Benchmarking Large Language Models for Data Quality Diagnosis

Anonymous ACL submission

Abstract

Data agents built on large language models are increasingly trusted to clean and preprocess tabular datasets, but their evaluations entangle diagnosis with repair, leaving open a basic question: can LLMs actually identify what is wrong with a dataset? We present DATAPRISM, a fine-grained benchmark that evaluates the data diagnosis capability of LLMs. We first formulate this task as a closed-set multi-label task: given a dirty dataset and candidate label definitions, a model must identify the applicable data quality problems. We then construct DATAPRISM which contains 40 formally specified labels derived from a data preprocessing taxonomy and 1,362 verified dirty datasets constructed from clean baselines through a controlled injection-audit pipeline, covering 8 domains, 3 injection intensity levels, and both single-label and multi-label settings. Evaluating 10 representative LLMs reveals a pronounced symptom–mechanism gap: models achieve hit rates above 80% on locally visible defects but fall below 20% on mechanism-level labels requiring multi-column statistical reasoning. Even the strongest model rarely recovers the full defect set, achieving only 2.8% exact match. We release our benchmark data and evaluation code at https://anonymous.4open.science/status/LLMDQD-9A8A.

1 Introduction

Large language models (LLMs) are increasingly used as data agents that inspect, clean, transform, and analyze tabular datasets. Recent systems such as AutoDCWorkflow (Li et al., 2024), AutoPrep (Fan et al., 2024), DeepPrep (Fan et al., 2026), CleanAgent (Qi et al., 2024), and IterClean (Ni et al., 2024) typically evaluate this ability through end-to-end cleaning or workflow performance. However, before repair, the agent must first answer a more basic question: *what data quality problems are present in the dataset?* A failed cleaning pipeline may result from incorrect diagnosis, inappropriate repair, or execution failure. Without isolating diagnosis, it is difficult to identify the missing capability.

Table 1: Comparison of DATAPRISM with related benchmarks on key design dimensions. "✓" means YES; "×" means NO; "–" means Not Applicable / Not Reported.

Benchmark	#Types	Formal	Verified	#Inst.	Diagnosis
Hospital/Flights	2–4	×	×	3	×
AutoDCWorkflow	4	×	×	∼50	×
Bendinelli et al. (2025)	5	×	×	∼30	✓
DS-1000	–	–	–	1,000	×
MLAgentBench	–	–	–	15	×
DSAEval	–	–	–	285	×
DATAPRISM	**40**	✓	✓	**1,362**	✓

Data quality diagnosis is therefore both important and difficult: it determines whether downstream cleaning actions target the right problem, yet it is more than a simple pattern-matching task (García et al., 2015). Some defects are locally observable from cells, rows, or individual columns, such as missing entries, duplicate tuples, case inconsistency, or out-of-range values. Others require reasoning about global or multi-column statistical structure, such as heteroscedasticity, class-label noise, joint distribution shift, or class-boundary overlap. These two groups pose qualitatively different challenges: surface symptoms can often be detected by direct inspection, whereas mechanism-level defects require the model to infer distributional relationships not visible from isolated column summaries. Yet existing data-agent and data-science benchmarks are not designed for such fine-grained diagnosis: traditional cleaning benchmarks contain few error types and instances (Chu et al., 2016), general data-science benchmarks evaluate broader coding or analysis abilities (Lai et al., 2023; Huang et al., 2024; Sun et al., 2026), and prior diagnostic studies cover only coarse-grained defects at limited scale (Bendinelli et al., 2025). This raises a cen-

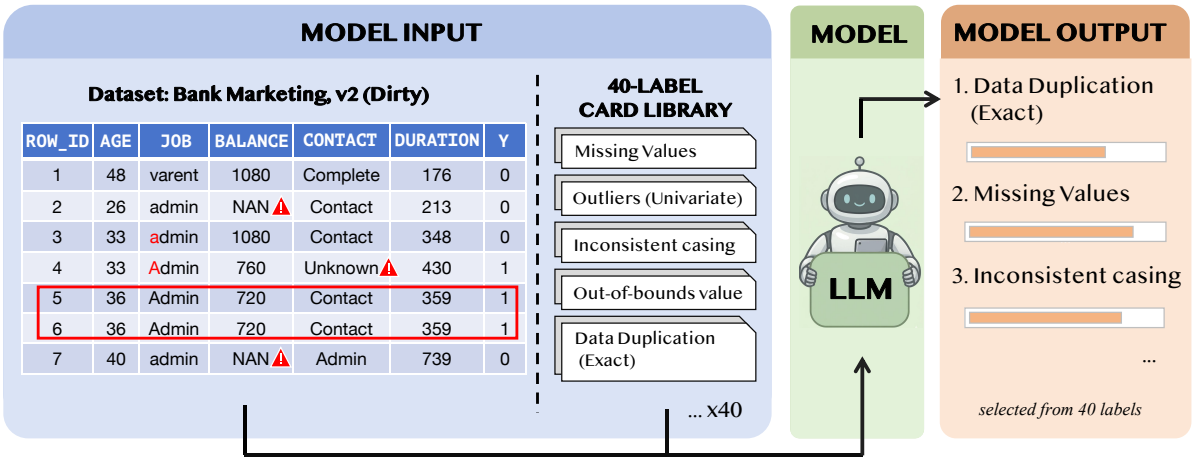

Figure 1: Illustration of the DATAPRISM evaluation task. The model receives a complete dirty dataset together with the name and formal mathematical definition of 40 candidate labels as input, and outputs the subset of labels it considers applicable. The task is formalized as closed-set multi-label classification.

tral question: *Can diagnose data quality problems beyond locally visible symptoms, including defects that require reasoning about deeper data-generating mechanisms?*

To this end, we introduce DATAPRISM, a fine-grained benchmark designed to evaluate data quality diagnosis as an independent capability. Its construction follows three principles: broad coverage of heterogeneous data quality problems, precise and operational label definitions, and label-level instance verification so that model errors can be attributed to diagnostic behavior rather than benchmark ambiguity. Firstly, we formulate diagnosis as a closed-set multi-label classification task: given a complete dirty dataset and a library of candidate label definitions, the model outputs the subset of data quality labels that best describe the dataset (Figure 1). Secondly, we define 40 formally specified labels for data quality problems, drawing from the data preprocessing taxonomy of García et al. (2015). These labels cover missing values, outliers, noise, redundant attributes, distribution shift, format inconsistency, label noise, high-dimensional sparsity, and more. Based on these labels, we further construct noise injection functions and corresponding noise audit functions for each defect type, ensuring that data corruption and subsequent diagnosis can be performed in a controlled and verifiable manner. Thirdly, we design a pipeline to construct benchmark instances through controlled injection. We first generate clean baseline datasets using public dataset schemas derived from widely used repositories such as OpenML[1] and UCI[2], in combination with appropriate statistical distributions. We then inject one or more target defects at three intensity levels, producing both isolated single-label cases and realistic multi-label combinations using noise injection functions. After injection, noise audit functions are executed to identify intended labels and possible side-effect labels; ambiguous cases are regenerated, revised, or reviewed before inclusion. This process yields 1,362 verified dirty datasets spanning 8 real-world domains (including healthcare, finance, transportation, education, and more), and covers a range of label complexities, including single-label, two-to-five-label, and five-plus-label settings (Table 1 compares DATAPRISM with existing benchmarks along these dimensions).

We evaluate 10 representative LLMs under a unified zero-shot closed-set protocol and observe a consistent *symptom–mechanism gap*: models achieve hit rates above 80% on locally visible defects such as missing values and duplicate tuples, but fall below 20% on mechanism-level labels such as heteroscedasticity, joint distribution shift, and class-label noise. Even the strongest model reaches only 32.9% Overall score and 2.8% exact match, suggesting that current LLMs remain far from complete fine-grained data quality diagnosis under this setting.

Our contributions are threefold:

- We isolate data quality diagnosis from repair and execution, formulating it as a closed-set multi-label task that enables direct evaluation

[1] https://www.openml.org/
[2] https://archive.ics.uci.edu/

of diagnostic behavior.

- We construct DATAPRISM, containing 1,362 verified dirty datasets across 40 formal labels, 8 domains, and 3 injection intensities, enabling controlled evaluation across surface-level and mechanism-level defects

- We evaluate 10 LLMs and reveal a robust symptommechanism gap: models reliably detect visible surface defects but fail on defects requiring multi-column statistical reasoning.

2 Related Work

2.1 LLM-Based Data Preprocessing

Recent work increasingly treats LLMs as autonomous agents for data preparation and cleaning. AutoDCWorkflow (Li et al., 2024) generates executable cleaning workflows from natural language specifications, while AutoPrep (Fan et al., 2024) and DeepPrep (Fan et al., 2026) formulate preparation as multi-step agentic pipelines. Other systems focus on particular cleaning settings: CleanAgent (Qi et al., 2024) targets data standardization, and IterClean (Ni et al., 2024) studies iterative LLM-assisted cleaning. Although these systems demonstrate the promise of LLMs for automating data preprocessing, their evaluations are typically end-to-end: a model is judged by the quality of the resulting workflow, cleaned table, or downstream performance. As a result, diagnostic behavior is entangled with repair planning, tool use, and execution, making it difficult to determine whether a failure comes from misidentifying the underlying data quality problem or from choosing and carrying out an inappropriate repair.

A smaller line of work examines LLMs' ability to reason about data quality problems more directly. Bendinelli et al. (2025), for example, show that LLM agents struggle with globally structured defects in tabular machine learning datasets. However, their study covers only a small set of coarse-grained error types and a limited number of test instances, leaving open how LLMs behave under a broader and more fine-grained taxonomy of data quality problems. DATAPRISM addresses this gap by isolating diagnosis as a standalone closed-set multi-label task, with 40 formally defined labels and 1,362 instance-verified dirty datasets.

2.2 Data Science Benchmarks

Recent data science benchmarks evaluate broad capabilities such as code generation, machine learning experimentation, and data analysis. For example, DS-1000 (Lai et al., 2023) focuses on data science code generation, MLAgentBench (Huang et al., 2024) evaluates autonomous agents for machine learning experiments, and DSAEval (Sun et al., 2026) covers diverse real-world data science tasks. These benchmarks assess general data-centric reasoning, but they do not specifically test whether models can localize and distinguish fine-grained data quality problems. Conversely, traditional cleaning benchmarks such as Hospital/Flights (Chu et al., 2016) center on data quality but contain only a few error types and instances (Table 1), limiting systematic diagnostic evaluation. DATAPRISM fills this gap by combining broad problem coverage with formal label definitions and instance-level verification.

3 Methodology

3.1 Task Formulation

Data quality diagnosis is formalized as a closed-set multi-label classification task. Let $\mathcal{D}$ denote the space of tabular datasets, $\mathcal{L}$ the predefined label space of data quality problems, and $\mathcal{G} = \{d_\ell : \ell \in \mathcal{L}\}$ the corresponding definition set, where d_ℓ specifies the application conditions of label ℓ.

Given $(D, \mathcal{G})$, where $D \in \mathcal{D}$, a large language model M produces a predicted label set $\hat{Y} = M(D, \mathcal{G})$, where $\hat{Y} \subseteq \mathcal{L}$. The ground-truth label set is denoted as $Y^* \subseteq \mathcal{L}$. The task is closed-set because predictions are restricted to $\mathcal{L}$, and multi-label because a dataset may contain multiple simultaneously valid quality problems. This formulation concerns diagnosis only: it specifies which quality problems are present, without downstream repair, cleaning, or code execution.

3.2 Label Taxonomy Construction

García et al. (2015) provides a systematic treatment of preprocessing problems, spanning common data defects, statistical assumption violations, and representation issues. We therefore use it as the structured source for constructing the DATAPRISM label taxonomy. The goal is not to enumerate every possible defect, but to obtain a broad, formally defined, and operationally testable set of diagnostic labels.

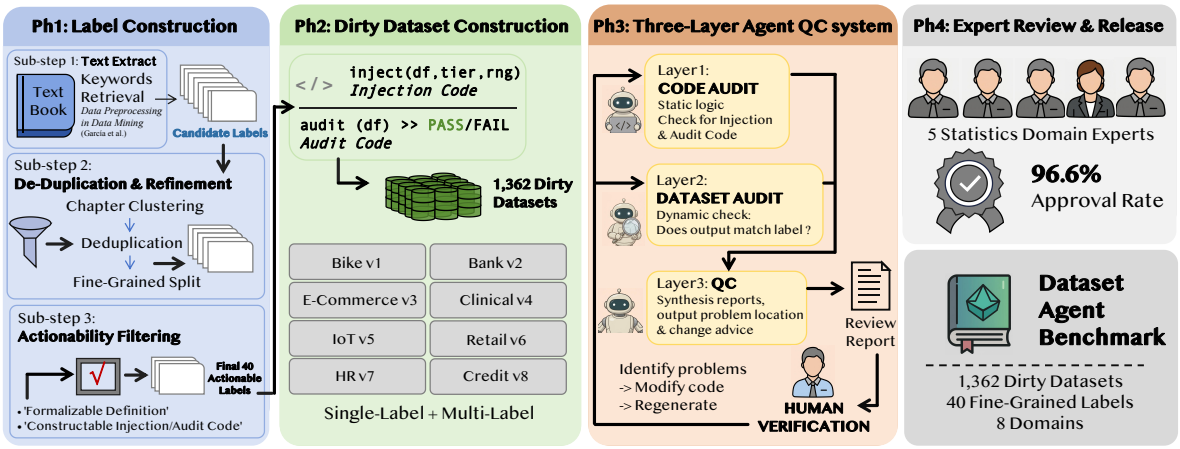

Figure 2: Full DATAPRISM construction pipeline. The pipeline first derives a formally defined label taxonomy, then constructs dirty datasets through controlled injection and independent audit functions, and finally applies agent-assisted QC plus human expert validation before benchmark release.

The pipeline proceeds in stages: we first extract and aggregate coarse label candidates from textbook, then split them into fine-grained labels, and finally retain only labels that can be both formally defined and operationally instantiated.

Label candidate extraction. We parse the textbook with MinerU[3] into LaTeX-style structured text preserving pages, sections, tables, and mathematical expressions. Let t_i denote the parsed content of page i. An LLM extractor E reads each page and returns candidate tuples $C_i = E(t_i) = \{(n, e, c)\}$, where n is a candidate label name, e is the supporting evidence span, and c is local context. We take $C = \bigcup_i C_i$ and merge candidates that differ only in wording, while retaining chapter and page information to separate distinct phenomena.

Fine-grained label splitting. Coarse candidates often conflate distinct diagnostic mechanisms under the same surface term, making model errors difficult to interpret. We therefore call an LLM splitter to refine each coarse candidate $c \in C$ according to the textbook's concrete descriptions, subcategories, evidence spans, and operational conditions, producing a set of mechanism-level candidates $S(c) = \{\ell_{c,1}, \ldots, \ell_{c,k}\}$. The resulting candidate pool is $L_{\text{raw}} = \bigcup_{c \in C} S(c)$. Semantically equivalent variants in L_{raw} are then merged to obtain a refined fine-grained candidate set.

Label formalization and filtering. Fine-grained candidates remain natural-language concepts unless their boundaries and observable trigger conditions are made explicit. We therefore formalize and filter them so that every retained label supports controlled injection, independent auditing, and unambiguous evaluation. For each candidate $\ell \in L_{\text{raw}}$, we construct a definition $d_\ell = (o_\ell, f_\ell, \theta_\ell, b_\ell)$, covering data objects, trigger condition, operational parameters, and boundary exclusions. Let $\text{Accept}(\ell)$ denote that ℓ is both formalizable and operationalizable. The final label space is:

$$\mathcal{L} = \{\ell \in L_{\text{raw}} : \text{Accept}(\ell)\}. \quad (1)$$

Here operationalizable means that we can implement both an injection function that introduces the defect and an independent audit function that checks f_ℓ. The retained labels cover missing values, outliers, attribute noise, redundancy, distribution shift, format inconsistency, label noise, high-dimensional sparsity, and related statistical assumption violations. Appendix B lists the full label set.

Boundary control. For every retained label, we explicitly record exclusions against nearby labels. For instance, "presence of missing entries" only checks whether any value is unobserved, while "high missing rate" requires a rate threshold and "missing-value distributional deviation" requires a shift between complete and observed distributions. This boundary documentation is used both during dataset construction and during expert review, reducing ambiguity when multiple labels are superficially related.

3.3 Dirty Dataset Construction

Benchmark instances are created through a controlled generation pipeline. The pipeline screens

[3] https://github.com/opendatalab/MinerU

clean baseline tables, applies label-specific injection functions, verifies the resulting dirty tables with independent audit functions, adjudicates valid side-effect labels, and records generation metadata for reproducibility. This subsection gives the operational details of each step; aggregate statistics are reported separately in Section 3.5.

Clean baseline screening. Each baseline table has a fixed schema, typed columns, and domain metadata. Before injection, it must pass the full audit suite with every label returning "absent"; otherwise it is discarded. This ensures that gold labels are introduced by construction rather than inherited from pre-existing defects.

Injection functions. For each label ℓ, an injection function I_ℓ maps a clean table D_0, a target column set C, an intensity tier t, and a random seed s to a dirty table:

$$D' = I_\ell(D_0, C, t, s). \tag{2}$$

The intensity tiers are designed to test signal strength. The weakest setting satisfies the formal trigger condition with a small margin, the moderate setting represents a plausible defect, and the strongest setting produces a more salient signal. For row- or cell-level defects, tiers are parameterized by affected-row or affected-cell rates; for distribution-level defects, they are parameterized by effect size, shift magnitude, variance ratio, or neighborhood-structure change, depending on the label definition.

Audit functions. Each label also has an audit function $A_\ell(D) \in \{0, 1\}$ derived from the formal trigger condition of d_ℓ. Injection and audit code are implemented separately: injection code modifies data, while audit code only reads the resulting dataset and evaluates the formal condition. After every injection, the complete audit suite is run. If K labels are intended, the target labels should return "present" and unrelated labels should return "absent" unless a side effect is semantically valid.

Single-label and multi-label construction. Single-label instances contain a single intended target label and are balanced across the intensity tiers. Multi-label instances are constructed by sequentially applying multiple injection functions. When possible, different labels are assigned to different columns or feature groups to avoid interference. For mechanism-level labels that necessarily involve multiple columns, the affected column group is recorded in the metadata. The order of injection and the random seed are stored for reproducibility.

Side-effect handling. Some injections can legitimately trigger additional labels. For example, a large-magnitude value error may also create an out-of-range value under a domain constraint. We therefore apply a retain-or-reinject rule. If the additional label is semantically valid under its formal definition, it is added to the gold set; if it is an implementation artifact, the injection is revised and regenerated. This prevents hidden label leakage while preserving real multi-label cases.

3.4 Quality Assurance

The benchmark uses complementary checks: formal audits and human expert review. Only the formal audits and human review determine final inclusion; agent reports surface possible construction defects.

Formal audit gate. Every generated dataset must pass a full audit matrix over the complete label space. For instance i, let $Y_i^{\text{audit}} = \{\ell : A_\ell(D_i) = 1\}$. The instance is eligible only if Y_i^{audit} is consistent with the intended gold set after side-effect adjudication. Datasets with unresolved audit conflicts are regenerated or excluded.

Agent-assisted Quality Control. We use a layered agent QC system to identify construction errors before human review. The code-audit layer statically inspects injection and audit scripts for no-op branches, type incompatibilities, and parameter mistakes. The dataset-audit layer checks whether summary statistics and distributional evidence match the target label definition. The final QC layer compares both reports and proposes a root cause when they disagree. Human reviewers inspect the reports and modify the code or generation recipe, not the dirty dataset itself, so fixes remain reproducible.

Expert validation. All released datasets and their gold labels are reviewed by statistics experts. Reviewers are shown the dirty dataset summary, proposed gold labels, affected columns, label definitions, and audit evidence. An instance is approved when the assigned labels are semantically appropriate and no missing label is identified. Rejected cases are revised at the generation-code level and re-audited before inclusion.

3.5 Dataset Statistics

The final DATAPRISM release contains 1,362 dirty datasets across 8 domains (Appendix Table 4). It

Table 2: Overall evaluation results for 10 models on the DATAPRISM benchmark (1,362 dirty datasets, v1–v8 full set). All values are percentages; sorted in descending order by Overall. Bold indicates the column-wise best.

Model	Hit Rate	Precision	Recall	F1	Exact Match	Overall
GPT-5.4	54.7	18.5	39.0	22.7	0.3	25.9
Claude Opus 4.6	**68.9**	**22.1**	**53.5**	**27.0**	**2.8**	**32.9**
Gemini 3 Pro	66.4	13.7	50.7	19.9	0.0	28.8
Kimi K2.5	57.6	17.4	40.9	22.0	0.2	26.6
DeepSeek-V4-Pro	64.1	11.1	47.3	16.6	0.0	26.9
DeepSeek-V4-Flash	58.3	10.7	42.5	15.8	0.0	24.7
Qwen3.5-397B-A17B	46.0	9.0	30.6	12.8	0.0	19.6
Qwen3.5-122B-A10B	50.7	8.4	33.9	12.4	0.0	21.0
Qwen3.5-35B-A3B	47.3	13.8	31.2	17.2	0.0	21.5
Qwen-Plus	42.6	11.3	26.8	14.3	0.0	19.0

includes 768 single-label instances, 505 two-to-five-label multi-label instances, and 89 five-plus-label scenario stacks. The single-label portion is balanced across the three injection intensity tiers, with 256 instances for each of T1, T2, and T3. The benchmark covers 40 diagnostic labels across heterogeneous tabular domains, including healthcare, finance, transportation, education, retail, human resources, sensor/IoT, and credit application scenarios. All datasets are reviewed by 5 statistics experts, with a 96.6% approval rate before release. These statistics are reported separately to distinguish benchmark generation from final release composition.

3.6 Evaluation Protocol

At evaluation time, each model receives the complete dirty dataset in CSV format and the complete definition library for all 40 labels. No clean baseline, injection metadata, audit output, or ground-truth label count is provided. The model is asked to return a ranked list of at most twelve candidate labels with supporting evidence; for scoring, we discard confidence scores and ranks and convert the output to an unordered set $\hat{Y}$. Invalid labels outside $\mathcal{L}$ are ignored. If no valid label can be parsed, $\hat{Y}$ is treated as the empty set. All models are evaluated under the same zero-shot prompt template and output parser.

3.7 Metrics

All metrics are computed at the instance level and then macro-averaged over N datasets, following example-based multi-label evaluation (Zhang and Zhou, 2014). For instance i, let $\hat{Y}_i$ be the predicted set and Y_i^* be the gold set. Precision is defined as

$$P_i = \begin{cases} \frac{|\hat{Y}_i \cap Y_i^*|}{|\hat{Y}_i|}, & |\hat{Y}_i| > 0, \\ 0, & |\hat{Y}_i| = 0, \end{cases} \tag{3}$$

and recall is $R_i = |\hat{Y}_i \cap Y_i^*|/|Y_i^*|$. The instance-level F1 is

$$F1_i = \begin{cases} \frac{2P_iR_i}{P_i+R_i}, & P_i + R_i > 0, \\ 0, & P_i + R_i = 0. \end{cases} \tag{4}$$

We also report Hit Rate,

$$\frac{1}{N}\sum_i \mathbf{1}[\hat{Y}_i \cap Y_i^* \neq \emptyset], \tag{5}$$

which measures whether the model detects at least one true defect, and Exact Match,

$$\frac{1}{N}\sum_i \mathbf{1}[\hat{Y}_i = Y_i^*], \tag{6}$$

which measures complete recovery of the gold set. The Overall score is the arithmetic mean of Hit Rate, macro-F1, and Exact Match. These three quantities capture complementary levels of diagnostic success: partial detection, set-level quality, and complete localization. We also report precision and recall to expose over- and under-prediction behavior.

4 Experiments

4.1 Baseline Models

We evaluate 10 representative large language models spanning different vendors and parameter scales: Claude Opus 4.6 (Anthropic, 2026), Gemini 3 Pro (Google DeepMind, 2025), GPT-5.4 (OpenAI, 2026), DeepSeek-V4-Pro and DeepSeek-V4-Flash (DeepSeek, 2026), Kimi K2.5 (Moonshot AI, 2026), Qwen3.5-35B-A3B, Qwen3.5-122B-A10B, Qwen3.5-397B-A17B, and Qwen-Plus (Alibaba Qwen Team, 2026). All models have context windows of at least 1M tokens and each receives the dirty dataset CSV together with the complete definitions of all 40 candidate labels as input, generating predicted label sets in a zero-shot setting (see Appendix D).

Table 3: HR, F1, and EM for each model on single-label (N=768), 2–5-label (N=505), and 5+-label (N=89) dirty datasets. All values are percentages.

Model	Single (N=768)			2–5-Label (N=505)			5+-Label (N=89)		
	HR	F1	EM	HR	F1	EM	HR	F1	EM
Claude Opus 4.6	**56.9**	**21.6**	**4.2**	82.8	**31.6**	**1.2**	94.4	**48.2**	0.0
Gemini 3 Pro	51.4	13.1	0.0	**84.0**	26.8	0.0	96.6	39.0	0.0
DeepSeek-V4-Pro	48.0	10.2	0.0	83.0	23.4	0.0	95.5	32.7	0.0
Kimi K2.5	41.8	15.8	0.1	74.7	28.4	0.4	**97.8**	40.0	0.0
GPT-5.4	40.5	17.7	0.4	69.9	27.4	0.2	91.0	39.7	0.0
DeepSeek-V4-Flash	41.8	9.8	0.0	76.4	21.7	0.0	**97.8**	33.6	0.0
Qwen3.5-35B-A3B	32.3	12.3	0.0	62.0	22.1	0.0	93.3	32.0	0.0
Qwen3.5-122B-A10B	34.8	8.3	0.0	67.7	16.3	0.0	91.0	26.1	0.0
Qwen3.5-397B-A17B	32.7	9.3	0.0	59.4	15.8	0.0	85.4	26.1	0.0
Qwen-Plus	28.4	10.2	0.0	57.0	19.0	0.0	83.1	23.3	0.0

4.2 Overall Results

As shown in Table 2, Claude Opus 4.6 ranks first on all six metrics with an Overall score of 32.9%, yet all models fall within the narrow band of 19.0%–32.9%, indicating a systematic challenge faced by all 10 tested LLMs rather than a clear qualitative separation between models.

Exact Match Reveals the Fundamental Bottleneck. Exact Match Reveals the Fundamental Bottleneck. The EM of every model approaches zero (highest: 2.8% by Claude Opus 4.6), exposing a structural gap between detecting at least one signal and fully recovering the ground-truth label set. This difficulty is further amplified by the absence of explicit supervision on label cardinality, causing models to frequently over-predict or under-predict the number of labels required for each dataset. All models accurately match locally observable surface patternssuch as missing rate, value range, and duplicate countbut fail to reconstruct the multi-column joint structure required for mechanism-level labels such as heteroscedasticity and label noise, consistently driving EM near zero.

Prediction Styles and Scale Paradox. Claude Opus 4.6 achieves the highest Precision (22.1%), Recall (53.5%), and Overall (32.9%) simultaneously, representing a balanced prediction style; Gemini 3 Pro, DeepSeek-V4-Pro, and DeepSeek-V4-Flash represent high-recall expansive predictors; GPT-5.4, Kimi K2.5, and the Qwen family tend toward relatively conservative predictions. The three Qwen3.5 variants all score below 22% in Overall, with performance negatively correlated with model scale (Qwen3.5-397B-A17B $<$ Qwen3.5-122B-A10B $\approx$ Qwen3.5-35B-A3B), demonstrating that the bottleneck resides not in parameter count but in whether the training regime develops distributional reasoning capabilities.

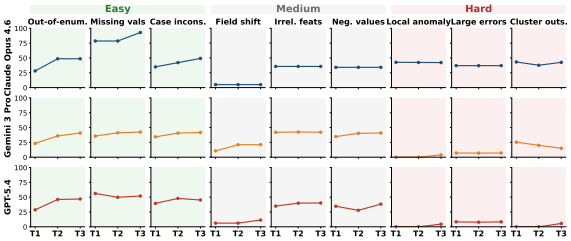

Figure 3: Small-multiples Overall plots for Claude Opus 4.6, Gemini 3 Pro, and GPT-5.4 (rows) $\times$ 9 representative labels (columns: 3 Easy / 3 Medium / 3 Hard) across T1/T2/T3. Easy/Medium/Hard is based on average Overall; labels are chosen to avoid all-zero trajectories when possible.

4.3 Single-Label vs. Multi-Label Analysis

As shown in Table 3, model hit rate increases with label count—single-label (28%–57%) to 2–5-label (57%–84%) to 5+-label (83%–98%)—because multi-label instances exhibit richer signals and any correct identification counts as a hit. Single-label hit rates therefore provide a more faithful reflection of diagnostic capability. The EM decreases monotonically from 4.2% on single-label instances to 1.2% on 2–5-label instances and 0.0% on 5+-label instances, confirming that the diagnostic bottleneck lies in complete problem localization rather than partial detection.

4.4 Injection Intensity Analysis

Each single-label dataset is provided in three injection intensity versions: T1 (weak, minimum margin above the audit threshold), T2 (moderate, typical real-world magnitude), and T3 (strong, clearly visible signals). For row-level injections, T1/T2/T3 correspond to approximately 0.5%/2%/5% of affected rows; for distribution-level injections, the intensity is expressed as a shift magnitude of $0.5\sigma/0.7\sigma/1.2\sigma$.

Figure 3 shows that stronger injection generally yields higher Overall, but responses remain label-dependent: easy labels approach perfect detection even at T1, whereas hard labels stay near 0% across all intensities, indicating that the bottleneck lies in recognizing global structural patterns rather than in signal strength.

4.5 Fine-Grained Analysis across 40 Labels

Figure 4 presents per-label Overall performance on single-label instances, revealing three natural difficulty tiers. Easy labels ($\geq$30%; e.g., missing entries, case inconsistency, out-of-range values) possess signals directly observable from raw data without requiring cross-column reasoning.

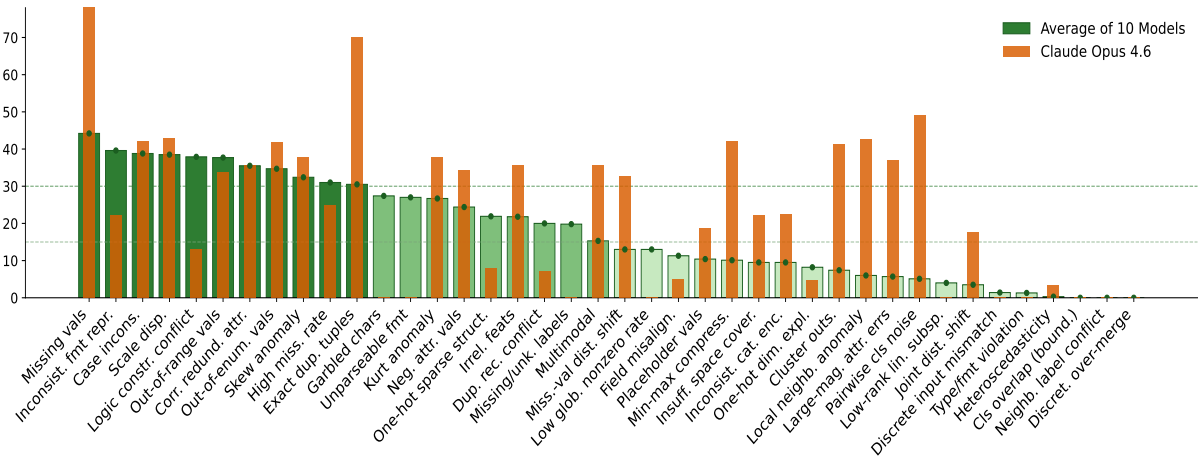

Figure 4: Per-label Overall for the 40 formal labels on single-label dirty datasets. Labels are sorted by average Overall across 10 models and overlaid with Claude Opus 4.6. Colors denote ■ Easy (≥30%), ■ Medium (15%–30%), and ■ Hard (<15%).

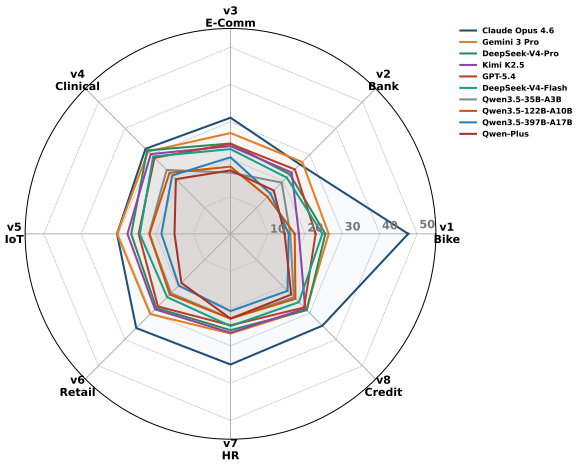

Figure 5: Overall radar chart for 10 models across the 8 domains (v1–v8).

Medium labels (15%–30%; e.g., irrelevant features, duplicate conflicts, multimodality) require cross-column comparison but retain readable local signals. Hard labels (<15%; e.g., one-hot dimensionality explosion, cluster outliers, distribution shift, heteroscedasticity) demand global distributional understanding and remain near zero for most models, reflecting a fundamental limitation of current LLMs in inferring multi-column joint structure. Models exhibit differentiated strengths on specific labels: Claude Opus 4.6 effectively detects pairwise class-label noise; DeepSeek-V4-Pro leads on one-hot dimensionality explosion, suggesting model selection in practice should be task-specific rather than uniform.

4.6 Domain-Level Analysis

Figure 5 reveals relatively stable performance across the 8 domains, with most models showing a cross-domain coefficient of variation below 15%, indicating that diagnostic capability is driven more by label difficulty than by domain-specific knowledge. Higher-dimensional datasets v3–v8 consistently yield higher Overall than v1–v2, because more columns provide richer observable signals for problem localization. Taken together, these patterns suggest that effective LLM-based data preprocessing systems should combine semantic interpretation with dedicated statistical modules to bridge the symptom-mechanism gap. DATAPRISM provides a principled evaluation platform for such systems through its fine-grained label taxonomy and three-tier intensity design.

5 Conclusion

We introduced DATAPRISM, a benchmark for fine-grained data quality diagnosis with 40 formally defined labels and 1,362 verified dirty datasets across 8 domains and 3 injection intensities. Evaluation of 10 representative LLMs reveals a clear symptom–mechanism gap: surface-symptom labels achieve hit rates above 80% while mechanism-level labels fall below 20%, with the best Overall score at 32.9% and exact match of 2.8%. Bridging this gap requires moving beyond column-level pattern matching toward multi-column joint structure reasoning, and hybrid architectures combining LLM semantic interpretation with dedicated statistical modules are a promising direction. DATAPRISM provides a principled testbed through its fine-grained taxonomy and three-tier intensity design.

Limitations

This work has the following primary limitations. First, all dirty datasets are constructed through controlled injection, which provides precise semantic labels but may not fully capture the complexity of composite, context-dependent data quality problems that arise naturally in real-world settings. Second, the label space is limited to 40 labels; although these cover the major operationalizable problem types catalogued in the textbook, the full space of data quality problems is broader. Third, the current evaluation uses a fixed closed-set, zero-shot, CSV-input protocol; models equipped with statistical tools or code execution may behave differently.

Ethical Statements

All datasets in DATAPRISM are synthetically constructed from publicly described statistical distributions (Gaussian, uniform) and publicly available dataset schemas (Bike Sharing, Bank Marketing, UCI data repository, etc.). No real personal information is included. All baseline datasets are generated with deterministic random seeds, ensuring full reproducibility. This research aims to improve the measurability and reliability of automated data quality diagnosis systems and does not involve the identification of personal sensitive information or high-risk automated decision-making.

References

Alibaba Qwen Team. 2026. Qwen3.5. https://qwen.ai/research.

Anthropic. 2026. Claude opus 4.6. https://www.anthropic.com/news/claude-opus-4-6.

Tommaso Bendinelli, Artur Dox, and Christian Holz. 2025. Exploring LLM agents for cleaning tabular machine learning datasets. *ArXiv preprint*, abs/2503.06664.

Xu Chu, Ihab F. Ilyas, Sanjay Krishnan, and Jiannan Wang. 2016. Data cleaning: Overview and emerging challenges. In *Proceedings of the 2016 International Conference on Management of Data, SIGMOD Conference 2016, San Francisco, CA, USA, June 26 - July 01, 2016*, pages 2201–2206. ACM.

DeepSeek. 2026. DeepSeek-V4: Towards highly efficient million-token context intelligence. https://www.deepseek.com.

Meihao Fan, Ju Fan, Nan Tang, Lei Cao, Guoliang Li, and Xiaoyong Du. 2024. AutoPrep: Natural language question-aware data preparation with a multi-agent framework. *ArXiv preprint*, abs/2412.10422.

Meihao Fan, Ju Fan, Yuxin Zhang, Shaolei Zhang, Xiaoyong Du, Jie Song, Peng Li, Fuxin Jiang, Tieying Zhang, and Jianjun Chen. 2026. DeepPrep: An LLM-powered agentic system for autonomous data preparation. *ArXiv preprint*, abs/2602.07371.

Salvador García, Julián Luengo, and Francisco Herrera. 2015. *Data Preprocessing in Data Mining*, volume 72 of *Intelligent Systems Reference Library*. Springer, Cham.

Google DeepMind. 2025. Gemini 3 Pro. https://blog.google/products-and-platforms/products/gemini/gemini-3/.

Qian Huang, Jian Vora, Percy Liang, and Jure Leskovec. 2024. MLAgentBench: Evaluating language agents on machine learning experimentation. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 20271–20309. PMLR.

Yuhang Lai, Chengxi Li, Yiming Wang, Tianyi Zhang, Ruiqi Zhong, Luke Zettlemoyer, Scott Wen-tau Yih, Daniel Fried, Sida Wang, and Tao Yu. 2023. DS-1000: A natural and reliable benchmark for data science code generation. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 18319–18345. PMLR.

Lan Li, Liri Fang, Bertram Ludäscher, and Vetle I. Torvik. 2024. AutoDCWorkflow: LLM-based data cleaning workflow auto-generation and benchmark. *ArXiv preprint*, abs/2412.06724.

Moonshot AI. 2026. Kimi K2.5. https://github.com/MoonshotAI/Kimi-K2.5.

Wei Ni, Kaihang Zhang, Xiaoye Miao, Xiangyu Zhao, Yangyang Wu, and Jianwei Yin. 2024. IterClean: An iterative data cleaning framework with large language models. In *Proceedings of the ACM Turing Award Celebration Conference (ACM-TURC)*. ACM.

OpenAI. 2026. GPT-5.4. https://openai.com/index/introducing-gpt-5-4/.

Danrui Qi, Zhengjie Miao, and Jiannan Wang. 2024. CleanAgent: Automating data standardization with LLM-based agents. *ArXiv preprint*, abs/2403.08291.

Maojun Sun, Yifei Xie, Yue Wu, Ruijian Han, Binyan Jiang, Defeng Sun, Yancheng Yuan, and Jian Huang. 2026. DSAEval: Evaluating data science agents on a wide range of real-world data science problems. *ArXiv preprint*, abs/2601.13591.

Min-Ling Zhang and Zhi-Hua Zhou. 2014. A review on multi-label learning algorithms. *IEEE Transactions on Knowledge and Data Engineering*, 26(8):1819–1837.

A DATAPRISM Dataset Statistics

Table 4 presents basic statistics for each of the 8 benchmark versions, including dataset dimensions and the breakdown of dirty datasets by label count.

B Overview of All 40 Labels

The full set of 40 data quality labels is listed below. Each label is accompanied by a four-part formal mathematical definition (notation, mathematical definition, parameters and criteria, boundary and exclusions). To avoid table overflow in double-column format, we present the labels as a simple list.

1. **Incomplete annotation: missing or unknown labels**
   Some sample labels are missing or marked as unknown values, with labeled and unlabeled samples mixed together.

2. **High-dimensional sparsity: high-cardinality one-hot sparse structure**
   After one-hot encoding a high-cardinality attribute, each row contains only a tiny number of ones, with nearly all other entries being zero.

3. **Dirty data: garbled characters outside allowed character set**
   Text fields contain large numbers of characters outside the allowed character set, appearing as garbled or unreadable symbol strings.

4. **Missing values: high missing rate**
   The overall cell-level missing proportion in the dataset is substantially high, reducing the density of usable information.

5. **Dirty data: unparseable field format**
   Field contents cannot be parsed according to the expected type or format; type conversion or format validation fails.

6. **Missing values: presence of missing entries**
   At least one cell in the data matrix has an unobserved or unrecorded value, represented as NA, NULL, or equivalent.

7. **Data redundancy: exact duplicate tuples**
   The dataset contains two or more records that are identical across all field values.

8. **Missing value noise effect: distributional deviation**
   Missing values cause the observable feature distribution to systematically deviate from the true complete distribution.

9. **Outliers: distance-based cluster outliers**
   A small number of samples lie considerably farther from the identified main clusters, falling outside them.

10. **Class overlap: boundary neighborhood mixing**
    In the local neighborhood near class boundaries, nearest neighbors from different classes coexist, making class assignment unstable.

11. **Dataset distribution shift: joint distribution shift**
    The joint distribution of features and labels undergoes an overall shift across data partitions, leading to unstable cross-partition evaluation.

12. **Erroneous attribute values: type or format constraint violation**
    Field values do not conform to the data type or format constraints specified by the field definition.

13. **Non-normal distribution: skewness anomaly**
    The sample distribution is skewed to one side, with a long tail in the histogram and systematic deviation in the Q-Q plot.

14. **Erroneous attribute values: out-of-enumeration values**
    An enumeration field contains values not in the allowed set, typically resulting from encoding or data entry errors.

15. **Non-normal distribution: kurtosis anomaly**
    The peak or tails of the distribution deviate significantly from normality, exhibiting leptokurtic, platykurtic, or heavy-tailed shapes.

16. **Incomplete attributes: placeholder values**
    Some samples have attributes filled with "don't care" or "not applicable" placeholders that are non-empty in form but semantically invalid.

17. **Non-normal distribution: multimodality**
    The empirical distribution of samples exhibits two or more distinct peaks, violating the unimodal normal assumption.

18. **Attribute noise: large-magnitude value errors**
    Observations of a numerical attribute exhibit anomalous jumps that deviate far from the true value with substantially larger error magnitude.

19. **Homoscedasticity violation: heteroscedasticity**
    The variances of group subsets are inconsistent, violating the equal-variance assumption required by parametric tests.

20. **Class-label noise: pairwise class noise**
    Label errors concentrate between one specific pair of classes, forming a systematic bidirectional mutual mislabeling pattern.

21. **Data inconsistency: logical constraint conflicts**
    Combinations of values across different records for the same entity violate business or data logic constraints, producing mutual contradictions.

22. **Low-dimensional manifold: global low-rank linear subspace**
    The primary variation in high-dimensional data concentrates along a small number of directions, exhibiting an overall low-rank structure.

23. **Redundant attributes: highly correlated features**
    The dataset contains pairs of attributes that are statistically highly correlated with heavily overlapping information content.

Table 4: Basic statistics of DATAPRISM benchmark datasets across 8 domains.

Version	Domain	Rows	Cols	Total	Single	2–5-Label	5+-Label
v1	Bike Sharing	3,100	11	54	54	0	0
v2	Bank Marketing	1,970	23	176	138	38	0
v3	E-Commerce	554	105	162	96	66	0
v4	Clinical	395	109	194	96	81	17
v5	IoT/Sensor	380	104	194	96	80	18
v6	Retail	333	124	194	96	80	18
v7	HR	335	118	194	96	80	18
v8	Credit App	334	119	194	96	80	18
Total				**1,362**	**768**	**505**	**89**

24. **High-cardinality variables: one-hot dimensionality explosion**
One-hot encoding of a discrete variable adds a large number of binary feature columns, causing the feature space to expand dramatically.

25. **Duplicate record conflict: conflicting field values**
Among multiple duplicate records for the same entity, the same field contains different versions of the value.

26. **Irrelevant or redundant features: irrelevant features**
The dataset contains features with no actual association with the target variable, contributing only dimensional redundancy.

27. **Synonymous values: inconsistent format representation**
The same semantic value appears in multiple different encodings or formats, causing errors in deduplication and matching.

28. **Data errors: values outside the reasonable range**
A numerical field contains obviously unreasonable values that exceed business rules or physically feasible ranges.

29. **Non-standard representation: case inconsistency**
Semantically identical strings in the same field appear in multiple forms due to differences in letter case.

30. **Data errors: field misalignment (column swap)**
A field is filled with values that properly belong to a different field, causing a mismatch between the field name and its content.

31. **Outlier-induced scaling bias: min-max interval compression**
After min-max normalization, extreme values compress most normal samples into a very small interval.

32. **Data errors: inconsistent categorical encoding**
A categorical field contains values outside the legal set, or the same category is represented using inconsistent encodings.

33. **Excessive attribute scale disparity: single-attribute dominance**
Without scale alignment, attributes with large magnitudes dominate the distance or inner-product measure.

34. **Noisy instances: majority-label conflict in neighborhood**
A sample's class label is inconsistent with the label held by the majority of its local nearest neighbors.

35. **Limited sample size: insufficient input space coverage**
Training samples cover only a narrow portion of the feature space, leaving obvious unsampled regions.

36. **Noisy instances: anomalous local neighborhood distance**
Distances between a sample and its neighbors are significantly larger than expected, exhibiting a locally sparse structure in feature space.

37. **Non-positive attribute values: presence of negative values**
Observations of a continuous attribute include negative numbers, violating the strictly positive domain ($x > 0$) requirement.

38. **Continuous attribute mismatch: incompatibility with discrete input**
A high-cardinality numerical feature has too many distinct values to be treated by algorithms as a finite enumerable discrete input.

39. **High-dimensional sparsity: extremely low global non-zero rate**
The vast majority of entries in the sample-feature matrix are zero, resulting in an extremely low overall non-zero rate.

40. **Discretization over-merging: inconsistent same-value mapping**
After discretization, samples with the same value are assigned to different bins, violating the principle that equal values should map to the same bin.

C Sample Formal Label Definition

We illustrate the four-part formal definition structure through the label "Missing Values: Presence of Missing Entries" (CH4_S4_4_010).

Notation and Objects: $\mathcal{D}$: dataset ($N \times p$ matrix); x_{ij}: value at row i, column j; $\mathrm{isnull}(x_{ij})$: missing-value detection predicate.

Mathematical Definition: $\mathrm{HasMissing}(\mathcal{D}) = 1 \iff \exists\, i \in \{1, \ldots, N\},\ j \in \{1, \ldots, p\}\colon \mathrm{isnull}(x_{ij}) = 1$

Parameters and Criteria: The sole required parameter is the implementation of $\mathrm{isnull}(\cdot)$ (e.g., `pd.isna()` in Python). The trigger condition is the existence of at least one missing value in the data matrix.

Boundary and Exclusions: This label covers only the binary determination of whether any missing value exists and does not characterize the missing rate or the missing mechanism (MCAR/MAR/MNAR). Whether empty strings are treated as missing depends on the configuration of $\mathrm{isnull}(\cdot)$ and must be fixed prior to implementation.

D Evaluation Prompt Template

All evaluated models receive an identical system prompt. The following presents a summary of the core content (the full version includes detailed descriptions and a JSON schema for all 40 labels):

> *You are a data quality diagnosis evaluation model for the LFMD benchmark.*
>
> *Your task is to select 1–12 labels from the 40 candidate labels that best match the current dataset's problems, ranked by degree of match.*
>
> *Matching principles: (1) Judge based on the dirty dataset itself; do not infer from the filename. (2) Focus on observable primary defect semantics; multiple labels may be selected simultaneously. (3) Retain a weak side-effect label only if it constitutes an independent problem.*
>
> *Input: dirty dataset CSV and complete definitions of all 40 candidate labels (name and formal mathematical definition). No clean baseline is provided.*
>
> *Output: a JSON structure containing selected_labels (label ID + confidence + evidence), decision_trace (3–6 reasoning steps), candidate_assessment (comparisons of at least 3 candidates), and uncertainty (uncertainty assessment).*

E Sample Ground Truth for Dirty Datasets

The following example illustrates the structure of `output_gt.json` for a single-label dataset:

```
{"label_schema_version": "v1.0",
 "dataset_type": "single_label",
 "applied_labels": [{
   "label_id": "CH4_S4_4_010_0010_v3_008_31",
   "name": "Missing Values: Presence of
            Missing Entries",
   "tier": "T1",
   "injected_columns": ["balance"],
   "template": "MISSING",
   "params": {"fraction": 0.005}
 }]}
```

In multi-label datasets, `applied_labels` contains multiple entries, with each entry operating on a distinct feature column. When injecting label A incidentally triggers label B, B is marked with `"source": "alignment_model_consensus"`.

F Sample Model Output

The following presents an abbreviated example of a prediction output by GPT-5.4 for a single-label dataset:

```
{"selected_labels": [{
   "label_id": "CH4_S4_4_010_0010_v3_008_31",
   "name": "Missing Values: Presence of
            Missing Entries",
   "confidence": 0.97,
   "evidence": "The balance column contains
     approximately 0.5% NaN values; no other
     column has missing values."
 }],
 "decision_trace": [
   "Summary statistics show a missing rate of
    0.47% in the balance column; all other
    columns have zero missing values.",
   "The missing distribution does not follow
    an MCAR pattern; systematic missingness
    is suspected.",
   "Out-of-enumeration values, duplicate rows,
    and case inconsistencies are not detected."
 ],
 "candidate_assessment": [
   "High missing rate (CH4_S4_4_006): missing
    rate < 1%, below the 30% threshold.
    Excluded.",
   "MNAR effect (CH4_S4_6_018): no
    distributional deviation signal detected.
    Excluded."
 ],
 "uncertainty": "low"}
```

G Three-Layer Agent QC System

The DATAPRISM construction pipeline employs a three-layer agent QC system to automatically inspect the injection quality of each dirty dataset.

Layer 1: Code Audit Agent

System prompt summary:

> You are the code audit agent for the LFMD project, specializing in analyzing the injection logic of data quality injection recipe scripts. Task: identify code issues that may cause injection to fail, including template-column type incompatibility, parameters causing no-ops, and logic branches causing silent skips. Output: `code_verdict` (FAIL/RISK/PASS) + `risks` list + `code_summary`.

Sample input (injection code snippet):

```
def inject_missing(df, tier, rng):
    col = 'balance'
    n = int(len(df) * 0.005)  # T1
    idx = rng.choice(df.index, n,
                     replace=False)
    df.loc[idx, col] = np.nan
```

```
    return df,
      [{'template':'MISSING','column':col}]
```

Sample output:

```
{"code_verdict": "PASS",
 "risks": [],
 "code_summary": "MISSING template operates
   on balance (float64), fraction=0.005;
   logic is correct with no silent-skip
   risk."}
```

Layer 2: Dataset Audit Agent

System prompt summary:

> You are the dataset audit agent for the LFMD project. Task: receive the raw statistics computed by the statistical layer, interpret their meaning, and determine whether the dirty dataset injection is correct and effective. The label definition is the ground truth. Output: `data_verdict` (PASS/WARN/PARTIAL/FAIL) + `anomalies` + `key_findings` + `data_summary`.

Sample input (statistical layer output):

```
{"label": "CH4_S4_4_010", "tier": "T1",
 "clean": {"balance": {"missing": 0.0,
   "mean": 1428.3}},
 "dirty": {"balance": {"missing": 0.0047,
   "mean": 1421.6}},
 "delta": {"balance":
   {"missing_delta": +0.0047}}}
```

Sample output:

```
{"data_verdict": "PASS",
 "anomalies": [{"column": "balance",
   "type": "missing", "rate": 0.0047,
   "injection_verified": true}],
 "key_findings": "balance column missing
   rate 0.47%, injection verified; no
   anomalies in the remaining 19 columns.",
 "data_summary": "Single-label MISSING
   injection passed."}
```

Layer 3: QC Agent

System prompt summary:

> You are the QC agent for the LFMD project. Task: receive the two independent audit reports from Code Audit and Dataset Audit, cross-validate them, locate the root cause, and issue a final verdict. Output: `final_verdict` + `root_cause_type` + `root_cause_detail` + `fix_suggestion`.

Sample input (both reports merged, defect case):

```
{"code_audit":
   {"code_verdict":"PASS","risks":[]},
 "dataset_audit":
   {"data_verdict":"FAIL",
    "anomalies":[],
    "key_findings":"balance column missing
      rate 0.00%; injection did not take
      effect."}}
```

Sample output:

```
{"final_verdict": "FAIL",
 "root_cause_type": "injection_silent_skip",
 "root_cause_detail": "The injection function
   silently returns an empty list at tier=T1
   because fraction=0.003 < min_threshold,
   leaving the dataset unmodified.",
 "fix_suggestion": "Raise the T1 fraction
   to 0.005 or remove the threshold guard
   logic."}
```