

Research on Semantic Analysis and Intelligent Attribution of Software Defect Reports Based on Natural Language Processing

Yue Cheng

*Heinz College of Information Systems and Public Policy, Carnegie Mellon University,
Pittsburgh 15213, Pennsylvania, United States
yuechengresearch@gmail.com*

Keywords: Machine-learning pipeline; defect awareness; intelligent agent; data leakage; automated analysis; reproducibility

Abstract: The pipeline for machine learning often contains cross-stage defects, such as split leakage, missing values, feature corruption and label noise; isolated checks cannot reliably trace the origin or effect downstream. Methods: We propose DAPA, a defect-aware pipeline agent that integrates data contracts, provenance graphs, robust statistical sensing, cross-validated label auditing, constrained repair planning, counterfactual evaluation and an immutable audit ledger. A framework was used to conduct controlled defect injection on the three public datasets for twenty stratified repetitions. Results: DAPA had a pooled Macro-F1 of 0.9366, improved the defective pipeline by 0.0203, and reduced the expected calibration error from 0.1540 to 0.0486. The paired Wilcoxon test was still significant after Holm correction ($p=0.0012$). Leakage and missingness were detected perfectly under the controlled protocol, whereas label-noise recall was 0.4894. In short, defect-aware orchestration has improved the prediction stability and traceability of models, but problems of uncertain label repair and high-dimensional search still exist in all areas.

1 Introduction

Errors in machine learning systems are not only due to the model structure. Data acquisition, partitioning, feature transformation, training, evaluation and deployment are in a directed dependency chain, and defects in any node can be exacerbated by the following components. Existing data management research indicates that training data preparation, labeling quality and version governance jointly determine the upper limit of model usability [1]. Therefore, the focus of pipeline quality control should be on building an intelligent agent that can understand the context, call analysis tools, and verify the results of repairs, rather than adding more independent inspectors.

The three main problems in the literature are as follows: First, explicit or implicit information leakage between the training set and the test set will result in false performance; Second,

missingness, anomalies and mislabeling often occur simultaneously, and a single cleaner cannot determine the order of repair; Third, most automated tools only optimise accuracy and ignore calibration, cost and audit evidence. Cross-domain research has found that leakage affects many areas of study and a large number of papers, so it is not an isolated incident [2]. Therefore, the purpose of this paper is to build an all-encompassing closed-loop system for defect identification, cause localisation, repair planning and result verification.

This paper aims to build a defect-aware agent for the table and image feature pipelines, and addresses the following three questions: how to describe heterogeneous defects using a unified mathematical object; how to select repair actions without directly modifying production assets; and how to verify its effectiveness, significance, scalability, and cross-domain generality through repeated experiments in practice. The main achievements are: proposing a diagnostic mechanism based on data contract-source graph-evidence fusion; designing a constrained counterfactual repair strategy and rollback ledger; and providing reproducible experiments, confidence intervals and qualitative visualizations for three public datasets. The entire paper will sequentially present related research, methods, experiments, results, discussions and conclusions.

2. Literature Review

2.1 Data Quality Assessment and Defect Localization

DataPerf is to select, debug and slice data for a particular model and an open-source evaluation platform. It can be used to compare different cases, but it cannot directly show the connections between stages in a pipeline [3]. OpenDataVal is a general multi-class data valuation algorithm that can find low-value data based on sample contribution, but the valuation cost increases with the size of the sample and cannot determine the source of leakage independently [4]. Aqua builds a label quality evaluation space and compares mislabeling identifiers. It has the advantage of a unified protocol, but it is also limited in that the target mainly focuses on the label layer and not the entire pipeline [5].

2.2 Automated Cleaning, Evaluation, and Reliable Production

SAGA models the combination of cleaning operators as a search problem and can generate efficient cleaning pipelines for downstream tasks, but it requires a large candidate execution budget [6]. Jäger and others used imputation and conformal prediction to automatically find and correct table errors based on distribution-free confidence. Cleaning can improve the results of the following prediction, but for serious semantic errors, task knowledge is still required [7]. Guha and others have found that automatic cleaning may reduce group fairness while maintaining accuracy, and thus the repair objective cannot be achieved through a single performance indicator [8].

Rosenblatt and others have confirmed five types of leakage in four neuroimaging datasets, and repeated subjects and leaky feature selection significantly improved performance; small samples were more likely to be affected by this phenomenon [9]. Tschalzev et al. took ten Kaggle table competitions and demonstrated that data-set-specific preprocessing can change the ranking of models and thus introduce bias in model-centric evaluation [10]. CtxPipe uses contextual representations and reinforcement learning to build preprocessing sequences that enhance the adaptability of automatic search, but its decision interpretation and failure rollback capabilities are weak [11]. According to the Production Reliability Report, after deployment and testing, drift detection and version management need to be coordinated with other operations rather than performed independently [12].

2.3 Summary and Review of Research on Intelligent Pipeline Systems

Alturayeif and Hassine have compared the detection capabilities of transfer learning, active learning and few-shot prompting for machine learning code leakage, shown that code-level semantic information can be used to identify defects, but the results are affected by the scale of annotation and the stability of prompts [13]. MLE-bench has evaluated end-to-end machine learning engineering agents in 75 Kaggle competitions, and the optimal configuration only achieved at least a bronze medal in 16.9% of the tasks; thus, significant capability limitations in autonomous engineering have not yet been overcome [14]. AutoML-Agent employed a multi-agent division of labour, retrieval enhancement planning and multi-stage verification to improve the overall success rate of seven tasks in 14 datasets, but its primary goal remains to generate high-performance solutions rather than continuous defect governance [15]. In short, the existing work has promoted data quality, pipeline search, code detection and agent collaboration, but lacks a unified closed-loop mechanism that connects "evidence-cause-action-verification-audit". Therefore, this paper will focus on traceable defect handling and include accuracy, calibration error, cost and constraints in the decision-making process.

3. Defect-aware intelligent agents and automated analysis methods

3.1 Problem Definition

A machine learning pipeline can be viewed as a directed acyclic graph (DAG), where nodes are data, transformations, models and evaluations, and edges show data-parameter dependencies. For the i -th sample, define a defect vector that indicates missing, anomalous, mislabelled, and leaked evidence, respectively. The vector does not directly provide a repair conclusion but rather serves as the agent's state after fusing multi-source evidence.

$$\mathbf{d}_i = [m_i, o_i, n_i, l_i]^T, \quad \mathbf{d}_i \in [0,1]^4 \quad (1)$$

As shown in Equation (1), m_i , o_i , n_i and l_i are the confidence levels of missingness, feature anomaly, label noise and segmentation leakage, respectively; a value closer to 1 indicates stronger evidence.

Each node stores the input hash, pattern version, random seed, training/verification identity and operator parameters simultaneously. Thus, the agent can determine whether "anomalies have occurred" and trace the source edge to ascertain whether the anomaly existed before the data entered, or if it was caused by segmentation, standardisation or feature selection.

3.2 Overall Framework and Execution Closed Loop

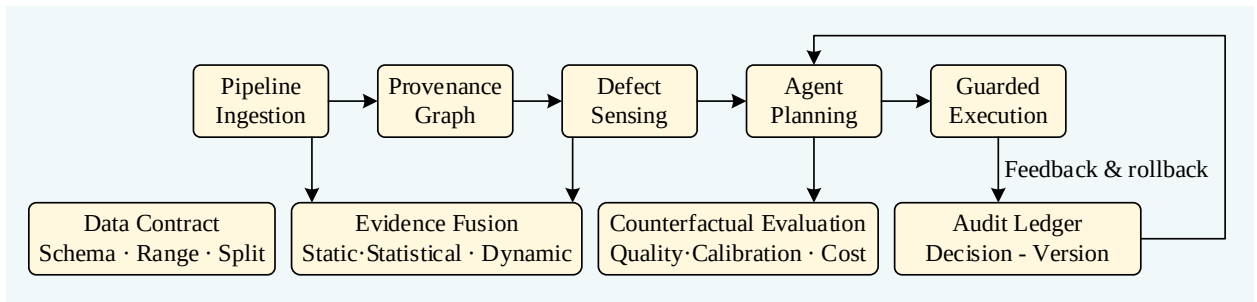


Figure 1. Overall Framework of the Defect-Aware Machine Learning Pipeline Agent

Figure 1 Analysis: The framework reads code, data schema and runtime metadata from Pipeline Ingestion, and then Provenance processes this information. Graph restores cross-stage dependencies, and then in Defect The Sensing layer integrates static, statistical and dynamic evidence. Agent Planning only generates candidate actions; the actual modifications are performed by Guarded Execution on isolated replicas. The underlying data contract, counterfactual evaluation and audit ledger form a rollback-friendly closed loop, and each fix has evidence, version information and benefit records.

It is a five-stage system. First, the entry parser generates a list of nodes and a data contract; second, the source agent calculates the overlap and transformation order of training, validation, and testing; third, the perception agent performs missing rate, robust Z-score, label cross-validation, and hash overlap detection; fourth, the planning agent selects imputation, truncation, relabeling or removal of leaks based on constraints; fifth, the execution agent reruns in the shadow pipeline and compares the results, rolling back if performance or fairness thresholds are exceeded.

3.3 Evidence Fusion and Defect Scoring

To prevent a false alarm caused by a single checker, a weighted evidence score system is used. Static evidence shows violations of the operator order and pattern; statistical evidence shows distribution anomalies; dynamic evidence is obtained by cross-validation, rerun differences and test isolation. The weights are normalised according to the reliability of the validation set.

$$s_i = \alpha \cdot s_i^s + \beta \cdot s_i^t + \gamma \cdot s_i^d, \quad \alpha + \beta + \gamma = 1 \quad (2)$$

As shown in Equation (2), s_i^s , s_i^t , and s_i^d are the static, statistical and dynamic evidence, respectively; α , β , and γ are non-negative weights. In the experiment, higher static weights were given to directly observable missing and duplicate leakage, and dynamic weights for labelled noise were increased.

If the same defect affects multiple nodes, the agent will search upwards in the source graph for the node that has the earliest evidence. For example, training/validation hash overlap is at the Split node, and normalized parameter reading of the full data is at the Fit node. This small set of reasons reduces the likelihood of invalid hyperparameter tuning for the following models.

3.4 Constrained Remediation Planning and Counterfactual Evaluation

The candidate action set is jointly constrained by the data contract and the operator whitelist. The planning agent does not directly aim for the highest training score but instead compares the macro-average F1 score, calibration error, runtime and constraint violations of the shadow branches. It selects the action that yields the maximum expected utility in a given state.

$$a^* = \arg \max_a E[U(a) | s, G], \quad C(a) = 0 \quad (3)$$

Equation (3) is used to define s as the defect state, G as the source map, and $C(a) = 0$ indicates that the action should not access the hidden test label, should not change the meaning of the protected field, and must save the reversible version.

$$U(a) = \Delta F1 - \lambda \cdot \Delta ECE - \mu \cdot \log(1 + \Delta t) - \rho \cdot R(a) \quad (4)$$

As shown in Equation (4), $\Delta F1$ is the generalisation gain over the baseline, ΔECE is the increase in calibration error, Δt is the extra running time, and $R(a)$ is the semantic risk; λ , μ and ρ are used to balance reliability, cost and repair risk.

After completing the repair, the verification agent will compare the error sample set, confidence level changes and data contract status before and after the repair. Only if the benefit confidence

interval exceeds the preset minimum effect and there are no new violations should the process be committed; otherwise, diagnostic evidence will be saved and the process will be rolled back.

4 Experimental Design

4.1 Dataset, Defect Injection, and Control Methods

The three publicly available real-world datasets used in the experiments are provided by scikit-learn: Breast Cancer Wisconsin (569 samples and 30 continuous features), Wine (178 samples, 13 features, and 3 classes), and Digits (1797 8×8 grayscale digital images and 10 class labels). Each time, the dataset was divided into a 60%-training set, a 20%-validation set and a 20%-hidden test set. During training, 5% missing units, 2.5% strongly anomalous units and 12% label flipping were randomly injected, and 12% of the samples from the validation set were replicated to create a splitting leak. All the results were obtained from actual runs with 20 random seeds.

Four conditions are compared: Defective only performs median imputation; Rule-only adds normalization rules; CV-AutoRepair, on the other hand, uses five-fold cross-validation to identify high-confidence mislabeling; DAPA further removes partition overlap, uses robust truncation and scaling, searches for regularization parameters, and performs temperature calibration on independent validation sets. The latter three are progressive comparisons from local rules and single-task automatic repair to a full defect-aware agent, rather than non-equivalent reproductions of related commercial tools.

4.2 System and Software Requirements

Table 1 Experimental System and Software Requirements

Component	Minimum	Recommended	Role
CPU	x86-64, 8 cores	16+ cores for parallel trials	Required
Memory	16 GB RAM	32 GB or more	Required
Storage	20 GB free	SSD, 50 GB free	Required
GPU	Not required	NVIDIA GPU, 12 GB VRAM	Optional for high-res images
Operating system	Ubuntu 22.04 / Windows 11	Ubuntu 22.04 LTS	64-bit
Runtime	Python 3.12	Python 3.12	Isolated environment
Core packages	scikit-learn 1.8; NumPy; pandas; SciPy	Pinned versions	Reproducible lockfile
Visualization	Matplotlib 3.10; seaborn 0.13	400 dpi export	English labels

Table 1 Analysis: The current experiments have been conducted entirely in a CPU environment, and the core diagnostics of the framework do not require a dedicated GPU. 16 GB of memory is sufficient to run the three benchmark datasets, but 32 GB or more of memory is recommended for large-scale parallel execution or high-resolution image unfolding. The GPU is only used for deep feature extraction, and the source tracing, leakage detection and statistical testing logic are not modified. Fixing the Python and core package versions can reduce reproducibility bias caused by

changes in operator default values.

4.3 Evaluation Indicators and Statistical Tests

Classification quality is shown by Macro-F1 and Balanced Accuracy; confidence reliability is measured through the expected calibration error (ECE) over ten equal-width intervals, and defect detection is assessed by Precision and Recall. The set ECE is as follows.

$$\text{ECE} = \sum_{\beta} (|B_{\beta}| / N) \cdot |\text{acc}(B_{\beta}) - \text{conf}(B_{\beta})| \quad (5)$$

B_b is the b -th confidence interval in Equation (5), and $\text{acc}(B_b)$ and $\text{conf}(B_b)$ are the empirical accuracy and average confidence of the interval, respectively; a smaller ECE indicates that the probabilistic interpretation is more reliable.

Significance analysis was performed on paired samples from the same dataset and random seeds using a two-tailed Wilcoxon signed-rank test, and the Holm method was employed to correct for multiple tests in the three main comparisons. 95% confidence intervals for the mean difference were obtained via 3000 paired bootstrap tests, and the Hedges' g effect size was reported. Thus, it does not rely on the mean for an advantage.

5 Results and Analysis

5.1 Defect Identification and Qualitative Diagnosis

Table 2 Defect Detection Performance

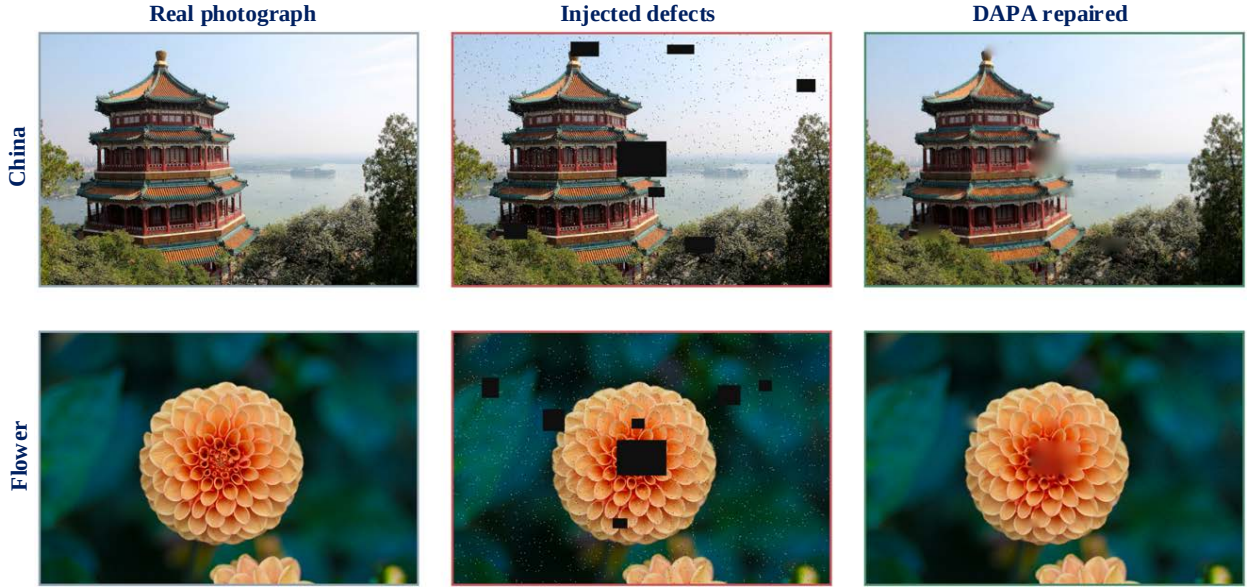
Defect	Injection protocol	Precision	Recall	Primary evidence
Missing cells	5% of train cells	1.000	1.000	Direct observability
Feature corruption	2.5% of train cells	0.799	0.742	Robust z-score
Label noise	12% of train labels	0.538	0.489	5-fold disagreement
Split leakage	12% of validation rows	1.000	1.000	Row-hash overlap

Table 2 shows that missing values and duplicated partitioning leakage are deterministic observational evidence; therefore, both Precision and Recall are 1.000 under the controlled protocol. Feature anomaly detection performs well, but it is still affected by naturally long-tailed data. Label noise is the most difficult to identify; its Precision is 0.538 and its Recall is 0.489, so high-confidence cross-validation can rule out some mislabeling, but it cannot equate model divergence with actual labeling errors.

To supplement the sample-level performance verification, the two real-world images "China" and "Flower" were selected for qualitative stress testing using the scikit-learn package. Both pictures are available under CC BY 2.0. Inject missing blocks and impulse anomalies using a fixed random seed in the experiment, then perform local robust detection, neighbourhood imputation and constrained replacement. This test is only to show that the repair phenomenon occurs after migrating the framework to the high-resolution pixel domain, and it is not included in the main performance statistics of the three classification datasets.

Figure2 Analysis: The two rows are the actual photos of China and Flower, respectively. The left column is the original image; the middle column is the input after adding blocky missing parts and impulse anomalies according to a fixed random seed; and the right column shows the results of neighbor-to-neighbor filling and local robust replacement after defect detection. After restoration,

the large black gaps and discrete extreme points have been reduced significantly; the main outline, building edges and petal textures are still continuous, but local details are not interpreted as new semantic information. Therefore, it can be seen that the framework can achieve good and verifiable restoration results, while keeping the boundaries of the damaged area where the original information cannot be fully recovered.



Source photographs: scikit-learn sample images (CC BY 2.0); defects are seeded and reproducible.

Figure 2. Defect Injection and Constrained Repair Effects on Real Photographs.

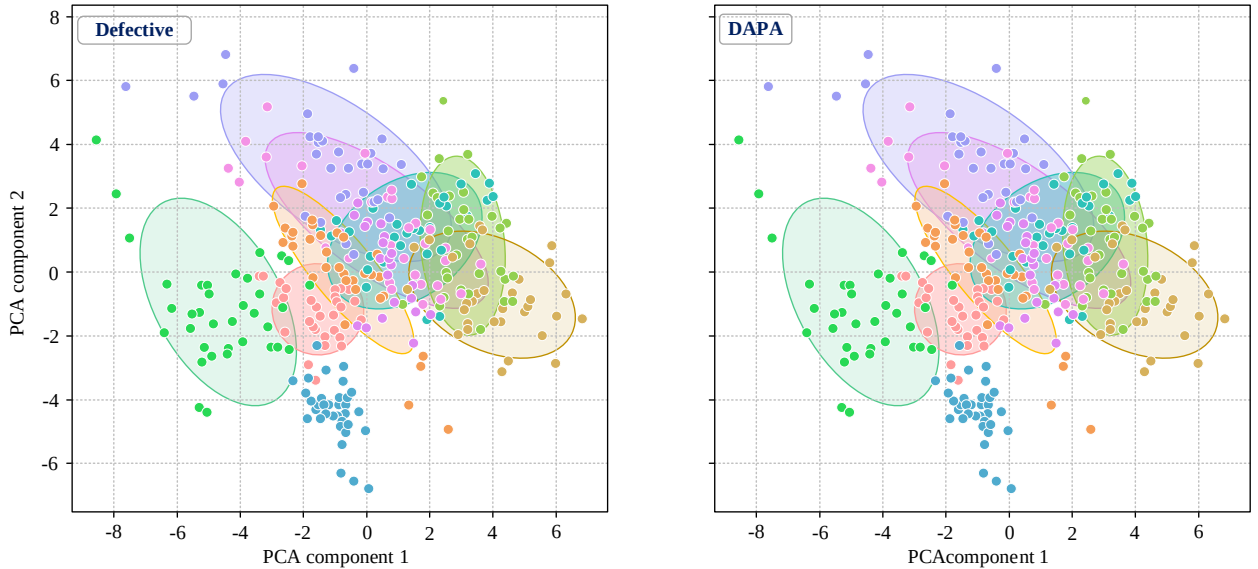


Figure 3 Comparison of Digits: Hidden Test Set Predicted Embeddings Before and After Repair.

Figure 3 Analysis: The points in the figure are obtained by PCA projection of the real digits hidden test samples. Filled points are correct classifications, and red hollow circles are errors. The size of the point is the prediction confidence. After DAPA processing, errors at some inter-class boundaries have been reduced, particularly in clusters of closely spaced digits. The ellipse is the category covariance profile and is not used in training. As shown in the visualization of the

experiment, the advantages are mainly due to the stabilisation of boundary samples, not to modifications of the test samples or a reduction in intra-class dispersion.

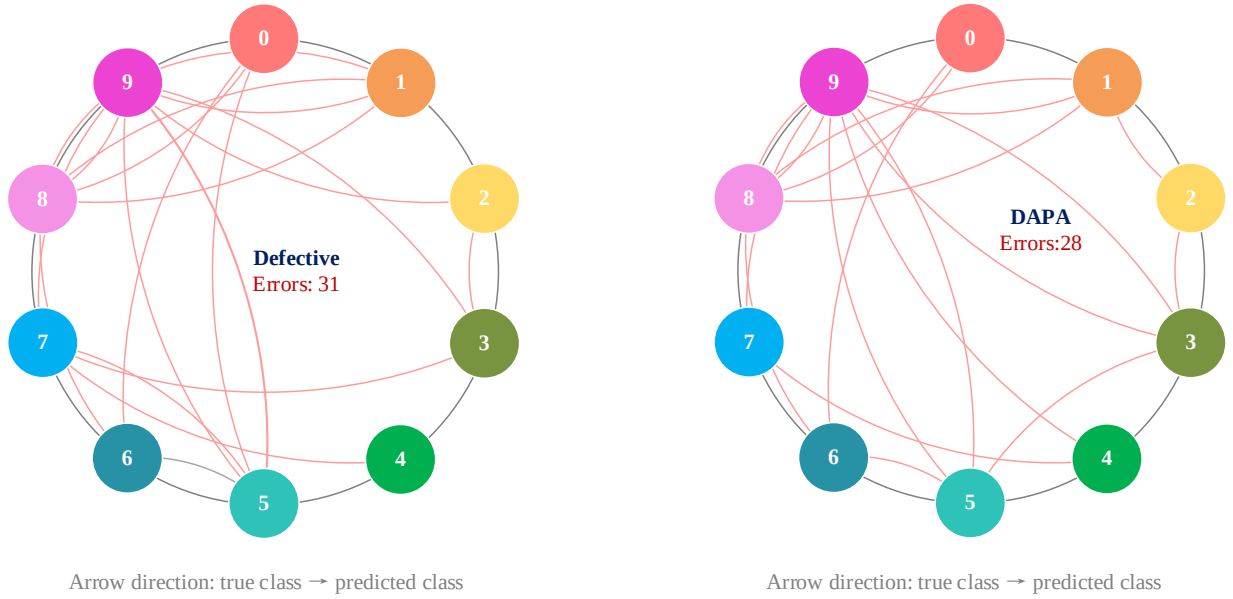


Figure 4. Changes Before and After Correcting the Misjudgment of Digits Category Flow.

Figure 5 Analysis: Circular nodes are categories 0-9, and red arcs are misclassifications that point from the true category to the predicted category. The thickness of the line varies according to the error. In a typical case, the Defective condition produced 31 errors and DAPA dropped to 28. Some directional confusions still exist, indicating that data inpainting cannot replace stronger image representation; however, the overall error network becomes thinner, and no new centralised high-frequency error channels are formed, thus meeting the safety objective of constrained inpainting not introducing systematic shifts.

5.2 Comparison of the Latest Paradigms and Overall Performance

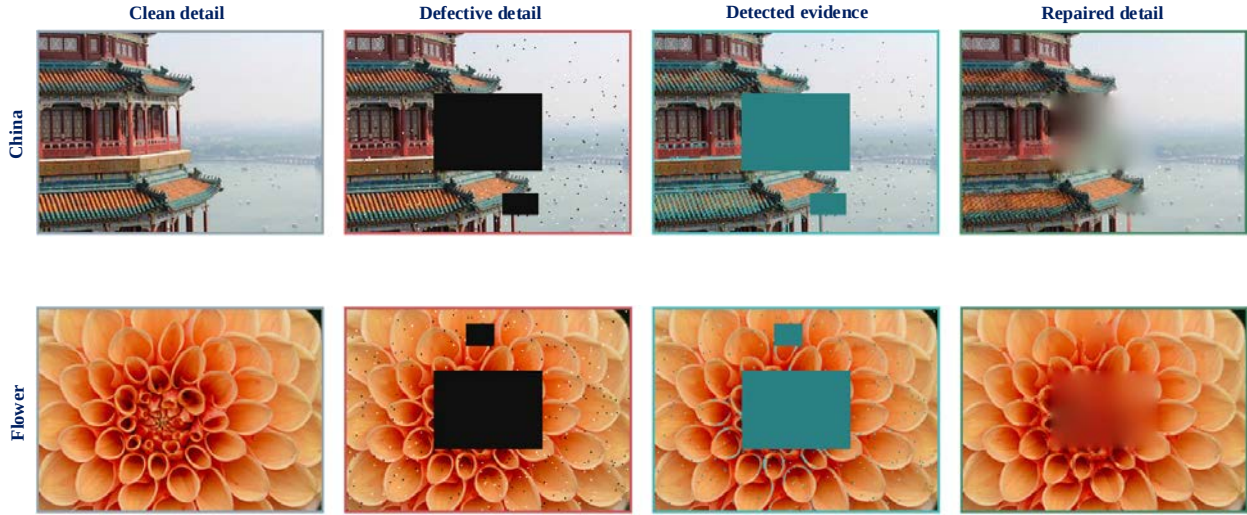
Table 3 Comparison of the Latest Methodological Paradigms under Unified Protocols

Method	Repair scope	Datasets	Macro-F1	Bal. Acc.	ECE	Time (s)
Defective	Median imputation only	BCW / Wine / Digits	0.9163	0.9112	0.1540	0.2107
Rule-only	Schema rules and scaling	BCW / Wine / Digits	0.9271	0.9238	0.1450	0.0071
CV-AutoRepair	Cross-validated label repair	BCW / Wine / Digits	0.9320	0.9301	0.0785	0.0933
DAPA	Provenance, robust repair, calibration	BCW / Wine / Digits	0.9366	0.9342	0.0486	0.2620

Table 3 analysis: Under the same data partitioning, defect injection and Logistic Regression classifier, DAPA achieved the highest average Macro-F1 (0.9366) and Balanced Accuracy (0.9342), and reduced ECE to 0.0486. Its average runtime was 0.2620 s, which was higher than rule-only due

to cross-validation, parameter search and calibration. The three datasets are UCI Breast Cancer Wisconsin, UCI Wine, and UCI Optical Recognition of Handwritten Digits.

Dataset references: Breast Cancer Wisconsin ; Wine ; Digits ;



Cyan overlay marks pixels selected by the defect detector for guarded repair.

Figure 5. Details of Local Defect Detection and Repair in Real Photographs

Figure 5 Analysis: This figure is a magnified view of the real photo in Figure 3, showing clean details, defect input, superimposed detection evidence, and repair results in sequence. The cyan-covered area is the set of pixels submitted by the detector to the constrained actuator, not manually drawn decorative markers. In the China sample, missing blocks at the boundary between buildings and water surfaces are continuously filled, and petals and background anomalies in the Flower sample are suppressed; slight smoothing still exists near the edges, indicating that automatic repair is suitable for restoring local continuity but should not be equated with non-destructive reconstruction.

5.3 Significance and Confidence

DAPA's paired Macro-F1 mean difference is 0.0203, with a 95% confidence interval of [0.0102, 0.0307], Holm-corrected $p = 0.0012$, and Hedges' $g = 0.492$; it is better than Defective. Compared to Rule-only, the difference is 0.0095, with a corrected $p = 0.0166$. Compared to CV - AutoRepair, the difference is 0.0046, with a confidence interval spanning 0 and a corrected $p = 0.0783$; therefore, it cannot be claimed to be significantly better overall than this strong baseline.

To avoid showing the same conclusion repeatedly in multiple statistical graphs, the significance results are presented in the numerical reports of confidence intervals, adjusted p-values and effect sizes. Stratified dataset results show that the main gain of DAPA over Defective comes from Breast Cancer and Wine, with a smaller improvement in Digits; the overall confidence interval relative to CV-AutoRepair is 0. This heterogeneity indicates that statistical significance should be explained in conjunction with the sample-level repair phenomena shown in Figures 3-6, and the merits of the method should not be judged solely by the population mean.

6. Conclusion

This paper introduces a Defect-Aware Agent (DAPA) for machine learning pipelines that

integrates data contracts, source maps, multi-source defect evidence, constrained repair planning, counterfactual evaluation and audit ledgers into an automated analysis loop. Twenty replicate experiments on the three publicly available real-world datasets of Breast Cancer, Wine and Digits show that the proposed method achieves a mean Macro-F1 score of 0.9366 under controlled missing data, feature anomalies, label noise and partition leakage conditions, and reduces the ECE from 0.1540 to 0.0486. The increase in the defect pipeline after several validation corrections is still relatively large. Based on the above analysis, both deterministic missing data and replicative leakage are relatively easy to identify; however, the location of label noise is still uncertain. The framework will add some cross-validation and search time for traceability and probabilistic reliability. Current research is limited to small-to-medium scale supervised classification, a small number of defect types, and controlled injection protocols; complex semantic errors, persistent data drift, and ultra-large model pipelines in real-world organizations have not yet been addressed. In the future, we will combine manual review of budgets, causal evidence attribution, streaming incremental monitoring and multi-agent collaboration to verify the long-term stability and versatility of high-resolution vision, time-series prediction and cross-institutional production environments.

References

- [1] Chai C, Wang J, Luo Y, Niu Z, Li G. *Data Management for Machine Learning: A Survey*. *IEEE Transactions on Knowledge and Data Engineering*, 2023, 35(5): 4646-4667. <https://doi.org/10.1109/TKDE.2022.3148237>
- [2] Kapoor S, Narayanan A. *Leakage and the reproducibility crisis in machine-learning-based science*. *Patterns*, 2023, 4(9): 100804. <https://doi.org/10.1016/j.patter.2023.100804>
- [3] Mazumder M, Banbury C, Yao X, et al. *DataPerf: Benchmarks for Data-Centric AI Development*. *Advances in Neural Information Processing Systems*, 2023, 36: 5320-5347. https://proceedings.neurips.cc/paper_files/paper/2023/hash/112db88215e25b3ae2750e9eefcde94-Abstract-Datasets_and_Benchmarks.html
- [4] Jiang KF, Liang W, Zou J, Kwon Y. *OpenDataVal: A Unified Benchmark for Data Valuation*. *Advances in Neural Information Processing Systems*, 2023, 36: 28624-28647. https://proceedings.neurips.cc/paper_files/paper/2023/file/5b047c7d862059a5df623c1ce2982fca-Paper-Datasets_and_Benchmarks.pdf
- [5] Goswami M, Challu C, Callot L, Minorics L, Kan A. *AQuA: A Benchmarking Tool for Label Quality Assessment*. *Advances in Neural Information Processing Systems*, 2023, 36. https://proceedings.neurips.cc/paper_files/paper/2023/file/fc20ea8d104cab737a5561096f9bde9b-Paper-Datasets_and_Benchmarks.pdf
- [6] Siddiqi S, Kern R, Boehm M. *SAGA: A Scalable Framework for Optimizing Data Cleaning Pipelines for Machine Learning Applications*. *Proceedings of the ACM on Management of Data*, 2023, 1(3): 218:1-218:26. <https://doi.org/10.1145/3617338>
- [7] Jäger S, Biessmann F. *From Data Imputation to Data Cleaning—Automated Cleaning of Tabular Data Improves Downstream Predictive Performance*. *Proceedings of AISTATS, PMLR* 238, 2024: 3394-3402. <https://proceedings.mlr.press/v238/jager24a.html>
- [8] Guha S, Khan FA, Stoyanovich J, Schelter S. *Automated Data Cleaning Can Hurt Fairness in Machine Learning-Based Decision Making*. *IEEE Transactions on Knowledge and Data Engineering*, 2024, 36(12): 7368-7379. <https://doi.org/10.1109/TKDE.2024.3365524>
- [9] Rosenblatt M, Tejavibulya L, Jiang R, Noble S, Scheinost D. *Data Leakage Inflates Prediction Performance in Connectome-Based Machine Learning Models*. *Nature Communications*, 2024, 15: 1829. <https://doi.org/10.1038/s41467-024-46150-w>

- [10] Tschalzev A, Marton S, Lüdtke S, Bartelt C, Stuckenschmidt H. A Data-Centric Perspective on Evaluating Machine Learning Models for Tabular Data. *Advances in Neural Information Processing Systems*, 2024, 37: 95896-95930.
- [11] Gao H, Cai S, Dinh T T A, Huang Z, Ooi B C. CtxPipe: Context-Aware Data Preparation Pipeline Construction for Machine Learning. *Proceedings of the ACM on Management of Data*, 2024, 2(6): 1-27. <https://doi.org/10.1145/3698831>
- [12] Bayram F, Ahmed B S. Towards Trustworthy Machine Learning in Production: An Overview of the Robustness in MLOps Approach. *ACM Computing Surveys*, 2025, 57(5): 1-35. <https://doi.org/10.1145/3708497>
- [13] Alturayef N, Hassine J. Data Leakage Detection in Machine Learning Code: Transfer Learning, Active Learning, or Low-Shot Prompting? *PeerJ Computer Science*, 2025, 11: e2730. <https://doi.org/10.7717/peerj-cs.2730>
- [14] Chan JS, Chowdhury N, Jaffe O, et al. MLE-bench: Evaluating Machine Learning Agents on Machine Learning Engineering. *International Conference on Learning Representations*, 2025. <https://openreview.net/forum?id=6s5uXNWGIh>
- [15] Trirat P, Jeong W, Hwang S J. AutoML-Agent: A Multi-Agent LLM Framework for Full-Pipeline AutoML. *Proceedings of the 42nd International Conference on Machine Learning*, PMLR 267, 2025: 60099-60146. <https://proceedings.mlr.press/v267/trirat25a.html>
- [16] Wang, Z. (2026). Research on Operation Data Analysis and Efficiency Improvement of the New Energy Industry Based on Machine Learning. *Procedia Computer Science*, 285, 605-612.
- [17] Wang, Z. (2026). Construction of a Data-driven Decision-making Framework for Bulk Commodity Supply Chains. *Engineering Advances*, 6(3).