the original table insufficient to determine the answer, while automatic answer computations avoid the need to manually label each instance. The source supplies correlations, mixed data types, and competing patterns; the intervention specifies the target phenomenon and its acceptance criterion. This creates a middle ground between manually labeling observational data and synthesizing an entire table: the benchmark retains a source table's background structure while controlling the answer-defining evidence. Regenerating interventions with new random seeds reduces dependence on a fixed set of public answers that future LLM could train on.

These choices allow us to evaluate whether agents can identify statistical patterns across observations while keeping reference answers verifiable and evaluation instances refreshable. Our contributions are:

- We introduce a benchmark construction framework that injects statistical phenomena into public tables, validates their observable effects, and computes reference answers independently of the evaluated LLM and its analysis tools. Because the injected phenomenon determines the answer, memorized knowledge of the original table is insufficient, while reusable validators and answer computations reduce per-instance curation and support refreshed cases.
- We provide 17 question templates backed by 14 injectors, covering data-quality problems and feature contributions (single-feature effects and pairwise interactions). Each template is a reusable task specification that defines compatible variables, an injection, alternate question framings, an answer format, and validation criteria.
- We evaluate six LLMs using Python alone and Python augmented with the Empirical Layer, which consists of structured statistical evidence about the data. This paired comparison tests whether making candidate evidence easier to inspect changes statistical-discovery performance, informing how analysis interfaces should present candidate patterns. With Python alone, the models handle data-quality questions well but recover feature contributions poorly; the Empirical Layer raises feature-contribution accuracy from 41.9% to 68.0% and overall accuracy from 64.6% to 76.7%.

2 RELATED WORK

Our work has two components: constructing a benchmark through controlled statistical injections into public source tables, and evaluating how access to structured statistical evidence affects data analysis agents. We review related agent systems and benchmarks, followed by exploratory data analysis and interpretable models that inform the artifacts in our Empirical Layer.

Data analysis agents. LLM-based data analysis agents combine natural-language interaction with planning, code execution, and iterative feedback to carry out analytical workflows. TaskWeaver supports stateful execution and integrates generated code with domain-specific plugins (Qiao et al., 2023). Data Interpreter represents task dependencies through hierarchical graphs and progressively verifies and refines intermediate steps (Hong et al., 2025). LAMBDA separates code generation and debugging into programmer and inspector agents, while supporting user intervention and external analytical methods (Sun et al., 2025). SLEUTHBENCH provides a complementary evaluation setting for such agents: it specifies the target phenomenon and reference answer while allowing the agent to choose its analysis procedure.

Data science benchmarks. Existing benchmarks assess different stages of analytical work. WikiTableQuestions and TabFact evaluate compositional question answering and fact verification over tables (Pasupat & Liang, 2015; Chen et al., 2020), while DS-1000 evaluates data science code using functional tests and implementation constraints (Lai et al., 2023). InfiAgent-DABench and DA-Code evaluate analysis through interaction with execution environments (Hu et al., 2024; Huang et al., 2024); DSBench extends coverage to realistic analysis and modeling workflows (Jing et al., 2025). QRData focuses on statistical and causal reasoning (Liu et al., 2024b), and DiscoveryBench evaluates data-driven discovery using both research-derived and synthetic tasks (Majumder et al.,

2025). DataSciBench addresses complex evaluation criteria through semi-automated ground-truth construction with human verification and programmatic evaluation (Zhang et al., 2026). Benchmark construction also addresses memorization: DS-1000 perturbs source problems, while LiveBench refreshes questions from recently released sources (Lai et al., 2023; White et al., 2025). SLEUTHBENCH complements these benchmarks through its construction approach: it injects controlled statistical phenomena into public source tables, validates the resulting evidence, and computes reference answers deterministically.

Exploratory data analysis and interpretable models. Exploratory data analysis uses statistical summaries and graphical representations to examine distributions, relationships, and unusual observations, guiding subsequent analysis (Tukey, 1977). Visualization recommendation systems assist the search for informative views. Voyager 2 combines manual and automatic chart specification to support both broad exploration and focused questions (Wongsuphasawat et al., 2017). Foresight ranks candidate insights using statistical properties such as correlation, skewness, and outliers (Demiralp et al., 2017). Interpretable models provide a complementary approach: generalized additive models with pairwise interactions, including Explainable Boosting Machines, decompose predictions into inspectable feature effects and interaction surfaces (Lou et al., 2013; Nori et al., 2019). Such models have also been used to investigate missingness and imputation behavior (Chen et al., 2023). We combine statistical summaries and interpretable-model outputs into the Empirical Layer (defined precisely in Section 5), and evaluate how access to these artifacts affects agents' ability to recover the prescribed phenomena.

3 SLEUTHBENCH: A BENCHMARK FOR STATISTICAL DISCOVERY

3.1 BENCHMARK OVERVIEW

SLEUTHBENCH evaluates whether an agent can recover a prescribed statistical phenomenon from a supplied table. Each benchmark instance contains three pieces: an edited table, a natural-language question, and a verifiable reference answer. We construct an instance by combining a public base table with a compatible *question template*. A template is a reusable task specification: it states what statistical pattern to create, which tables and variables are eligible, what question to ask, and what form the answer should take. For example, the inverted-U template asks which numerical feature has a relationship with the target that peaks at an interior value, and where that peak occurs.

Construction proceeds in three steps (Figure 1). First, a randomized injector edits the base table to create the requested pattern and records what it changed. Second, a separate task-specific validator measures the edited table and checks that the pattern satisfies the template's acceptance criteria. Third, a deterministic function computes the reference answer from the accepted table and intervention record. Only candidates that pass validation and receive a computed reference answer enter the benchmark; Section 3.2 gives the technical construction, and the supplementary material describe every template, injection, and question.

Data collection. We assemble public tables from the UCI repository, scikit-learn, and Kaggle, covering maintenance, transportation, housing, and energy use. Our experiments use 1,000-row tables from six base datasets: AI4I 2020 Predictive Maintenance, Bike Sharing, California Housing, King County Housing, Metro Interstate Traffic Volume, and Steel Industry Energy Consumption. Each source snapshot is standardized with a designated target column, row identifiers, and a schema summary, and its rows are shuffled. Template compatibility determines which injections can be attempted, while validation determines which resulting instances enter the benchmark. Full integrity checks, preprocessing steps, and additional prepared table sizes are described in Appendix B.

Phenomena and question framings. The benchmark defines 17 templates implemented by 14 injectors in two template families (Table 1): data-quality questions ask what is wrong with a table, and feature-contribution questions ask how features shape the target. Each template specifies compatible feature types, applicability constraints, a statistical task, and an answer format.

Every accepted instance supports two question framings: a data-science prompt using analytical terminology and a business prompt describing an operational concern. The framings differ in terminology and in how explicitly they specify the intended analysis. Both use the same table, selected

variables, and reference answer, enabling paired comparisons of how question framing affects evidence recovery and interpretation. Appendix A lists the templates and injections.

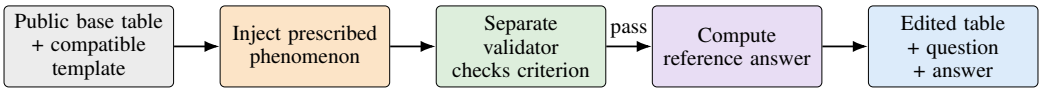


Figure 1: SLEUTHBENCH construction. A separate validator remeasures the injected pattern before a deterministic function computes the reference answer.

Table 1: The 17 benchmark question templates. Appendix A (Table 4) lists every template with its injection and both question wordings.

Family	Templates	Representative injections	Representative tasks
Data quality	8	Corrupt a subset of target values and add a binary column marking those rows; replace some category values with a missing-like label; add a group column with one very rare group.	Identify the marker column or the missing-like label; decide whether there is sufficient statistical evidence to make claims about the rare group.
Feature contribution	9	Permute one numerical feature to break its link to the target; add a smooth inverted-U effect of one feature on the target; add an XOR-shaped effect on the target over two numerical features.	Name the noise feature; name the feature and the value at which its effect peaks; name the interacting pair or its conditional direction.

3.2 PHENOMENON INJECTION AND QUALITY CONTROL

Constrained intervention. Let $\mathcal{D}$ denote a public base table, τ a compatible template, and θ the template's intervention parameters. The randomized injection operator returns

$$(\widetilde{\mathcal{D}}, \mathbf{e}) = I_\tau(\mathcal{D}; \theta), \tag{1}$$

where $\widetilde{\mathcal{D}}$ is the edited table and $\mathbf{e}$ records the selected variables and the exact edit. Before applying the operator, template matching checks feature types, cardinalities, target type, and other task-specific requirements.

Running example: injecting an inverted-U effect

The inverted-U template applies to a table with a numerical feature and target. Its injector adds a target effect centered at an interior feature value and decreasing on both sides.

Validation and quality control. An edit can execute successfully yet remain ambiguous or be overwhelmed by the base table. Each candidate therefore undergoes a phenomenon-specific validation gate $V_\tau(\widetilde{\mathcal{D}}, \mathbf{e})$. The validator measures the resulting pattern directly from the final table and accepts a candidate only when the answer-defining pattern meets the template's operational criterion and, where required, exceeds competing features or pairs. More details are in Appendix C.

Running example: validating an inverted-U effect

After the inverted-U template has been applied, the validator measures the relationship between the selected feature and the target on the final table. It groups the rows into five equal-count bins by feature value and checks three things: the bin with the highest mean target is unique; the injected center falls inside that bin; and the feature shows a clear rise-then-fall pattern.

Independent ground truth. For an accepted candidate, a template-specific function computes the ground truth $a^\star = A_\tau(\widetilde{\mathcal{D}}, \mathbf{e})$. The template determines whether this answer is, for example, a feature name, feature pair, row set, or feature–value pair. Validation determines whether the edited table satisfies the template's acceptance criteria, whereas answer computation supplies the value used for grading.

Running example: computing ground truth for an inverted-U effect

Once the inverted-U injection has passed the validator, the automatic answer computation provides the reference answer in two parts. The feature name is read directly from the edit record. The peak value is computed on the edited table: the function fits two smooth curves to the feature–target relationship, finds the peak on each curve and averages the two locations. The LLM is then asked the following data science question:
"*One feature has an inverted-U relationship with* {`OUTCOME_COL`} *— performance peaks at a specific value and declines on both sides. Which feature, and at what value is the peak? Return your answer as: column_name, value*."

3.3 Injection Design

The design of the injections began with a simple question: what does a data analyst do when examining a new table? An analyst typically first inspects the data itself, checking for outliers, missing values, and other data-quality problems, and then fits a model to examine how features affect the target. These two stages define the tasks an agent should be able to carry out, and they give the benchmark its two families: data-quality questions, which ask what is wrong with a table, and feature-contribution questions, which ask how features shape the target. Each injection recreates the situation that one of these analyses is meant to detect.

A second consideration is memorization. An analyst who receives a table they have analyzed before, or one that is widely known, may skip parts of the analysis and rely on what they remember about the data; LLMs show the same tendency on popular public datasets (Bordt et al., 2024). We therefore inject specific patterns into the features and outcome of each table, so that recovering the injected pattern requires performing the analysis.

In addition, some of our templates draw on findings reported in real data science analyses. For example, in pneumonia data, asthma was associated with lower observed mortality, a pattern that Caruana et al. (2015) attribute to earlier ICU admission and more aggressive treatment. Our categorical-target-outlier injection plants an analogous pattern for a numerical target: one category has an unusually high or low mean outcome. Injections like this plant controlled versions of phenomena observed in real analyses.

3.4 Comparison with Existing Benchmarks

Existing benchmarks enable systematic evaluation of agents on data-analysis workflows and statistical reasoning tasks. However, challenges remain in balancing ground-truth curation costs with data realism, mitigating reliance on memorized public data, and refreshing evaluation instances over time. We discuss how SleuthBench addresses these challenges below.

1. **Ground-truth cost and realism.** Establishing and verifying reliable reference answers for rich analytical tasks over observational data can require substantial expert effort (Liu et al., 2024b; Majumder et al., 2025; Zhang et al., 2026). Fully synthetic data make labels inexpensive once a generator is specified, but their realism depends on that generator. SleuthBench combines public source tables with controlled statistical injections. Once the injection, validation, and answer-computation procedures are defined, they can be reused across compatible tables and seeds, reducing per-instance curation effort while retaining much of the source table's background structure.
2. **Memorization of source data.** Public tables and familiar schemas can be memorized (Bordt et al., 2024). By changing the answer-defining patterns, SleuthBench makes recall of the original table insufficient to determine the answer, although schema knowledge and domain priors may still help.

Table 2: Questions per model, by template family and question framing.

Framing	Data quality	Feature contribution	Total
Data science	38	32	70
Business	38	32	70
Total	76	64	140

3. **Refreshing benchmark instances.** Any published benchmark instance can enter later training corpora. LiveBench and LiveCodeBench address this by periodically writing new questions from newly released material, such as recent papers, news articles, datasets, and programming-contest problems, so that the questions postdate the models' training cutoffs (White et al., 2025; Jain et al., 2024). SLEUTHBENCH can instead simply rerun the pipeline with a new seed and produce fresh instances from the same source tables.

4 SLEUTHBENCH REMAINS CHALLENGING FOR PRETRAINED LLMS

4.1 EXPERIMENTAL SETUP

We first test whether pretrained LLMs can recover the prescribed phenomena using executable analysis alone. Each agent receives the edited table and question and has only a Python tool. This tool executes agent-generated Python code on a persistent pandas dataframe containing the edited table, allowing the agent to inspect the data, perform analyses, and revise its approach using execution outputs and errors. We provide a Python tool rather than placing the table directly in the prompt because LLMs alone are unreliable even at basic structural operations on tables such as cell lookup and row retrieval (Sui et al., 2024; Liu et al., 2024a).

We use the six 1,000-row datasets described in Section 3. All six LLM models are evaluated on the same 70 validated dataset–template combinations, each with a data science and a business question. Table 2 lists the question counts by template family and by framing. All counts and accuracies in this section and the following section refer to this question set. Candidates rejected by the validator during the pipeline construction are excluded from the model evaluation.

Scoring. Responses are graded CORRECT, PARTIAL, or INCORRECT against the reference answer using a template-specific rubric. Numerical and column–value answers use a 5% relative tolerance; structured answers are more exact and must identify the requested columns, category, or row set. The pipeline combines direct checks with LLM-based assessment of answers requiring interpretation (Zheng et al., 2023). Reported accuracy counts only CORRECT responses and is a percentage.

Models. We evaluate six frontier LLMs, Claude Sonnet 5, Claude Opus 4.8, Claude Fable 5, GPT-5.6 Sol, GPT-5.6 Terra, and GPT-5.6 Luna under the setup above.

4.2 RESULTS AND CHALLENGES

Python access alone leaves substantial errors across the six models (Table 3). Accuracy pooled across the data quality and feature contribution injection families is only 64.6% (543 correct answers), with individual scores ranging from 50.7% to 80.0%. On data-quality questions, every model performs well: pooled accuracy is 83.8%, and per-model accuracy ranges from 77.6% to 86.8%. On feature-contribution questions, however, pooled accuracy drops to 41.9%, with per-model accuracy ranging from 18.8% to 71.9%. Recovering how features shape the target, rather than what is wrong with the data table, is the main weakness for Python-only agents.

The large search space of statistical discovery. Statistical discovery requires choosing not only which variables to examine but also which analytical procedure to apply. For example, the same variables can be ranked in importance by many criteria, such as marginal association, predictive contribution, or conditional effect, and each criterion can produce a different ordering. An agent must therefore match its procedure to the question's operational definition. The low feature-contribution

Table 3: Python-only accuracy (%) by model and template family.

Model	Data quality	Feature contribution	Total
Claude Sonnet 5	77.6	18.8	50.7
Claude Opus 4.8	84.2	29.7	59.3
Claude Fable 5	85.5	45.3	67.1
GPT-5.6 Sol	86.8	71.9	80.0
GPT-5.6 Terra	85.5	51.6	70.0
GPT-5.6 Luna	82.9	34.4	60.7
All models	83.8	41.9	64.6

accuracy motivates examining how agents choose candidate relationships and diagnostics, as illustrated by the case study in Appendix F.

Specifying an analysis from business language. Question wording changes the analytical specification available to the agent. Business prompts can require translating an operational concern into a statistical diagnostic, whereas data science prompts often name that diagnostic. Python-only accuracy is 67.1% for data science questions and drops to 62.1% for their business counterparts.

Memorized knowledge versus evidence from the supplied table. Familiar column names and public-dataset regularities can suggest an answer before the agent even examines the edited table, but our interventions on the feature patterns make such priors insufficient for determining the reference answer.

5 LLMs Augmented with Structured Data Analysis Tools

5.1 Empirical Layer

The previous section showed that, with Python alone, agents answer data-quality questions accurately but struggle to recover feature contributions. Motivated by this observation, we develop the Empirical Layer (Figure 2), which provides precomputed statistical summaries and learned representations of the supplied table. For each validated edited table, we compute the same generic representation using only the table and its designated target column. The Empirical Layer comprises column statistics, distributions, correlations, missing-value patterns, and outlier reports. Explainable Boosting Machines (EBMs) additionally provide feature importance, fitted single-feature effects (shape functions), and interaction effects (Nori et al., 2019).

5.2 Experimental Setup

We compare *Python alone* with *Python + Empirical Layer* on the same question set in Section 4. The Empirical Layer tool is added alongside the Python tool; at each step, the agent chooses which tool to call. Throughout, Δ denotes the P+E minus P difference in accuracy, in percentage points. Everything else, including the models, tables, questions, reference answers, and grading, is identical to Section 4, so every P response has a paired P+E response.

5.3 Results

Overall. Adding the Empirical Layer raises pooled accuracy from 64.6% to 76.7%, corresponding to 543 versus 644 correct answers (Figure 3; exact values in Table 6, Appendix E).

The gain comes from feature contributions. The improvement is concentrated in the family where Python-only agents fail the most (Figure 3): Feature-contribution accuracy rises from 41.9% to 68.0% (161 versus 261 correct answers). Data-quality accuracy is unchanged, moving from 83.8% to 84.0% (a net gain of 1 correct answer).

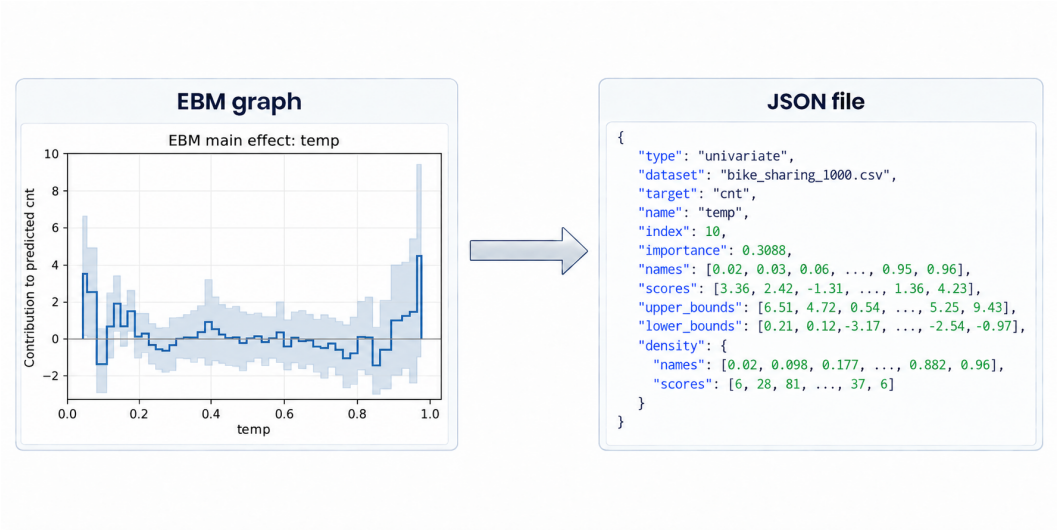


Figure 2: Example of an Empirical Layer artifact from the Bike Sharing dataset. The left panel shows the EBM single-feature effect of `temp` on predicted bike rentals (`cnt`). The right panel is an excerpt from the JSON file storing this artifact.

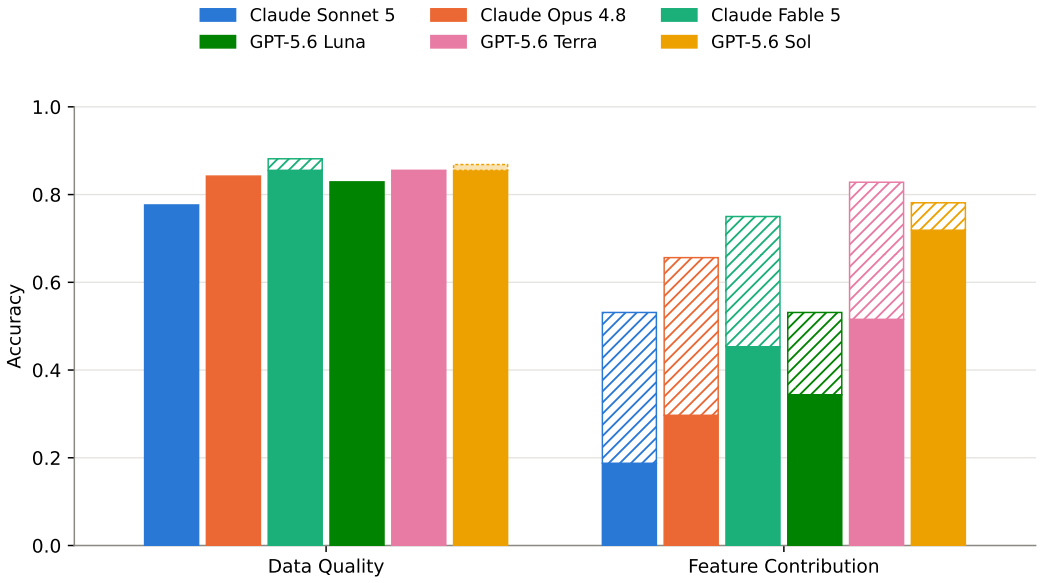


Figure 3: Accuracy by template family and model. Solid bars show accuracy with Python alone; hatched segments show the accuracy of Python + Empirical Layer. Also see Table 6 (Appendix E).

Differences across models. All six models improve, with overall gains ranging from 2.1 to 16.4 percentage points (Table 6, Appendix E). On feature-contribution questions the gains are larger, from 6.3 to 35.9 percentage points, whereas on data-quality questions no model changes by more than 2.6 percentage points. The gain is smallest for GPT-5.6 Sol, the strongest Python-only model, which already reaches 71.9% on feature-contribution questions with Python alone and moves to 78.1%.

Differences across question framings. Business questions gain more from the Empirical Layer than data science questions: 14.8 versus 9.3 percentage points overall (Figure 4; exact values in Table 9, Appendix E). The difference comes entirely from feature-contribution templates, where business questions gain 32.3 percentage points and data science questions 19.8. With Python alone,



Figure 4: Accuracy by template family and question framing, pooled over all six models. Solid bars show accuracy with Python alone; hatched segments show the accuracy of Python + Empirical Layer. Exact values are listed in Tables 9 and 10 (Appendix E).

agents do much worse on the business version of a feature-contribution question than on its data science version (32.8% versus 51.0%), and the Empirical Layer closes most of that gap (65.1% versus 70.8%). A potential reason follows from Section 4: a business prompt requires the agent to translate an operational concern into a statistical diagnostic, so it must search a much larger space of statistical analyses and procedures. The Empirical Layer narrows that search significantly.

6 Conclusion

Real-world data science requires agents not only to write code, but also to discover which relationships in a table support an analytical conclusion. Public-table benchmarks can also reward memorized answers, while fully synthetic tables may not retain the background structure of real data. We introduced SleuthBench to evaluate statistical discovery through controlled, validated interventions. Injecting curated statistical evidence reduces the usefulness of memorized answers to the original tables; reusable validators and automatic answer computation reduce manual ground-truth curation while retaining source-table context. We also introduced the Empirical Layer, which makes descriptive statistics, fitted EBM effects, and feature metadata available as structured evidence for agents. Across six LLMs, access to this layer raises feature-contribution accuracy from 41.9% to 68.0% and pooled accuracy from 64.6% to 76.7%, while leaving data-quality accuracy unchanged. Together, SleuthBench and the Empirical Layer provide a foundation for studying how data-science agents find and use statistical evidence.

AI USE STATEMENT

In this work, we used generative AI tools to provide feedback on research methodology or experiments, interpret results, refine hypotheses, and to implement methods such as the grading process in the SLEUTHBENCH pipeline. We have not used generative AI tools to assist with translation, clean and reformat datasets, or support qualitative and thematic data analysis. Generating synthetic data sets, helping develop theoretical models or conceptual frameworks, formulating mathematical claims, providing critical ingredients for proving mathematical claims, and assisting in the writing of proofs are not applicable to this work. Additionally, we used generative AI tools to create Figure 2, software code, draft parts of the research paper, edit the research paper to improve readability, identify relevant literature, format references, and suggest a structure for the research paper. We have reviewed all AI-assisted work. We take responsibility for the final content of this work, including text, claims, or artifacts produced with the aid of generative AI.

REFERENCES

AI4I. AI4I 2020 predictive maintenance dataset. UCI Machine Learning Repository, 2020. URL https://archive.ics.uci.edu/dataset/601/ai4i+2020+predictive+maintenance+dataset.

Sebastian Bordt, Harsha Nori, Vanessa Cristiny Rodrigues Vasconcelos, Besmira Nushi, and Rich Caruana. Elephants never forget: Memorization and learning of tabular data in large language models. In *First Conference on Language Modeling*, 2024. URL https://openreview.net/forum?id=HLoWN6m4fS.

Rich Caruana, Yin Lou, Johannes Gehrke, Paul Koch, Marc Sturm, and Noemie Elhadad. Intelligible models for healthcare: Predicting pneumonia risk and hospital 30-day readmission. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 1721–1730, 2015. doi: 10.1145/2783258.2788613.

Wenhu Chen, Hongmin Wang, Jianshu Chen, Yunkai Zhang, Hong Wang, Shiyang Li, Xiyou Zhou, and William Yang Wang. TabFact: A large-scale dataset for table-based fact verification. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=rkeJRhNYDH.

Zhi Chen, Sarah Tan, Urszula Chajewska, Cynthia Rudin, and Rich Caruana. Missing values and imputation in healthcare data: Can interpretable machine learning help? In *Proceedings of the Conference on Health, Inference, and Learning*, volume 209 of *Proceedings of Machine Learning Research*, pp. 86–99, 2023. URL https://proceedings.mlr.press/v209/chen23a.html.

Çağatay Demiralp, Peter J. Haas, Srinivasan Parthasarathy, and Tejaswini Pedapati. Foresight: Recommending visual insights. *Proceedings of the VLDB Endowment*, 10(12):1937–1940, 2017. URL https://www.vldb.org/pvldb/vol10/p1937-parthasarathy.pdf.

Hadi Fanaee-T. Bike sharing dataset. UCI Machine Learning Repository, 2013. URL https://archive.ics.uci.edu/dataset/275/bike+sharing+dataset.

Hadi Fanaee-T and João Gama. Event labeling combining ensemble detectors and background knowledge. *Progress in Artificial Intelligence*, 2:113–127, 2014. doi: 10.1007/s13748-013-0040-3.

John Hogue. Metro interstate traffic volume dataset. UCI Machine Learning Repository, 2019. URL https://archive.ics.uci.edu/dataset/492/metro+interstate+traffic+volume.

Sirui Hong, Yizhang Lin, Bang Liu, Bangbang Liu, Binhao Wu, Ceyao Zhang, Danyang Li, Jiaqi Chen, Jiayi Zhang, Jinlin Wang, Li Zhang, Lingyao Zhang, Min Yang, Mingchen Zhuge, Taicheng Guo, Tuo Zhou, Wei Tao, Robert Tang, Xiangtao Lu, Xiawu Zheng, Xinbing Liang, Yaying Fei, Yuheng Cheng, Yongxin Ni, Zhibin Gou, Zongze Xu, Yuyu Luo, and Chenglin Wu. Data interpreter: An LLM agent for data science. In *Findings of the Association for*

Computational Linguistics: ACL 2025, pp. 19796–19821. Association for Computational Linguistics, 2025. doi: 10.18653/v1/2025.findings-acl.1016. URL https://aclanthology.org/2025.findings-acl.1016/.

Xueyu Hu, Ziyu Zhao, Shuang Wei, Ziwei Chai, Qianli Ma, Guoyin Wang, Xuwu Wang, Jing Su, Jingjing Xu, Ming Zhu, Yao Cheng, Jianbo Yuan, Jiwei Li, Kun Kuang, Yang Yang, Hongxia Yang, and Fei Wu. InfiAgent-DABench: Evaluating agents on data analysis tasks. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 19544–19572, 2024. URL https://proceedings.mlr.press/v235/hu24s.html.

Yiming Huang, Jianwen Luo, Yan Yu, Yitong Zhang, Fangyu Lei, Yifan Wei, Shizhu He, Lifu Huang, Xiao Liu, Jun Zhao, and Kang Liu. DA-code: Agent data science code generation benchmark for large language models. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 13487–13521, 2024. doi: 10.18653/v1/2024.emnlp-main.748. URL https://aclanthology.org/2024.emnlp-main.748/.

Naman Jain, King Han, Alex Gu, Wen-Ding Li, Fanjia Yan, Tianjun Zhang, Sida Wang, Armando Solar-Lezama, Koushik Sen, and Ion Stoica. LiveCodeBench: Holistic and contamination-free evaluation of large language models for code. *arXiv preprint arXiv:2403.07974*, 2024. URL https://arxiv.org/abs/2403.07974.

Liqiang Jing, Zhehui Huang, Xiaoyang Wang, Wenlin Yao, Wenhao Yu, Kaixin Ma, Hongming Zhang, Xinya Du, and Dong Yu. DSBench: How far are data science agents from becoming data science experts? In *International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=DSsSPr0RZJ.

King County Housing. House sales in king county, usa. Kaggle, uploaded by harlfoxem, n.d. URL https://www.kaggle.com/datasets/harlfoxem/housesalesprediction.

Yuhang Lai, Chengxi Li, Yiming Wang, Tianyi Zhang, Ruiqi Zhong, Luke Zettlemoyer, Wen-tau Yih, Daniel Fried, Sida Wang, and Tao Yu. DS-1000: A natural and reliable benchmark for data science code generation. In *Proceedings of the 40th International Conference on Machine Learning*, pp. 18319–18345, 2023. URL https://proceedings.mlr.press/v202/lai23b.html.

Tianyang Liu, Fei Wang, and Muhao Chen. Rethinking tabular data understanding with large language models. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 450–482, 2024a. doi: 10.18653/v1/2024.naacl-long.26. URL https://aclanthology.org/2024.naacl-long.26/.

Xiao Liu, Zirui Wu, Xueqing Wu, Pan Lu, Kai-Wei Chang, and Yansong Feng. Are LLMs capable of data-based statistical and causal reasoning? benchmarking advanced quantitative reasoning with data. In *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 9215–9235, 2024b. doi: 10.18653/v1/2024.findings-acl.548. URL https://aclanthology.org/2024.findings-acl.548/.

Yin Lou, Rich Caruana, Johannes Gehrke, and Giles Hooker. Accurate intelligible models with pairwise interactions. In *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 623–631, 2013. doi: 10.1145/2487575.2487579.

Bodhisattwa Prasad Majumder, Harshit Surana, Dhruv Agarwal, Bhavana Dalvi Mishra, Abhijeetsingh Meena, Aryan Prakhar, Tirth Vora, Tushar Khot, Ashish Sabharwal, and Peter Clark. DiscoveryBench: Towards data-driven discovery with large language models. In *International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=vyflgpwfJW.

Harsha Nori, Samuel Jenkins, Paul Koch, and Rich Caruana. InterpretML: A unified framework for machine learning interpretability. *arXiv preprint arXiv:1909.09223*, 2019. URL https://arxiv.org/abs/1909.09223.

R. Kelley Pace and Ronald Barry. Sparse spatial autoregressions. *Statistics & Probability Letters*, 33(3):291–297, 1997. doi: 10.1016/S0167-7152(96)00140-X.

Panupong Pasupat and Percy Liang. Compositional semantic parsing on semi-structured tables. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*, pp. 1470–1480, 2015. doi: 10.3115/v1/P15-1142. URL https://aclanthology.org/P15-1142/.

Ronald K. Pearson. The problem of disguised missing data. *ACM SIGKDD Explorations Newsletter*, 8(1):83–92, 2006.

Bo Qiao, Liqun Li, Xu Zhang, Shilin He, Yu Kang, Chaoyun Zhang, Fangkai Yang, Hang Dong, Jue Zhang, Lu Wang, Minghua Ma, Pu Zhao, Si Qin, Xiaoting Qin, Chao Du, Yong Xu, Qingwei Lin, Saravan Rajmohan, and Dongmei Zhang. TaskWeaver: A code-first agent framework. *arXiv preprint arXiv:2311.17541*, 2023. URL https://arxiv.org/abs/2311.17541.

Sathishkumar V E, Changsun Shin, and Yongyun Cho. Steel industry energy consumption dataset. UCI Machine Learning Repository, 2021. URL https://archive.ics.uci.edu/dataset/851/steel+industry+energy+consumption.

scikit-learn. California housing dataset. scikit-learn Dataset Documentation, n.d. URL https://scikit-learn.org/stable/modules/generated/sklearn.datasets.fetch_california_housing.html.

Yuan Sui, Mengyu Zhou, Mingjie Zhou, Shi Han, and Dongmei Zhang. Table meets LLM: Can large language models understand structured table data? a benchmark and empirical study. In *Proceedings of the 17th ACM International Conference on Web Search and Data Mining*, pp. 645–654, 2024. doi: 10.1145/3616855.3635752.

Maojun Sun, Ruijian Han, Binyan Jiang, Houduo Qi, Defeng Sun, Yancheng Yuan, and Jian Huang. LAMBDA: A large model based data agent. *Journal of the American Statistical Association*, pp. 1–13, 2025. doi: 10.1080/01621459.2025.2510000. URL https://doi.org/10.1080/01621459.2025.2510000.

John W. Tukey. *Exploratory Data Analysis*. Addison-Wesley, Reading, Massachusetts, 1977. ISBN 0201076160.

Colin White, Samuel Dooley, Manley Roberts, Arka Pal, Benjamin Feuer, Siddhartha Jain, Ravid Shwartz-Ziv, Neel Jain, Khalid Saifullah, Sreemanti Dey, et al. LiveBench: A challenging, contamination-limited LLM benchmark. In *International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=sKYHBTAxVa.

Kanit Wongsuphasawat, Zening Qu, Dominik Moritz, Riley Chang, Felix Ouk, Anushka Anand, Jock Mackinlay, Bill Howe, and Jeffrey Heer. Voyager 2: Augmenting visual analysis with partial view specifications. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*, 2017. doi: 10.1145/3025453.3025768. URL https://idl.uw.edu/papers/voyager2.

Xianjie Wu, Jian Yang, Linzheng Chai, Ge Zhang, Jiaheng Liu, Xinrun Du, Di Liang, Daixin Shu, Xianfu Cheng, Tianzhen Sun, Guanglin Niu, Tongliang Li, and Zhoujun Li. TableBench: A comprehensive and complex benchmark for table question answering. *arXiv preprint arXiv:2408.09174*, 2024. URL https://arxiv.org/abs/2408.09174.

Dan Zhang, Sining Zhoubian, Min Cai, Fengzu Li, Lekang Yang, Wei Wang, Tianjiao Dong, Ziniu Hu, Jie Tang, and Yisong Yue. DataSciBench: An LLM agent benchmark for data science. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 3685–3728, 2026. doi: 10.18653/v1/2026.findings-acl.181. URL https://aclanthology.org/2026.findings-acl.181/.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. Judging LLM-as-a-judge with MT-Bench and chatbot arena. *arXiv preprint arXiv:2306.05685*, 2023. URL https://arxiv.org/abs/2306.05685.

APPENDIX

This appendix gives the detailed benchmark and evaluation specification. Appendix A catalogs the templates, injections, and paired question wordings. Appendix B describes source data and preprocessing; Appendix C defines reference answers and validation criteria; and Appendix D details the paired evaluation and grading. Appendix E reports detailed results. Appendix F presents a case study in which Empirical Layer artifacts help an agent identify the target interaction.

A TEMPLATES, INJECTIONS, AND QUESTIONS

SLEUTHBENCH defines 17 question templates across two families, implemented by 14 injectors. A template is a reusable task specification that defines compatible variables, an injection, two question framings, an answer format, and validation criteria.

Each accepted instance pairs data-science and business questions with a common table and a reference answer. Table 4 reproduces the data-science and business question templates. Uppercase braced placeholders are filled with instance-specific column names and values.

Table 4: The 17 benchmark templates and their injections, with the data-science and business question wording from the public template registry. Braced placeholders are filled when each instance is constructed.

Family	Template	What the injection changes	Question wording (data-science and business)
Data quality	Bad-row indicator	Corrupt a subset of target values and add a binary column whose positive entries mark exactly those rows.	*Data-science:* You can apply exactly one rule based on a single column to exclude problematic rows before analyzing {OUTCOME_COL}. Which column should the rule be based on? Return a single column name. *Business:* Some records have already been marked for review in a yes-or-no column. Which column contains that review marker? Return a single column name.
	Conditional anomaly indicator	Rewrite 5% of outcomes so that they remain within the observed support but become surprising given similar records. Add the true binary flag and same-prevalence placebo flags.	*Data-science:* Among the binary 0/1 columns, one indicator has 1 exactly on rows whose {OUTCOME_COL} outcome is conditionally inconsistent with otherwise similar records. The injected outcomes remain within the original target support. Which column is the indicator? Return a single column name. *Business:* Among the 0/1 columns, one indicator's 1 values mark records whose {OUTCOME_COL} outcome is inconsistent with otherwise similar records, even though the outcome itself remains within the dataset's observed range. Which column is that indicator? Return a single column name.

Family	Template	What the injection changes	Question wording (data-science and business)
	Category target outlier	Choose a sufficiently large category and shift its target values up or down until its mean stands out from the other categories.	*Data-science:* Is there a category in {FEATURE} whose {OUTCOME_COL} values are unusually high or low compared with the other categories? Return only the category label. *Business:* Is there a category in {FEATURE} whose {OUTCOME_COL} values stand out from the other categories? Return only the category label.
	Conditional bad rows	Change the targets of a small set of rows so they remain plausible in the overall target distribution but are unusual given their feature values.	*Data-science:* Some rows have {OUTCOME_COL} values that are plausible globally but unusual given their feature values. Identify the {N_INJECT_SAMPLES} affected rows and return their row_id values as a comma-separated list. *Business:* There are {N_INJECT_SAMPLES} rows in this data with an unusual pattern in {OUTCOME_COL}. Return the row_id values as a comma-separated list.
	Underpowered group evidence	Add a categorical group column with two categories: a common group A and a very rare group B; leave the target unchanged so the group comparison has a wide interval.	*Data-science:* For categorical feature {GROUP_COL}, is there enough statistical evidence to distinguish group A from group B with respect to {OUTCOME_COL}? Return one of: enough evidence, not enough evidence. *Business:* For the categorical feature {GROUP_COL}, is the evidence strong enough to say the groups behave differently with respect to {OUTCOME_COL}? Return one of: enough evidence, not enough evidence.
	Semantic missing label	Replace a small number of values in a categorical feature with a distinct missing-like label such as "Unknown" or "Not reported."	*Data-science:* Which category label in {FEATURE} appears to encode missing, unknown, or not-reported values? Return only the label. *Business:* Are there any labels in {FEATURE} that we need to pay attention to? If yes, return only the label. If no, return No.
	Target-dependent missing label	Rename a categorical feature's categories, then place rows from one target tail into an additional neutral bucket. The moved rows lose their original category, so the bucket acts as an undocumented missing-value code. In real data, missing values are often recorded as ordinary-looking codes that the metadata does not identify, a problem known as disguised missing data (Pearson, 2006).	*Data-science:* The category labels in {FEATURE} have been anonymised. One category label encodes missing or unknown values. Which missing-like label shows evidence of target-dependent missingness with respect to {OUTCOME_COL}? Return only the label. *Business:* Is there a category in {FEATURE} whose {OUTCOME_COL} values stand out from the other categories? Return only the category label.

Family	Template	What the injection changes	Question wording (data-science and business)
	Unreliable feature	Add target noise whose variance increases with one numerical feature, making the feature–target relationship much less consistent at one end.	*Data-science:* Which feature's relationship with {OUTCOME_COL} is most unreliable — i.e., has the highest prediction uncertainty? Return a single column name. *Business:* For one column, {OUTCOME_COL} results stay fairly close together at one end of the column but become much more spread out at the other end. Which column shows this uneven level of consistency? Return a single column name.
Feature contribution	Noise feature	Permute one numerical feature across rows, preserving its marginal distribution while breaking its row-wise relationship with the target.	*Data-science:* One numeric feature in this dataset has zero predictive importance for {OUTCOME_COL} — it contributes nothing to a predictive model. Which feature is it? Return a single column name. *Business:* If we had to remove one number-based column before trying to estimate {OUTCOME_COL}, which one could we remove with the least loss of useful information? Ignore columns used only to identify records. Return a single column name.
	Monotonicity reversal	Add a V-shaped effect to a numerical feature whose original binned relationship with the target was monotone, forcing a direction change.	*Data-science:* For each numeric feature, classify its relationship with {OUTCOME_COL} as MONOTONE (consistently increasing or decreasing) or NON-MONOTONE (changes direction at least once). Which feature has the most non-monotone relationship with the outcome? Return only the feature name. *Business:* For most number-based columns, {OUTCOME_COL} generally keeps moving in one direction as the column value changes. Which column breaks that simple pattern most clearly? Return only the column name.

Family	Template	What the injection changes	Question wording (data-science and business)
	Inverted-U peak	Add a smooth effect centered at an interior feature value that declines on both sides; the reference peak is measured on the edited table.	*Data-science:* One feature has an inverted-U relationship with {OUTCOME_COL} — performance peaks at a specific value and declines on both sides. Which feature, and at what value is the peak? Return your answer as: column_name, value *Business:* One column has a sweet spot for {OUTCOME_COL}: results are best around a middle value and worse when the value is either lower or higher. Which column is it, and approximately where is the sweet spot? Return your answer as: column_name, value
	Plateau threshold	Add a clipped-linear effect: the target changes with a numerical feature up to a threshold and then levels off.	*Data-science:* One feature has a threshold effect on {OUTCOME_COL} — it matters up to a point, then its effect levels off. Which feature, and at approximately what value does the effect plateau? Return your answer as: column_name, value *Business:* For one column, increasing its value is linked to changes in {OUTCOME_COL} only up to a point. After that, further increases make little difference. Which column is it, and around what value does that happen? Return your answer as: column_name, value
	Conditional direction	Add an XOR-shaped target effect over median splits of two numerical features, so their joint pattern dominates their marginal effects.	*Data-science:* Find the pair of features with the strongest interaction effect on {OUTCOME_COL}. Sort them alphabetically. When the first feature is above its median, does increasing the second feature tend to increase or decrease {OUTCOME_COL}? Return exactly: POSITIVE or NEGATIVE *Business:* First find the two columns that reveal the strongest pattern when looked at together, and put their names in alphabetical order. Among records where the first column is in its upper half, does increasing the second usually make {OUTCOME_COL} go up or down? Return exactly: POSITIVE for up or NEGATIVE for down.

Family	Template	What the injection changes	Question wording (data-science and business)
	Dominant pair	Add an XOR-shaped target effect over median splits of two numerical features, so their joint pattern dominates their marginal effects.	*Data-science:* Which pair of features has the strongest interaction effect on {OUTCOME_COL}? Return the two column names as: column_a, column_b *Business:* Which two columns look fairly ordinary when considered separately but reveal the strongest unusual pattern in {OUTCOME_COL} when looked at together? Return the two column names as: column_a, column_b
	Antagonistic reversal	Add positive effects for each of two conditions separately but a negative effect when both conditions hold.	*Data-science:* What are the two features whose combination has an effect opposite to each individual feature? Return the two column names as: column_a, column_b *Business:* Which pairs of features behave unexpectedly when considered together? Return the two column names as: column_a, column_b
	Compensatory reversal	Give the two conditions opposing individual effects, then add a positive joint interaction that reverses one individual direction.	*Data-science:* What are the two features whose combination reverses the direction suggested by one individual feature? Return the two column names as: column_a, column_b *Business:* Which pairs of features behave unexpectedly when considered together? Return the two column names as: column_a, column_b
	Positive synergy	Give each of two conditions a positive individual effect and add a further positive effect when both hold.	*Data-science:* What are the two features whose combination has an extra positive effect beyond each individual feature? Return the two column names as: column_a, column_b *Business:* Which pairs of features behave unexpectedly when considered together? Return the two column names as: column_a, column_b

Across the six datasets, construction produces 70 validated dataset–template combinations evaluated by all six models. Template applicability and task-specific validation determine which combinations enter evaluation.

B Source Data and Preparation

The experiments use 1,000-row versions of six public datasets (Table 5). Five sources are observational; AI4I is synthetic data designed to reflect industrial predictive maintenance (AI4I). Together they provide transportation, housing, maintenance, and energy-use settings with mixed numerical and categorical variables.

Metro and Steel timestamps are converted to month variables; Metro's literal holiday category `None` becomes `No Holiday`. The experiments use the 1,000-row versions of the standardized table.

Table 5: Source datasets. Each experimental base table has 1,000 rows before injection. Source row counts refer to the full prepared snapshots.

Dataset	Source rows	Target	Reference
AI4I 2020 Predictive Maintenance	10,000	Machine failure	(AI4I)
Bike Sharing	17,379	Rental count	(Fanaee-T, 2013)
California Housing	20,640	House value	(scikit-learn)
King County Housing	21,613	Sale price	(King County Housing)
Metro Interstate Traffic Volume	48,204	Traffic volume	(Hogue, 2019)
Steel Industry Energy Consumption	35,040	Energy use	(Sathishkumar V E et al., 2021)

Bike Sharing is associated with Fanaee-T & Gama (2014); California Housing derives from the census data studied by Pace & Barry (1997).

C Construction and Validation Details

We construct each benchmark instance from a prepared base table and a compatible question template. Construction has three steps: apply an injection, validate the edited table, and compute the reference answer.

Template matching and injection. We match templates against the table's column summary. Each template specifies eligible column types and cardinalities, dataset-level requirements, question slots, and injector parameters. For a compatible match, we assign columns and values to the slots and resolve the injector parameters. We apply each distinct injector and parameter setting to a copy of the base table with a fixed random seed derived from the global seed. The injector returns an edited table and a record of the realized intervention. Each output table contains one intervention, although multiple question templates can share it. We save the table with a manifest recording the source, seed, target, parameters, intervention record, slot assignments, and rendered question. The reference answer remains unset. Incompatible templates and unsuccessful interventions produce no instance.

Validation of the edited table. Validation is a separate stage that runs the validator registered for the injected phenomenon on each saved instance. The validator uses the intervention record to identify the intended signal, then measures it on the edited table. Acceptance criteria depend on the task. They can require a minimum number of rows in each compared group or bin, a specified shape or direction, sufficient signal strength, or a margin over competing features or pairs when the question requires a unique answer. The manifest records each validation check's measurement, threshold, and decision, together with the overall validation result. Because a single random draw may not yield an unambiguous signal on a given table, some injectors try several candidate interventions, such as a different feature or a fresh random draw. They run the same validator on each candidate and keep the first one that passes, and they reject the injection if none passes within a fixed number of attempts. Since the validator reads only the table and the intervention record, the saved table is still validated independently before acceptance.

Reference-answer computation. For each validated instance, we call the answer function registered for each applicable question template. The function receives the edited table, the template's slot assignments, and the intervention record. It writes the computed answer to the corresponding question entry in the manifest. Answer functions are template-specific because different questions can refer to the same intervention. Depending on the task, the answer is a column, column pair, category, decision, set of row identifiers, or column and value.

Reference answers follow explicit task definitions. Data-science and business phrasings of the same template use the same intervention and answer definition. Only questions with a passing validation result and a computed answer enter model evaluation.

D Evaluation Protocol and Grading

D.1 Paired Evaluation and Coverage

After the reference answers are computed, we evaluate six models: Claude Sonnet 5, Claude Opus 4.8, Claude Fable 5, and GPT-5.6 Sol, Terra, and Luna. Data-science and business questions are evaluated under two tool conditions: Python alone (P) and Python plus the Empirical Layer (P+E).

Each P/P+E pair consists of two responses to the same model–question instance, one under each tool condition. Data-science and business questions are treated as separate pairs.

Every model is evaluated on the same 140 questions: data-science and business questions for 70 dataset–template combinations. This gives 840 model–question cases per condition and 1680 responses in total. Data-science and business phrasings each contribute 420 cases per condition.

The P+E condition permits retrieval of structured statistics, EBM outputs, and language-based metadata. Agents use the interface in 611/840 P+E responses (72.7%), with 4120 retrieval calls in total. These comparisons measure the effect of access to the combined interface.

D.2 Grading

Responses receive Correct, Partial, or Incorrect; accuracy counts only Correct. Grades were assigned by a pipeline combining deterministic checks with an LLM judge (`gpt-4o-mini`). Evaluation errors and empty answers were assigned Incorrect without LLM judging. Unambiguous numerical and column–value answers were checked directly against the reference answer; remaining responses were assessed by the LLM judge using the question, reference answer, and model response.

Numerical values use a 5% relative tolerance. A column–value answer must also identify the correct column. Partial is used when a response contains a correct component but does not fully answer the question. Examples include an incomplete list or column pair, the correct qualitative direction with an incorrect requested value, and the correct column paired with a value outside the tolerance. A standalone numerical answer outside the tolerance is Incorrect. A response that gives the correct answer remains Correct even if it adds an explanation despite an answer-only instruction, provided the explanation does not contradict or obscure the answer.

All reported results use the grades produced by this pipeline without manual adjustment; the graded result files are preserved.

Execution failures are distinguished from missing evaluations. The GPT runs that reached iteration or token limits have assigned Incorrect grades and remain in the response counts.

D.3 Prompt Formats

We reproduce the system prompts verbatim. The evaluation prompt is given to every evaluated model; the grading prompt is given to the LLM judge, followed by the question, the reference answer, and the model response.

Evaluation system prompt

"You are a data analyst. Answer the question about the dataset concisely. Return ONLY the answer in the exact format requested with no extra text. You may analyze only the preloaded pandas DataFrame `df` and outputs returned by the declared tools. Do not inspect or access the filesystem, directories, environment variables, network, subprocesses, source code, manifests, metadata artifacts, or answer files. Do not use open(), pathlib, os, glob, subprocess, socket, or shell commands."

Grading system prompt

"You are grading an LLM's answer to a data analysis question about a CSV dataset.

You will be given:
- The question asked
- The expected correct answer
- The model's answer

Grade the response as exactly one of:
- CORRECT: The answer matches the expected answer (allow minor formatting differences, extra explanation is fine as long as the core answer is right). For a single numeric value, treat it as CORRECT when it is within 5% relative error of the expected value.
- PARTIAL: The answer is partially correct (e.g. some items in a list are right, or the direction is right but the specific value is wrong)
- INCORRECT: The answer is wrong, irrelevant, or doesn't address the question

Return JSON with `"grade"` *and* `"reasoning"` *fields."*

E Detailed Results

Each N below counts model–question cases per condition; P and P + E are *Python alone* and *Python + Empirical Layer* as in Section 5.

Table 6: Accuracy (%) by model and template family under P and P + E.

	Data quality			Feature contribution			Total		
Model	P	P+E	Δ	P	P+E	Δ	P	P+E	Δ
Claude Sonnet 5	77.6	77.6	0.0	18.8	53.1	+34.4	50.7	66.4	+15.7
Claude Opus 4.8	84.2	84.2	0.0	29.7	65.6	+35.9	59.3	75.7	+16.4
Claude Fable 5	85.5	88.2	+2.6	45.3	75.0	+29.7	67.1	82.1	+15.0
GPT-5.6 Sol	86.8	85.5	-1.3	71.9	78.1	+6.3	80.0	82.1	+2.1
GPT-5.6 Terra	85.5	85.5	0.0	51.6	82.8	+31.3	70.0	84.3	+14.3
GPT-5.6 Luna	82.9	82.9	0.0	34.4	53.1	+18.8	60.7	69.3	+8.6
All models	83.8	84.0	+0.2	41.9	68.0	+26.0	64.6	76.7	+12.0

Table 7: Grade distributions by model and by family. C/P/I denotes CORRECT/PARTIAL/INCORRECT.

Model or family	N	P: C/P/I	P+E: C/P/I
Claude Sonnet 5	140	71/12/57	93/8/39
Claude Opus 4.8	140	83/9/48	106/8/26
Claude Fable 5	140	94/8/38	115/10/15
GPT-5.6 Sol	140	112/9/19	115/9/16
GPT-5.6 Terra	140	98/9/33	118/6/16
GPT-5.6 Luna	140	85/8/47	97/10/33
Data quality	456	382/24/50	383/12/61
Feature contribution	384	161/31/192	261/39/84
Total	840	543/55/242	644/51/145

F Case Studies and Agent Trajectories

The Metro Interstate Traffic Volume case compares GPT-5.6 Luna's analyses of the same dominant-interaction business question (Table 11). The reference pair is `month, temp`. The injected effect is positive when exactly one feature exceeds its median and negative otherwise, making the joint pattern central to the task. Python-only exploration compares several interaction proxies, including

Table 8: Accuracy by model and question framing. Every model is evaluated on both framings of the same 70 dataset–template combinations.

Model	Framing	N	P (%)	P+E (%)	Δ (pp)
Claude Sonnet 5	Business	70	47.1	67.1	+20.0
Claude Sonnet 5	Data science	70	54.3	65.7	+11.4
Claude Opus 4.8	Business	70	52.9	74.3	+21.4
Claude Opus 4.8	Data science	70	65.7	77.1	+11.4
Claude Fable 5	Business	70	62.9	84.3	+21.4
Claude Fable 5	Data science	70	71.4	80.0	+8.6
GPT-5.6 Sol	Business	70	87.1	81.4	-5.7
GPT-5.6 Sol	Data science	70	72.9	82.9	+10.0
GPT-5.6 Terra	Business	70	68.6	87.1	+18.6
GPT-5.6 Terra	Data science	70	71.4	81.4	+10.0
GPT-5.6 Luna	Business	70	54.3	67.1	+12.9
GPT-5.6 Luna	Data science	70	67.1	71.4	+4.3

Table 9: Accuracy by question framing, pooled over all six models and both template families.

Framing	P (%)	P+E (%)	Δ (pp)
Business	62.1	76.9	+14.8
Data science	67.1	76.4	+9.3

Table 10: Accuracy by template family and question framing, pooled over all six models.

Family	Framing	N	P (%)	P+E (%)	Δ (pp)
Data quality	Business	228	86.8	86.8	0.0
Data quality	Data science	228	80.7	81.1	+0.4
Feature contribution	Business	192	32.8	65.1	+32.3
Feature contribution	Data science	192	51.0	70.8	+19.8

empirical mutual information, before selecting `weather_description, month`. With Empirical Layer access, Luna inspects schema and metadata, retrieves target associations, and compares six EBM interaction graphs before selecting the reference pair.

Table 11: GPT-5.6 Luna on the Metro Interstate Traffic Volume business-language dominant-interaction question. The reference pair is `month, temp`.

Condition	Analysis sequence	Final pair	Grade
P	Compare interaction proxies, including empirical mutual information	`weather_description, month`	Incorrect
P+E	Inspect metadata and target associations; compare six EBM interaction graphs	`month, temp`	Correct

The Python-selected pair has 28 weather categories and 12 month values, allowing 336 joint cells for 1,000 rows. Sparse counts can favor such a pair under uncorrected empirical mutual information, providing a possible explanation for the final ranking. The P+E trajectory instead compares fitted interaction terms under a common model. The `temp–month` term has the largest interaction importance in the stored EBM artifacts, consistent with the agent's answer.