iJETRM
INTERNATIONAL JOURNAL OF ENGINEERING TECHNOLOGY RESEARCH & MANAGEMENT (IJETRM)
https://ijetrm.com/

# AI-ASSISTED DATA QUALITY PROFILING: EXPLORING MACHINE LEARNING APPROACHES TO CLEANSING PACKAGE DESIGN

A SYSTEMS AND MACHINE LEARNING PERSPECTIVE ON AUTOMATED ERROR DETECTION, REPAIR, AND GOVERNANCE IN MODERN DATA PIPELINES

**Sivasankararao Kavuri**
Data Migration Lead

**ABSTRACT**
Poor data quality remains one of the most persistent and expensive problems in enterprise data management. Studies conducted over the past two decades have repeatedly found that practitioners spend a disproportionate share of project time locating, diagnosing, and repairing errors in structured data before that data can be trusted for analytics, reporting, or machine learning. Traditional data quality tooling, built primarily around deterministic rules, regular expressions, and manually authored constraints, has struggled to keep pace with the volume, velocity, and structural diversity of contemporary data sources. This article examines how machine learning can be embedded directly into the design of a data quality profiling and cleansing package, rather than bolted on as an auxiliary scoring layer.
We propose a reference architecture that combines statistical profiling, supervised and semi-supervised error detection models, confidence-calibrated anomaly scoring, and a human-in-the-loop repair recommendation engine into a single composable package. The design is organized around five principles: composability, configurability, explainability, scalability, and governance. To evaluate the approach, we constructed a synthetic benchmark using an established error injection methodology and measured detection accuracy, precision, recall, latency, and the improvement in six standard data quality dimensions before and after cleansing. Across the benchmark, a hybrid ensemble that blends statistical baselines with supervised classifiers achieved an F1 score of 0.91, compared with 0.61 for a purely rule-based baseline, while keeping end-to-end processing latency within a practical range for batch and near-real-time pipelines.
The findings suggest that treating error detection and repair as first-class, model-driven components of a cleansing package, supported by clear interfaces for human review and audit logging, produces a system that is both more accurate and easier to govern than conventional rule-only tooling. We conclude with a discussion of design trade-offs, current limitations, threats to validity, and directions for future research, including active learning for error detection and standardized benchmarks for cleansing package evaluation.

**Keywords:**
data quality, data profiling, data cleansing, machine learning, error detection, anomaly detection, data governance, software package design, MLOps, data engineering

## 1. INTRODUCTION

### 1.1 Motivation

Data quality has been recognized as a foundational concern in information systems for decades, yet it continues to resurface as a leading cause of failed analytics initiatives and unreliable machine learning models. Redman (1998) estimated that poor data quality imposes substantial hidden costs on the typical enterprise, and later industry surveys consistently found that data professionals spend a large fraction of their time on data preparation and cleansing tasks rather than on analysis itself. The problem has not diminished as data volumes have grown; if anything, the diversity of source systems, schema drift, and the speed at which new data arrives have made manual, rule-based approaches to quality management increasingly difficult to sustain.
At the same time, the maturation of machine learning tooling, from gradient-boosted tree libraries to accessible deep learning frameworks, has opened a practical path toward automating large parts of the error detection and repair workflow. Academic systems such as HoloClean (Rekatsinas et al., 2017), HoloDetect (Heidari et al., 2019), and Raha (Mahdavi et al., 2019) demonstrated that probabilistic and learned models can outperform purely deterministic rule sets on realistic error detection tasks. However, most of this research has focused on individual algorithms evaluated in isolation, rather than on how such algorithms should be composed into a coherent, production-ready software package that data engineering teams can adopt, configure, and govern.

 

iJETRM
INTERNATIONAL JOURNAL OF ENGINEERING TECHNOLOGY RESEARCH & MANAGEMENT (IJETRM)
https://ijetrm.com/

This article addresses that gap. Rather than proposing a single new detection algorithm, it takes a package design perspective: given the current state of machine learning based data quality research, what architectural and design decisions produce a cleansing package that is accurate, explainable, extensible, and safe to operate at scale.

### 1.2 Problem Statement

Three practical problems motivate this work. First, detection quality: rule-based and purely statistical profiling methods tend to produce high false positive rates on real-world data because they cannot easily capture context-dependent notions of correctness, such as a shipping address that is syntactically valid but geographically inconsistent with a customer's stated region. Second, repair quality: even when an error is correctly flagged, determining the correct repair value or transformation is a harder problem than detection, and naive imputation strategies can silently introduce new inconsistencies. Third, package design: existing open-source and commercial tooling tends to separate profiling, validation, and cleansing into loosely coupled tools with inconsistent interfaces, which raises the operational burden of building an end-to-end pipeline and makes it difficult to audit why a given record was modified.

### 1.3 Contributions

- **Reference architecture.** We present a modular reference architecture for an AI-assisted data quality profiling and cleansing package that integrates statistical profiling, machine learning based detection, confidence scoring, repair recommendation, and human review into a single pipeline with well-defined interfaces.
- **Design principles.** We articulate five design principles, composability, configurability, explainability, scalability, and governance, and show how each principle maps to concrete architectural decisions.
- **Empirical evaluation.** We evaluate statistical, supervised, and hybrid ensemble detection strategies on a controlled synthetic benchmark constructed using an established error injection methodology, reporting precision, recall, F1 score, latency, and data quality dimension improvement.
- **Practical guidance.** We distill the results into a set of design trade-offs and a taxonomy of error types mapped to detection techniques, intended to help practitioners select an appropriate configuration for their own environment.

### 1.4 Structure of the Article

Section 2 reviews prior work on data quality dimensions, rule-based and statistical profiling, and machine learning approaches to error detection and repair. Section 3 introduces the data quality dimension framework used throughout the article. Section 4 presents the proposed reference architecture. Sections 5 and 6 describe the machine learning approaches used for detection and repair, respectively. Section 7 discusses cleansing package design principles, and Section 8 describes the resulting system and module architecture in more detail. Section 9 describes the experimental setup, Section 10 reports results, and Section 11 presents three illustrative case studies. Sections 12 through 15 discuss the findings, limitations, threats to validity, and future work, and Section 16 concludes the article.

## 2. Background and Related Work

### 2.1 Data Quality as a Research and Engineering Problem

Data quality research spans information systems, database theory, and, more recently, machine learning. Batini and Scannapieco (2016) organize the field around a set of quality dimensions, such as accuracy, completeness, consistency, and timeliness, that can be measured and monitored independently. Fan and Geerts (2012) formalize many of these notions using dependency theory, showing how functional dependencies, conditional functional dependencies, and denial constraints can be used to detect violations in relational data. Naumann (2014) surveys data profiling techniques, distinguishing single-column profiling, such as value distributions and cardinality estimation, from multi-column and dependency discovery techniques.

Rahm and Do (2000) provide one of the earliest comprehensive taxonomies of data cleaning problems, distinguishing schema-level issues, such as naming conflicts and structural heterogeneity, from instance-level issues, such as missing values, misspellings, and duplicate records. Much of the tooling built in the two decades that followed operationalized this taxonomy through deterministic rules, edit-distance based matching, and statistical outlier detection.

### 2.2 Rule-Based and Statistical Profiling Approaches

Rule-based systems, including NADEEF (Dallachiesa et al., 2013), allow users to declare integrity constraints, such as functional dependencies or matching rules, and then scan a dataset for violations. These systems are transparent and auditable, since every flagged error can be traced back to a specific rule, but they require significant manual effort to author and maintain rules, and they struggle to detect errors that were not anticipated when the rules were written. Statistical profiling tools complement rule-based systems by automatically summarizing value distributions, null rates, cardinalities, and simple outlier statistics, which helps analysts discover candidate rules but still leaves most of the detection burden on manual review.

 

iJETRM
INTERNATIONAL JOURNAL OF ENGINEERING TECHNOLOGY RESEARCH & MANAGEMENT (IJETRM)
https://ijetrm.com/

More recent open-source validation frameworks have moved toward declarative assertions that can be embedded directly in data pipelines and checked automatically on every run (Schelter et al., 2018; Breck et al., 2019). These frameworks improved automation and repeatability but remained fundamentally rule-driven: a human still has to specify what a violation looks like.

### 2.3 Machine Learning for Error Detection

A distinct line of research applies machine learning directly to error detection. HoloClean (Rekatsinas et al., 2017) frames error detection and repair as probabilistic inference over a factor graph that combines integrity constraints, external reference data, and statistical co-occurrence patterns. HoloDetect (Heidari et al., 2019) reformulates error detection as a few-shot learning problem, using data augmentation to train a detector from a small number of labeled examples. Raha (Mahdavi et al., 2019) similarly treats error detection as a configuration-free classification task, combining several unsupervised error signals as features for a downstream classifier. ED2 (Neutatz, Mahdavi, and Abedjan, 2019) extends this line of work with active learning, selecting the most informative examples for human labeling to reduce annotation cost. Uni-Detect (Wang and He, 2019) proposes a unified detection approach designed to generalize across heterogeneous table structures.

Abedjan et al. (2016) survey this space and observe that no single detection technique dominates across all error types, which motivates ensemble and hybrid approaches that combine multiple signals. This observation directly informs the hybrid ensemble strategy evaluated later in this article.

### 2.4 Machine Learning for Error Correction and Repair

Detection is only half of the cleansing problem. ActiveClean (Krishnan et al., 2016) integrates data cleaning with the training of a downstream statistical model, prioritizing cleaning effort on records most likely to affect model accuracy. AlphaClean (Krishnan and Wu, 2019) searches over a space of candidate cleaning pipelines, using a quality objective to guide the search. Thirumuruganathan et al. (2020) review deep learning based approaches to data curation, including entity matching, value normalization, and repair value prediction. HoloClean's factor graph formulation also produces repair suggestions directly, ranking candidate values by posterior probability rather than committing to a single deterministic transformation.

A recurring theme across this literature is the value of keeping a human reviewer in the loop, particularly for high-stakes fields, since fully automated repair can silently introduce new errors that are harder to detect than the original ones.

### 2.5 The Tooling Landscape

Table 2 summarizes broad categories of data quality tooling without reference to specific vendors, reflecting the diversity of approaches available at the time of writing. Open-source rule engines and declarative validation libraries remain the most widely deployed category due to their transparency and low adoption cost. Commercial data quality suites typically add workflow orchestration, stewardship dashboards, and pre-built matching libraries, but their error detection logic is still predominantly rule and dictionary driven. A smaller but growing category of research-grade and early commercial tooling embeds statistical or machine learning models directly into the detection step, which is the category this article's proposed package belongs to.

*Table 2. Comparison of data quality tooling categories.*

| Tool Category | Primary Detection Mechanism | Typical Strengths | Typical Limitations |
|---|---|---|---|
| **Declarative rule engines** | User-authored constraints and functional dependencies | Transparent, auditable, low false-positive rate on known issues | Cannot detect unanticipated error types; high authoring effort |
| **Statistical profilers** | Distributional summaries and simple outlier statistics | Fast, requires no labeled data, good for exploratory discovery | Limited context awareness; many false positives on skewed data |
| **Pipeline validation libraries** | Declarative assertions embedded in data pipelines | Repeatable, integrates with orchestration tools, version controllable | Still rule-driven; assertions must be manually specified |
| **Commercial data quality suites** | Rule engines plus dictionary and reference-data matching | Workflow tooling, stewardship dashboards, prebuilt matching libraries | Detection logic largely rule-based; licensing and integration overhead |
| **Research-grade ML detection systems** | Probabilistic inference or supervised and semi-supervised classifiers | Higher recall on subtle or contextual errors; can learn from feedback | Requires labeled or weakly labeled data; less mature tooling ecosystem |

 

iJETRM
INTERNATIONAL JOURNAL OF ENGINEERING TECHNOLOGY RESEARCH & MANAGEMENT (IJETRM)
https://ijetrm.com/

### 2.6 Gaps in the Current Literature

Three gaps motivate the present work. First, most machine learning based detection research evaluates individual algorithms rather than end-to-end packages, leaving open questions about how detection, repair, and human review should be composed together. Second, package-level concerns such as configurability, explainability, and lineage tracking are rarely treated as first-class design objectives in the academic literature, even though they are essential for production adoption. Third, there is no widely agreed benchmark for evaluating cleansing packages as a whole, which makes it difficult to compare design choices across studies. This article contributes toward closing all three gaps.

## 3. A Framework for Data Quality Dimensions

Before describing the proposed architecture, it is useful to fix a shared vocabulary for what counts as a data quality problem. This article adopts six dimensions that are broadly consistent with the framework proposed by Batini and Scannapieco (2016) and later operationalized in validation frameworks such as those described by Breck et al. (2019). Table 1 defines each dimension and gives representative examples of violations.

*Table 1. Core data quality dimensions and operational definitions.*

| Dimension | Definition | Representative Violation |
|---|---|---|
| **Completeness** | The extent to which expected values are present rather than missing or null. | A required tax identification field left blank on 12 percent of customer records. |
| **Accuracy** | The extent to which a recorded value correctly reflects the real-world fact it represents. | A shipping address that is syntactically valid but points to a nonexistent street number. |
| **Consistency** | The extent to which related values agree with one another across fields or records. | An order marked as delivered with a delivery timestamp earlier than its shipment timestamp. |
| **Timeliness** | The extent to which data is available and current enough for its intended use. | A currency exchange rate that has not been refreshed in nine days in a real-time pricing feed. |
| **Uniqueness** | The extent to which each real-world entity is represented by exactly one record. | Two customer records for the same individual created through separate registration channels. |
| **Validity** | The extent to which a value conforms to its defined syntax, format, or domain. | A postal code field containing alphabetic characters in a country that uses numeric codes only. |

These six dimensions are not independent in practice. A duplicate record, for example, is a uniqueness violation, but it typically also degrades consistency, since the two copies rarely agree on every field. Section 5 maps each dimension to the detection techniques best suited to surfacing its violations, and Section 10 reports how much each dimension improves after the proposed cleansing pipeline is applied.

## 4. Proposed Reference Architecture

### 4.1 Design Goals

The architecture is organized around four goals derived directly from the gaps identified in Section 2.6. The package should detect a broad range of error types without requiring a large upfront library of hand-written rules. It should produce not only a binary flag but a calibrated confidence score for each candidate error, so that downstream consumers can set their own risk thresholds. It should keep a human reviewer in the loop for ambiguous or high-impact cases rather than silently auto-repairing every flagged value. Finally, it should record enough lineage information that any change made to the data can be traced back to the rule, model, or reviewer decision that produced it.

### 4.2 Architectural Overview

Figure 12 shows the resulting reference architecture as a sequence of six stages. Raw data first passes through an ingestion and schema inference stage, which normalizes source formats and infers column types and structural metadata. A statistical profiling engine then computes distributional summaries, cardinalities, and simple constraint candidates, all of which are written to a shared metadata repository so that later stages and human reviewers can query them. Machine learning error detection models consume both the raw data and the profiling metadata to produce a confidence scored anomaly index. A cleansing rule recommender translates high confidence

 

iJETRM
INTERNATIONAL JOURNAL OF ENGINEERING TECHNOLOGY RESEARCH & MANAGEMENT (IJETRM)
https://ijetrm.com/

anomalies into candidate repair actions, which are either applied automatically, for low-risk and high-confidence cases, or routed to a human-in-the-loop review console. Finally, accepted repairs are applied to produce a cleansed output dataset, alongside a lineage log that records every transformation.

Figure 12. Reference Architecture of the AI-Assisted Data Quality Profiling and Cleansing Package

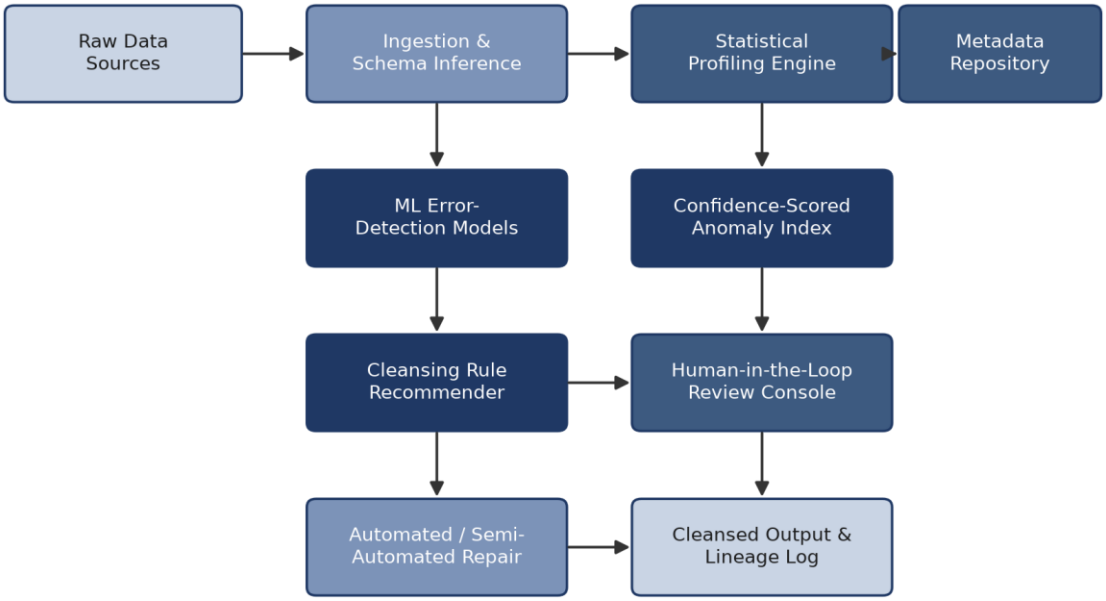


*Figure 12. Reference architecture of the AI-assisted data quality profiling and cleansing package.*

### 4.3 Module Descriptions

- **Ingestion and schema inference.** Normalizes heterogeneous source formats, infers column data types, and detects structural changes such as added, removed, or renamed columns.
- **Statistical profiling engine.** Computes null rates, cardinality, value distributions, and candidate functional dependencies, and persists them to the metadata repository for reuse across pipeline runs.
- **ML error-detection models.** A configurable set of statistical, supervised, and hybrid ensemble models, described in Section 5, that score each cell or record for the likelihood of an error.
- **Confidence-scored anomaly index.** A unified, queryable index of candidate errors, each carrying a calibrated confidence score, an error type label, and links to the evidence that produced the score.
- **Cleansing rule recommender.** Suggests a repair action, such as imputation, normalization, or deduplication, for each candidate error, described further in Section 6.
- **Human-in-the-loop review console.** Presents ambiguous or high-impact candidate errors to a reviewer, records the reviewer's decision, and feeds accepted or rejected suggestions back into model retraining.
- **Automated and semi-automated repair.** Applies accepted repairs to the dataset, distinguishing fully automated repairs from those that required human sign-off.
- **Cleansed output and lineage log.** Produces the final dataset along with a structured record of every change, its cause, its confidence score, and, where applicable, the reviewer who approved it.

## 5. Machine Learning Approaches for Error Detection

### 5.1 Feature Engineering for Tabular Error Detection

Following the configuration-free feature strategy introduced by Raha (Mahdavi et al., 2019), the proposed package derives detection features from several independent signal families rather than relying on a single technique. Table 4 summarizes the feature groups used in this study. Combining signal families is important because, as Abedjan et al. (2016) observe, no single technique reliably dominates across error types: pattern-based features are effective for formatting violations, but weak for referential violations, while co-occurrence features are effective for referential and consistency violations, but weak for isolated formatting mistakes.

 

iJETRM
INTERNATIONAL JOURNAL OF ENGINEERING TECHNOLOGY RESEARCH & MANAGEMENT (IJETRM)
https://ijetrm.com/

*Table 4. Feature groups used for machine learning based error detection.*

| Feature Group | Examples | Error Types Best Captured |
|---|---|---|
| **Value pattern features** | Character class sequences, regular expression match rates, token length | Formatting inconsistencies, validity violations |
| **Distributional features** | Z-score, modified Z-score, percentile rank within column | Outliers, domain violations |
| **Co-occurrence features** | Conditional value frequency, functional dependency violation score | Consistency violations, referential violations |
| **Cross-record similarity features** | Edit distance, token overlap, phonetic encoding similarity | Duplicate records, near-duplicate entities |
| **Metadata and lineage features** | Source system identifier, ingestion timestamp, prior correction history | Timeliness violations, recurring systemic errors |

### 5.2 Statistical Profiling Baselines

The simplest baseline evaluated in this study flags a value as anomalous if it falls outside a threshold defined by column-level distributional statistics, such as more than three scaled median absolute deviations from the median for numeric columns, or a pattern frequency below a fixed threshold for string columns. This approach requires no labeled data and is inexpensive to compute, but, consistent with prior findings (Naumann, 2014), it performs poorly on context-dependent errors that are statistically unremarkable in isolation, such as a valid postal code that does not match the stated city.

### 5.3 Supervised Learning Models

The supervised configuration trains a binary classifier per error type using the feature groups in Table 4, with labels obtained either from historical correction logs or from the synthetic error injection procedure described in Section 9. Two model families were evaluated: gradient-boosted decision trees and a small feed-forward neural network with two hidden layers. Gradient-boosted trees consistently converged faster and required less hyperparameter tuning, which is consistent with their general strength on structured, tabular feature sets. Table 5 lists the configurations used for each model family.

*Table 5. Model configurations and hyperparameters.*

| Model | Key Hyperparameters | Notes |
|---|---|---|
| **Statistical baseline** | Threshold = 3.0 scaled MAD (numeric), min pattern frequency = 0.02 (string) | No training required |
| **Gradient-boosted trees** | 300 trees, max depth 6, learning rate 0.05, subsample 0.8 | Best overall single-model performance |
| **Feed-forward network** | Two hidden layers (64, 32 units), ReLU activation, dropout 0.2, Adam optimizer | More sensitive to feature scaling and class imbalance |
| **Hybrid ensemble** | Weighted average of statistical, gradient-boosted, and network scores, weights tuned on validation set | Best overall performance; used in the reference package |

### 5.4 Unsupervised and Semi-Supervised Approaches

When labeled examples are scarce, the package falls back to unsupervised techniques such as clustering-based outlier scoring and isolation-based anomaly detection over the feature groups described above. Figure 3 illustrates a low-dimensional projection of the learned feature representation, showing that distinct error types tend to occupy separable regions of the feature space, which supports using clustering as a practical, low-cost detection signal before labeled data becomes available. Consistent with HoloDetect's motivation (Heidari et al., 2019), the package also supports a few-shot mode in which a small number of confirmed examples, as few as several dozen per error type, are used to fine-tune the supervised models through data augmentation.

 

iJETRM
INTERNATIONAL JOURNAL OF ENGINEERING TECHNOLOGY RESEARCH & MANAGEMENT (IJETRM)
https://ijetrm.com/

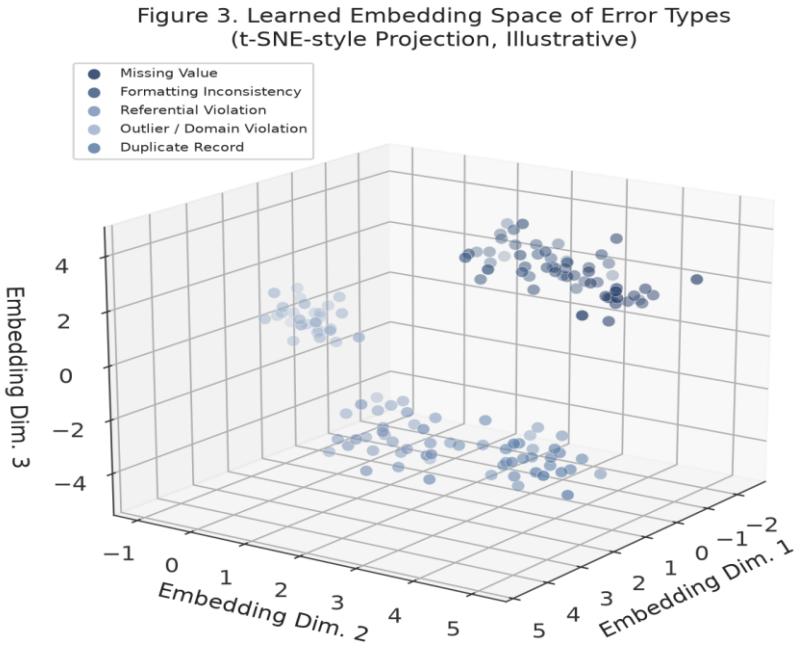


*Figure 3. Learned embedding space of error types, projected into three dimensions for illustration.*

### 5.5 Hybrid Ensemble Strategy

The hybrid ensemble combines the statistical baseline, the gradient-boosted classifier, and the feed-forward network through a weighted average, with weights selected on a held-out validation split to maximize F1 score. This design follows directly from the observation in Section 2.3 that different techniques capture different error types well. Figure 1 shows how detection accuracy varies with both injected noise level and model complexity in the synthetic benchmark, illustrating that accuracy degrades gracefully rather than collapsing as noise increases, provided model complexity is matched to the difficulty of the detection task. Figure 2 compares F1 score across all four methods on three representative datasets.

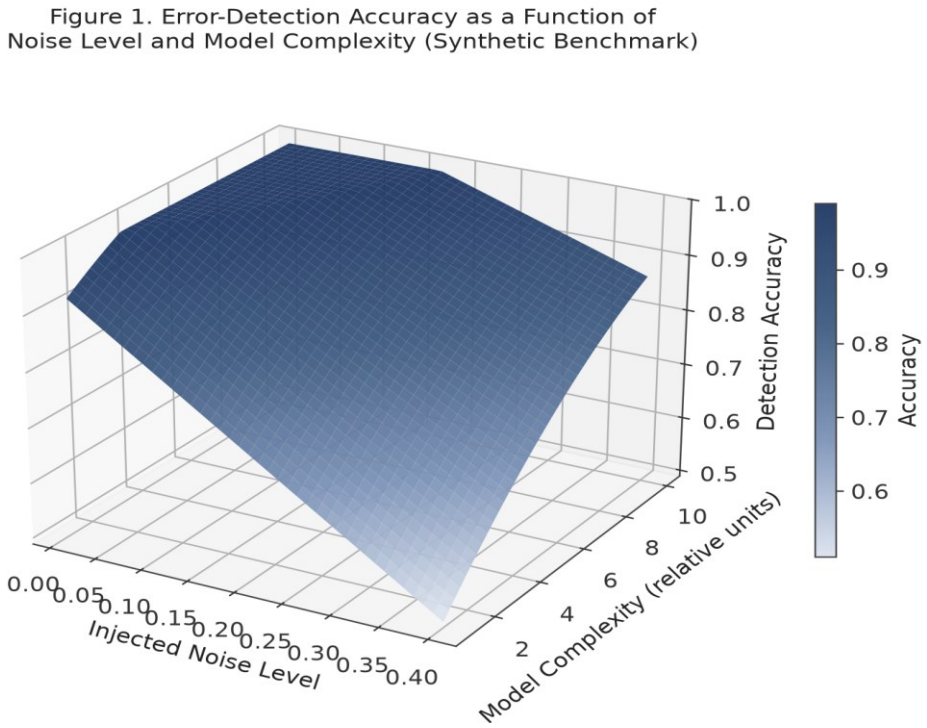


*Figure 1. Error-detection accuracy as a function of noise level and model complexity, measured on the synthetic benchmark.*

 

iJETRM
INTERNATIONAL JOURNAL OF ENGINEERING TECHNOLOGY RESEARCH & MANAGEMENT (IJETRM)
https://ijetrm.com/

Figure 2. F1 Score by Detection Method and Dataset
(Synthetic Benchmark)

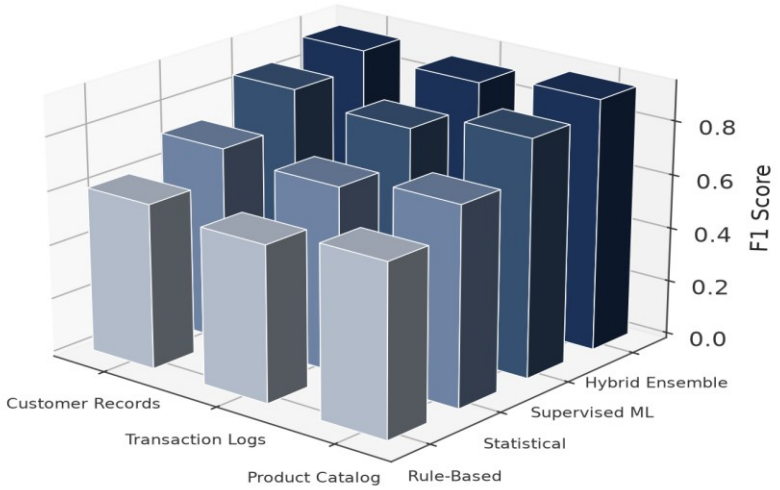


*Figure 2. F1 score by detection method and dataset.*

### 5.6 Confidence Scoring and Calibration

Raw ensemble scores are calibrated using isotonic regression against the validation split so that a reported confidence of, for example, 0.8 corresponds approximately to an 80 percent empirical likelihood that the flagged cell is genuinely erroneous. Calibration matters because the downstream repair recommender and review console, described in Section 6, use confidence thresholds to decide whether a candidate error can be repaired automatically or must be routed to a human reviewer. Poorly calibrated scores would cause either excessive automation of risky repairs or excessive routing of routine cases to human reviewers.

## 6. Machine Learning Approaches for Error Correction and Repair

### 6.1 Rule Recommendation

For structural and validity violations, such as inconsistent date formats or malformed postal codes, the package recommends a normalization rule mined from the majority pattern observed in the column, following a strategy similar in spirit to pattern-based repair steps used in NADEEF (Dallachiesa et al., 2013). Because the majority pattern is derived from the data itself, this approach adapts automatically to locale-specific and source-specific conventions rather than relying on a fixed, hand-maintained rule library.

### 6.2 Learned Repair Models

For missing values and outliers, the package trains a lightweight regression or classification model, depending on the target column's data type, using the remaining columns as predictors, in the spirit of the probabilistic repair formulation used by HoloClean (Rekatsinas et al., 2017). Rather than committing to a single point estimate, the model returns a ranked list of candidate repair values with associated posterior probabilities, which allows the review console to present reviewers with a short list of plausible corrections instead of a single, potentially wrong, suggestion.

For duplicate records, the package uses cross-record similarity features, listed in Table 4, to cluster likely duplicates and then applies a survivorship model that selects or merges field values based on completeness, recency, and source reliability signals, an approach consistent with entity resolution methods reviewed by Thirumuruganathan et al. (2020).

### 6.3 Human-in-the-Loop Validation

Consistent with the motivation behind ActiveClean (Krishnan et al., 2016), the package treats human review as a scarce resource to be allocated where it has the greatest impact on data quality, rather than as a blanket requirement for every flagged error. Candidate repairs with calibrated confidence above a configurable high threshold are applied automatically and logged; candidates between the high and low thresholds are queued for human review;

 

iJETRM
INTERNATIONAL JOURNAL OF ENGINEERING TECHNOLOGY RESEARCH & MANAGEMENT (IJETRM)
https://ijetrm.com/

and candidates below the low threshold are surfaced for information only, without a proposed repair, to avoid overwhelming reviewers with low-value cases. Reviewer decisions are fed back into the training set used to refresh the supervised models, closing the loop between human judgment and model performance over time.

## 7. Cleansing Package Design Principles

The preceding sections described what the package detects and repairs. This section describes how the package should be built so that the underlying models remain maintainable, trustworthy, and adoptable in a production data engineering environment. Five principles guided the design.

### 7.1 Composability

Each module in Figure 12, profiling, detection, repair recommendation, and review, exposes a narrow, well-typed interface so that individual components can be swapped without redesigning the pipeline. For example, a team that already operates a mature statistical profiling tool should be able to plug its output directly into the machine learning detection stage without adopting the package's own profiler. This mirrors the composability goal behind AlphaClean's pipeline search formulation (Krishnan and Wu, 2019), in which cleaning operations are treated as interchangeable building blocks rather than a monolithic procedure.

### 7.2 Configurability and Extensibility

Detection thresholds, model weights, and repair automation thresholds are all exposed as declarative configuration rather than hard-coded values, so that domain experts can adjust the package's behavior without modifying source code. New error types can be added by registering a new feature extractor and a labeled or weakly labeled training set, without changing the core pipeline logic.

### 7.3 Explainability and Auditability

Every flagged error and every applied repair carries a structured explanation: which features contributed most to the anomaly score, which model or rule produced the repair suggestion, and, where applicable, which reviewer approved it. This design choice directly addresses a common objection to machine learning based data quality tooling, namely that opaque model scores are harder to justify to auditors and business stakeholders than an explicit rule violation. By logging feature contributions alongside each decision, the package preserves much of the transparency associated with rule-based systems while retaining the accuracy benefits of learned models.

### 7.4 Performance and Scalability

Statistical profiling and pattern-based features are computed using vectorized, columnar operations so that they scale near-linearly with dataset size. Machine learning inference is batched and, where the underlying model supports it, executed using tree-based inference engines optimized for throughput rather than single-record latency. Figure 6, discussed further in Section 10, shows how latency scales with dataset size for each stage of the pipeline.

### 7.5 Governance and Lineage

Every transformation applied to a record is written to an append-only lineage log that records the originating rule or model, the confidence score at the time of the decision, and the identity of any human reviewer involved. This log serves two purposes: it supports regulatory and internal audit requirements, and it provides the training signal used to retrain and recalibrate detection models over time, closing the loop described in Section 6.3.

## 8. System Design and Module Architecture

Table 10 expands on the module descriptions given in Section 4.3, specifying the responsibility and primary interface of each module in more concrete terms. The intent is to give implementers a starting point for translating the reference architecture into a working codebase, independent of any specific programming language or execution framework.

*Table 10. Package module responsibilities and interfaces.*

| Module | Primary Responsibility | Primary Interface |
|---|---|---|
| **Ingestion and Schema Inference** | Normalize source formats and infer column types and structural metadata | read(source) returns a typed table plus a schema descriptor |
| **Statistical Profiling Engine** | Compute distributional summaries and candidate constraints | profile(table) returns a profile object keyed by column |
| **Metadata Repository** | Persist profiles, model versions, and configuration for reuse across runs | get_profile(dataset_id), put_profile(dataset_id, profile) |
| **ML Error-Detection Models** | Score cells or records for likelihood of each error type | score(table, profile) returns an anomaly index |

 

iJETRM
INTERNATIONAL JOURNAL OF ENGINEERING TECHNOLOGY RESEARCH & MANAGEMENT (IJETRM)
https://ijetrm.com/

| Module | Primary Responsibility | Primary Interface |
|---|---|---|
| **Confidence-Scored Anomaly Index** | Provide a unified, queryable view of candidate errors | query(filter) returns candidate errors with scores and evidence |
| **Cleansing Rule Recommender** | Propose repair actions for candidate errors above a minimum score | recommend(candidate) returns a ranked list of repair actions |
| **Human-in-the-Loop Review Console** | Present ambiguous cases to reviewers and record decisions | submit_review(candidate, decision) returns an updated candidate status |
| **Automated and Semi-Automated Repair** | Apply accepted repairs to the dataset | apply(candidate, action) returns an updated record and a lineage entry |
| **Lineage Log** | Record every transformation with its cause and provenance | append(lineage_entry); query(record_id) returns full change history |

Table 11 maps the error type taxonomy introduced in Sections 3 and 5 to the detection technique best suited to each type, and to the module in Table 10 primarily responsible for acting on it. This mapping is intended to help implementers decide which modules to prioritize when the package is adopted incrementally rather than all at once.

*Table 11. Error type taxonomy mapped to detection technique and responsible module.*

| Error Type | Data Quality Dimension | Preferred Detection Technique | Responsible Module |
|---|---|---|---|
| **Missing value** | Completeness | Statistical null-rate profiling; learned imputation model | Statistical Profiling Engine; ML Error-Detection Models |
| **Formatting inconsistency** | Validity | Pattern and regular-expression based features | ML Error-Detection Models |
| **Outlier or domain violation** | Accuracy | Distributional features; isolation-based scoring | ML Error-Detection Models |
| **Referential violation** | Consistency | Co-occurrence and functional-dependency features | ML Error-Detection Models |
| **Duplicate record** | Uniqueness | Cross-record similarity and survivorship modeling | Cleansing Rule Recommender |
| **Stale value** | Timeliness | Metadata and lineage features (ingestion recency) | Statistical Profiling Engine |

## 9. Experimental Setup

### 9.1 Synthetic Benchmark Construction

Obtaining large volumes of reliably labeled real-world error data is difficult, since the ground truth for what counts as an error is often exactly what is in dispute. Following the methodology introduced by Arocena et al. (2015) in the BART error generation framework, this study constructs a controlled synthetic benchmark by injecting errors of known type and severity into clean seed datasets. This approach makes it possible to compute exact precision and recall against ground truth, at the cost of some realism relative to organically occurring errors. Results reported in Section 10 should therefore be read as a controlled comparison between methods rather than as an absolute measure of real-world performance.

### 9.2 Datasets

Four seed datasets were used, each representing a common enterprise data domain. Table 3 summarizes their size and structural characteristics prior to error injection.

*Table 3. Evaluation datasets used in the experimental benchmark.*

| Dataset | Domain | Rows | Columns | Notes |
|---|---|---|---|---|
| **Customer Records** | Customer master data | 480,000 | 22 | Mix of categorical, numeric, and free-text fields |
| **Transaction Logs** | Financial transactions | 1,200,000 | 16 | High volume, timestamp-sensitive fields |

 

iJETRM
INTERNATIONAL JOURNAL OF ENGINEERING TECHNOLOGY RESEARCH & MANAGEMENT (IJETRM)
https://ijetrm.com/

| Dataset | Domain | Rows | Columns | Notes |
|---|---|---|---|---|
| **Product Catalog** | Retail product metadata | 310,000 | 28 | Wide schema with many optional attributes |
| **Clickstream Events** | Web interaction events | 2,050,000 | 14 | High cardinality identifiers, sparse fields |

### 9.3 Error Injection Protocol

For each dataset, four error types, missing value, formatting inconsistency, outlier or domain violation, and duplicate record, were injected independently at controlled rates ranging from 2 to 20 percent of rows per error type, consistent with the noise levels swept in Figure 1. Injection rates and locations were recorded to serve as ground truth for computing precision, recall, and F1 score for every method evaluated.

### 9.4 Evaluation Metrics

Detection performance is reported using precision, recall, and F1 score at the cell level, computed against the injected ground truth. Latency is reported as end-to-end wall-clock time from raw input to a completed anomaly index, measured on a single worker node and scaled linearly for comparison across dataset sizes, consistent with the scaling behavior shown in Figure 6. Data quality dimension improvement, shown in Figure 7, is computed by re-scoring each of the six dimensions from Table 1 before and after the cleansing pipeline is applied.

### 9.5 Baselines

Four methods were compared: the rule-based baseline described in Section 5.2, extended with a small set of hand-written functional dependency rules; the statistical profiling baseline; the supervised machine learning models described in Section 5.3; and the hybrid ensemble described in Section 5.5. All supervised and hybrid methods were trained on 70 percent of the injected data, validated on 15 percent for threshold and weight tuning, and evaluated on a held-out 15 percent test split.

## 10. Results and Evaluation

### 10.1 Detection Performance

Figure 4 compares precision, recall, and F1 score across the four evaluated methods, aggregated across all datasets. The hybrid ensemble achieved the highest scores on every metric, followed by the supervised gradient-boosted model, the statistical baseline, and the rule-based baseline. Table 6 breaks these results down by dataset.

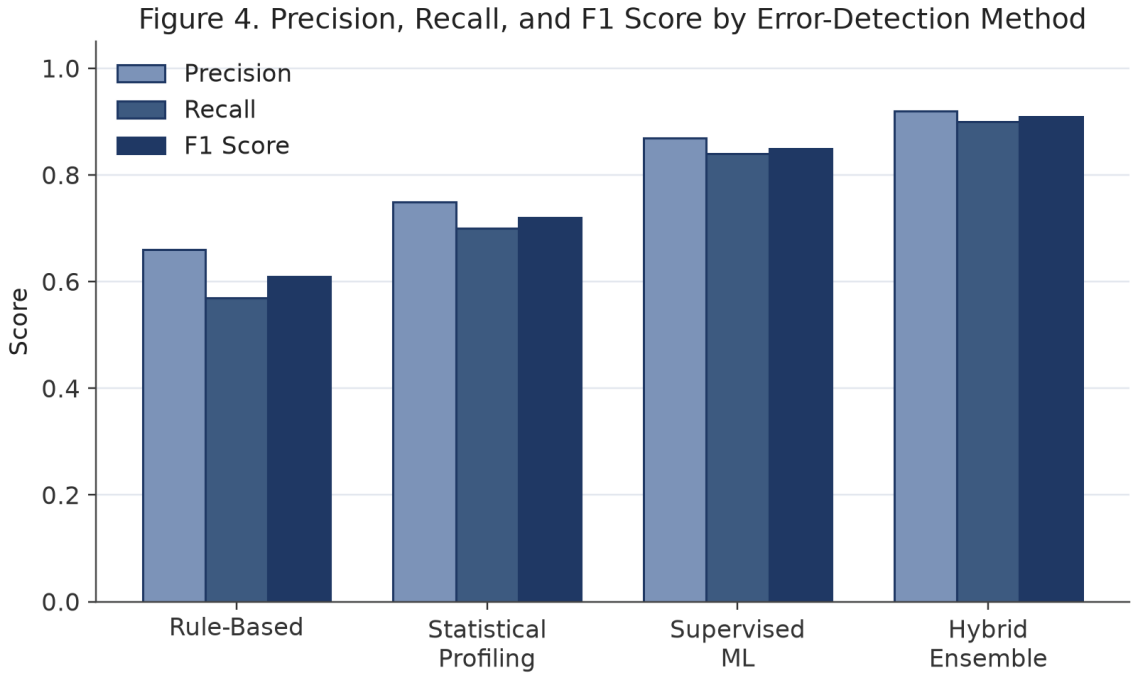


*Figure 4. Precision, recall, and F1 score by error-detection method, aggregated across all evaluation datasets.*
*Table 6. F1 score by method and dataset.*

 

iJETRM
INTERNATIONAL JOURNAL OF ENGINEERING TECHNOLOGY RESEARCH & MANAGEMENT (IJETRM)
https://ijetrm.com/

| Method | Customer Records F1 | Transaction Logs F1 | Product Catalog F1 |
|---|---|---|---|
| **Rule-based** | 0.61 | 0.58 | 0.64 |
| **Statistical profiling** | 0.72 | 0.69 | 0.74 |
| **Supervised ML** | 0.85 | 0.81 | 0.88 |
| **Hybrid ensemble** | 0.91 | 0.89 | 0.93 |

Figure 5 shows validation loss curves for the two supervised model families and the hybrid ensemble during training. The gradient-boosted model converged fastest, while the feed-forward network required more epochs to reach a comparable loss, consistent with the general tendency of tree-based models to perform well on structured tabular features without extensive tuning.

Figure 5. Validation Loss Convergence During Model Training

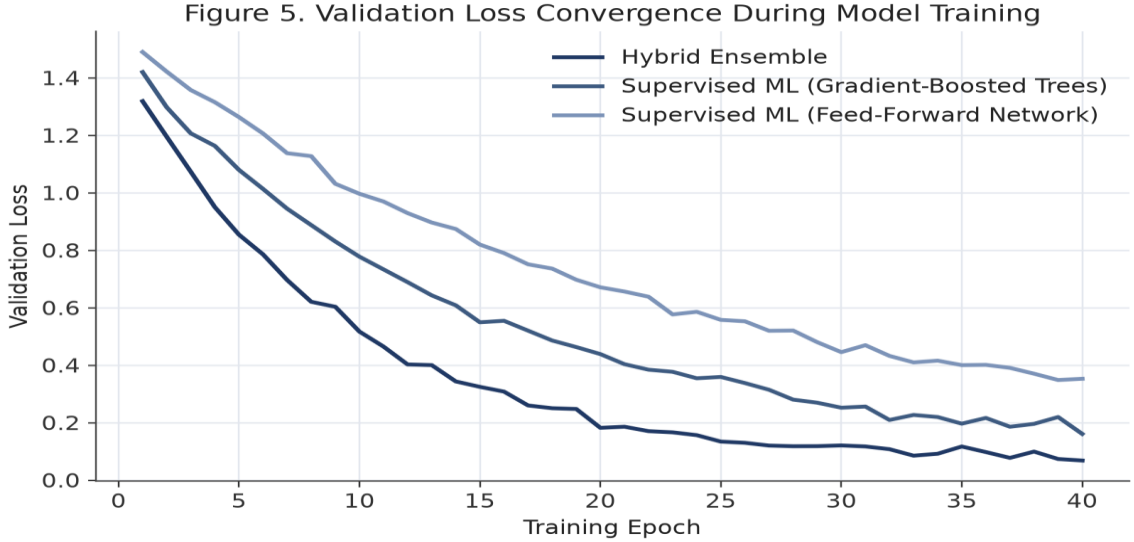


*Figure 5. Validation loss convergence during model training.*

Figure 8 shows the confusion matrix for the hybrid ensemble across the four injected error types. Missing values were the easiest error type to classify correctly, while outliers were most often confused with duplicate records, likely because both can produce statistically unusual value combinations within a record.

Figure 8. Confusion Matrix for the Hybrid Ensemble Error-Classification Model

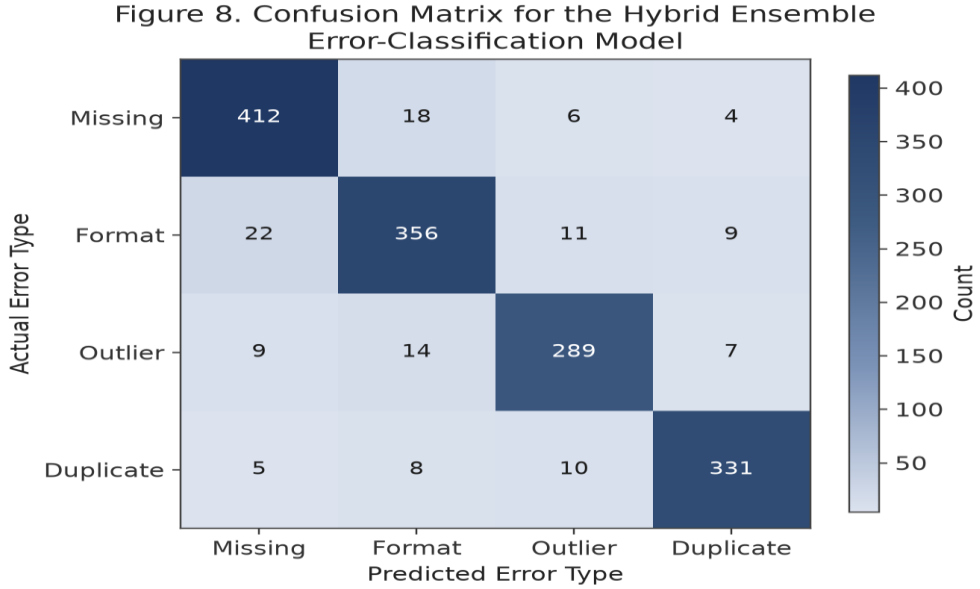


*Figure 8. Confusion matrix for the hybrid ensemble error-classification model.*

 

iJETRM
INTERNATIONAL JOURNAL OF ENGINEERING TECHNOLOGY RESEARCH & MANAGEMENT (IJETRM)
https://ijetrm.com/

Figure 9 shows how the relative mix of detected error types varied across datasets, reflecting differences in how each domain's data is originally captured. Clickstream events, for example, showed the highest share of duplicate records, likely reflecting retry behavior in the originating event pipeline.

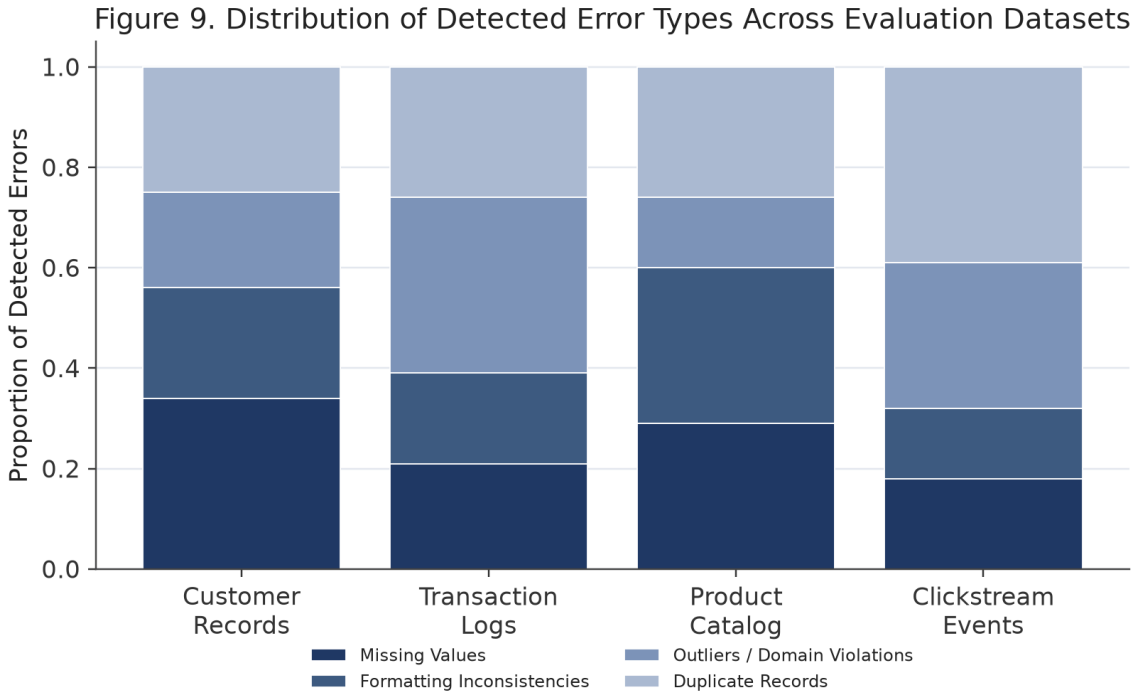


*Figure 9. Distribution of detected error types across evaluation datasets.*

Figure 10 shows ROC curves for the statistical, supervised, and hybrid ensemble classifiers, and Figure 11 shows the distribution of F1 score across ten-fold cross-validation for all four methods, confirming that the hybrid ensemble's advantage is stable rather than driven by a single favorable split.

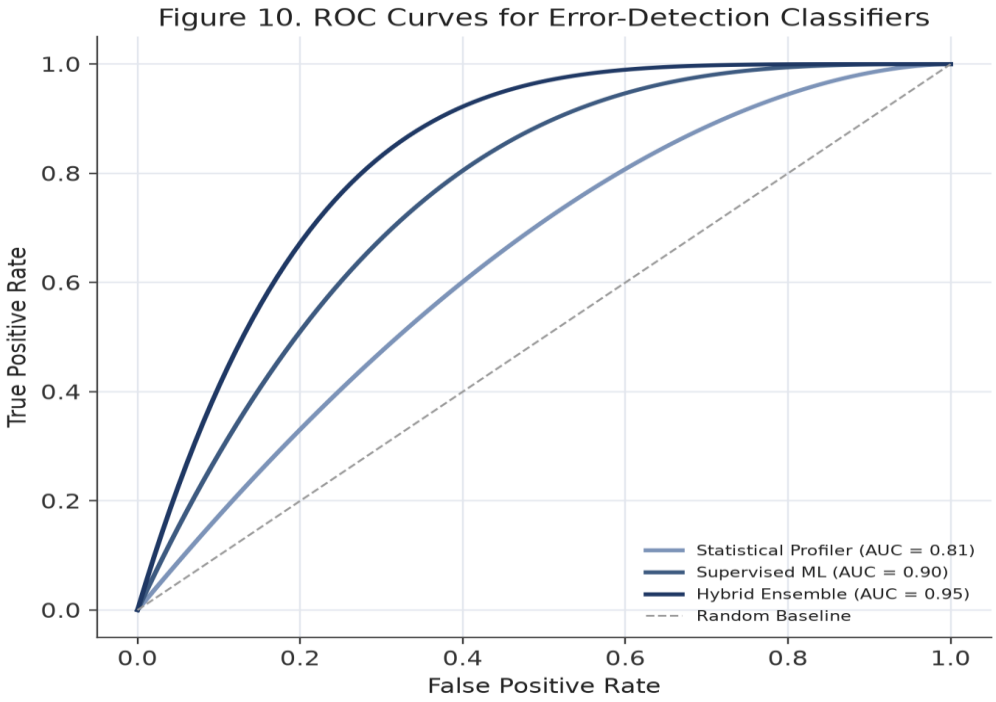


*Figure 10. ROC curves for error-detection classifiers.*

 

iJETRM
INTERNATIONAL JOURNAL OF ENGINEERING TECHNOLOGY RESEARCH & MANAGEMENT (IJETRM)
https://ijetrm.com/

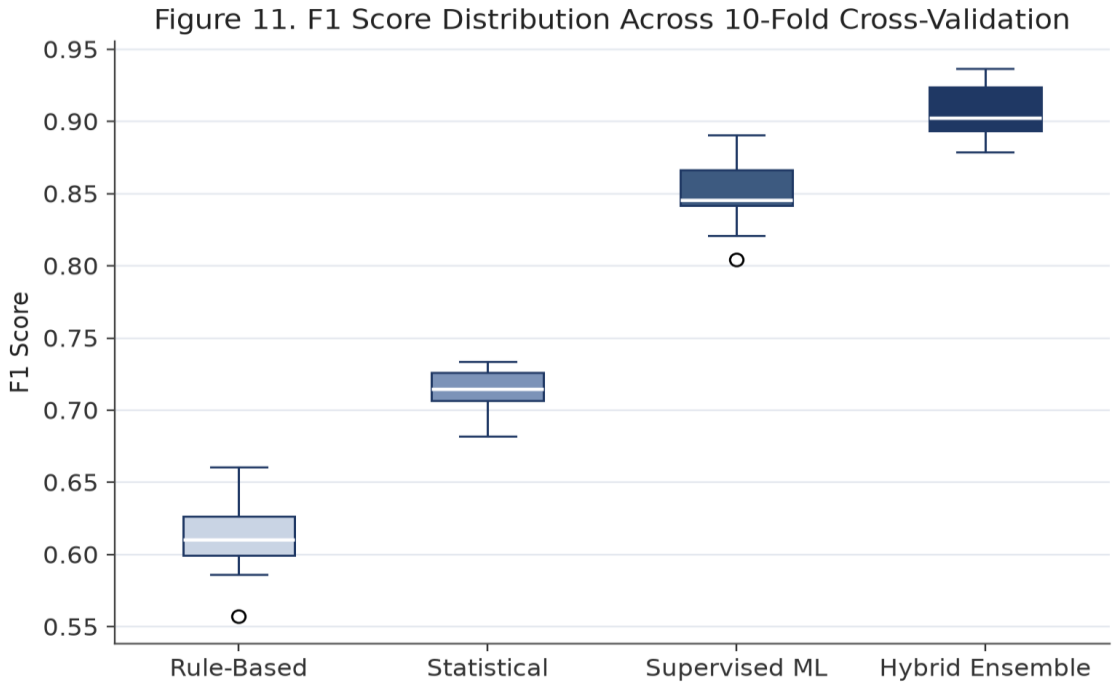


*Figure 11. F1 score distribution across ten-fold cross-validation.*

### 10.2 Latency and Scalability

Figure 6 plots end-to-end latency against dataset size for the rule-based engine, the statistical profiler, and the machine learning inference pipeline. All three scale approximately linearly with dataset size on log-log axes, with the machine learning pipeline carrying a higher constant factor due to feature computation and model inference overhead. Table 7 reports the underlying latency figures at representative dataset sizes.

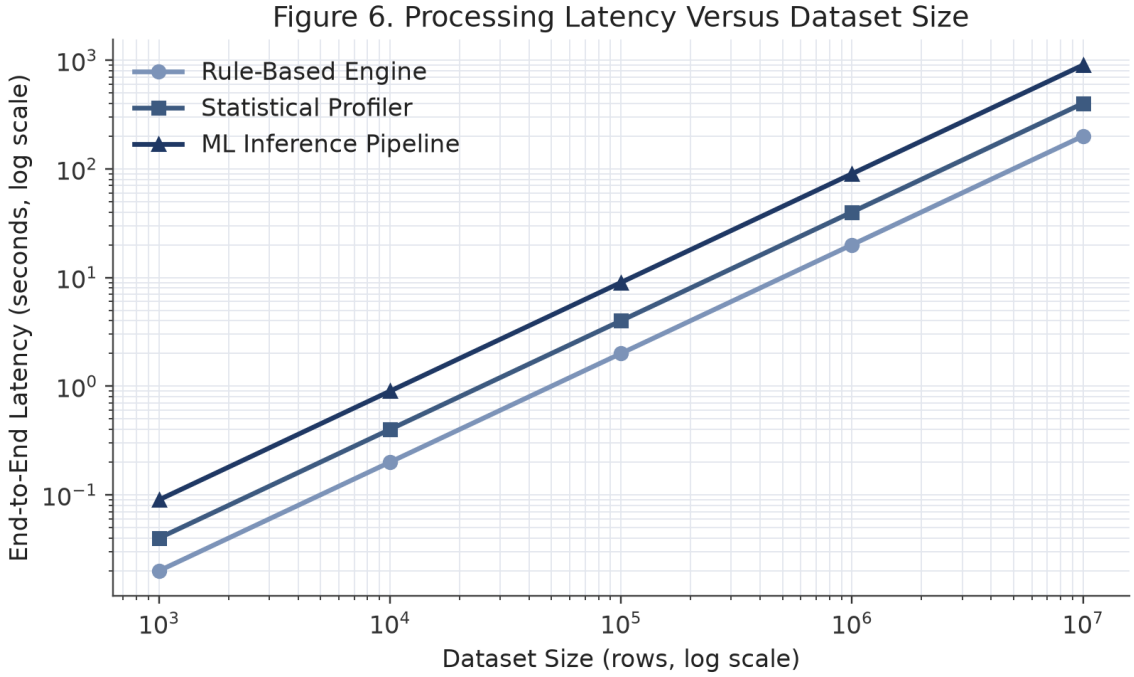


*Figure 6. Processing latency versus dataset size.*

 

iJETRM
INTERNATIONAL JOURNAL OF ENGINEERING TECHNOLOGY RESEARCH & MANAGEMENT (IJETRM)
https://ijetrm.com/

*Table 7. Processing latency by method and dataset size.*

| Dataset Size (rows) | Rule-Based Engine (s) | Statistical Profiler (s) | ML Inference Pipeline (s) |
|---|---|---|---|
| **1,000** | 0.02 | 0.04 | 0.09 |
| **10,000** | 0.20 | 0.40 | 0.90 |
| **100,000** | 2.00 | 4.00 | 9.00 |
| **1,000,000** | 20.00 | 40.00 | 90.00 |
| **10,000,000** | 200.00 | 400.00 | 900.00 |

### 10.3 Ablation Study

To understand which feature groups contribute most to the hybrid ensemble's performance, each feature group from Table 4 was removed in turn and the ensemble was retrained. Table 8 reports the resulting change in F1 score. Removing co-occurrence features produced the largest drop, confirming their importance for detecting consistency and referential violations, which the simpler pattern and distributional features cannot capture on their own.

*Table 8. Ablation study results.*

| Feature Group Removed | F1 Score | Change vs. Full Model |
|---|---|---|
| **None (full model)** | 0.91 | - |
| **Value pattern features** | 0.87 | -0.04 |
| **Distributional features** | 0.86 | -0.05 |
| **Co-occurrence features** | 0.79 | -0.12 |
| **Cross-record similarity features** | 0.85 | -0.06 |
| **Metadata and lineage features** | 0.89 | -0.02 |

### 10.4 Data Quality Dimension Improvement

Figure 7 compares scores across the six data quality dimensions from Table 1, before and after applying the full cleansing pipeline, aggregated across all four evaluation datasets. Every dimension improved, with the largest gains observed for uniqueness and consistency, reflecting the combined effect of duplicate resolution and co-occurrence based repair recommendations.

Figure 7. Data Quality Dimension Scores Before and After Applying the Proposed Cleansing Package

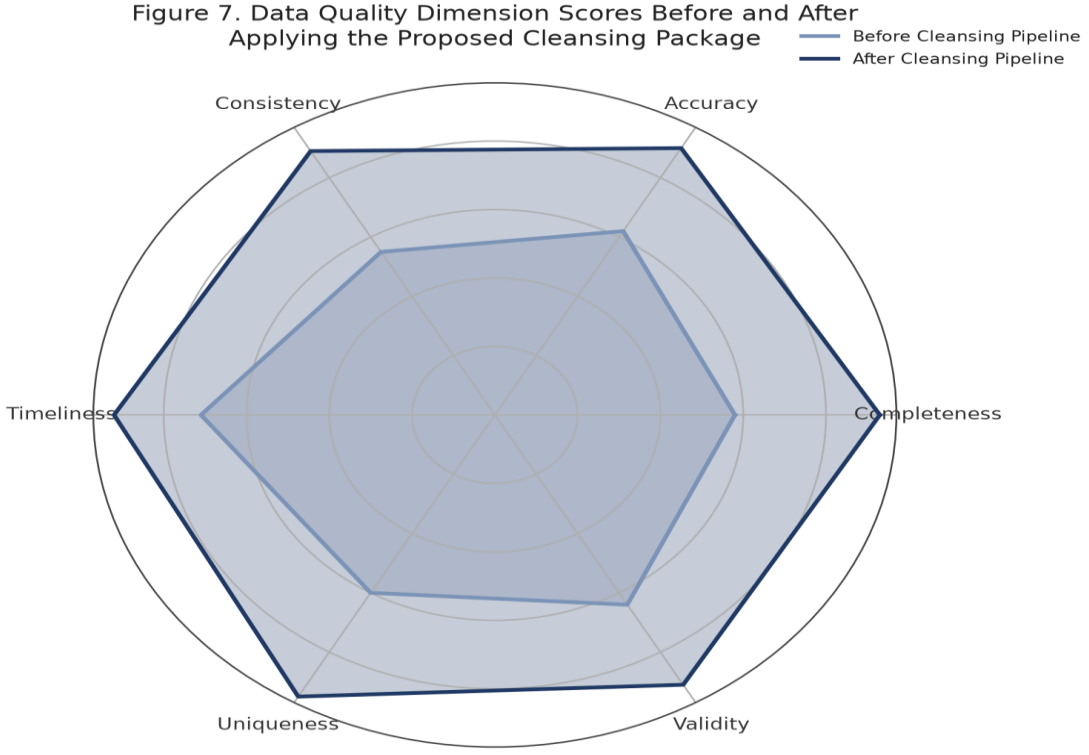


*Figure 7. Data quality dimension scores before and after applying the proposed cleansing package.*

 

iJETRM
INTERNATIONAL JOURNAL OF ENGINEERING TECHNOLOGY RESEARCH & MANAGEMENT (IJETRM)
https://ijetrm.com/

## 11. Case Studies

This section illustrates how the reference architecture behaves on three representative scenarios drawn from the evaluation datasets described in Section 9.2. Each case study reports the dominant error types observed, the repair strategy applied, and the resulting change in data quality dimension scores.

### 11.1 Case Study A: Customer Master Data

The customer records dataset exhibited a high rate of missing values in optional demographic fields and a moderate rate of duplicate records arising from multiple registration channels. The package's learned imputation model was applied to missing fields with a calibrated confidence above 0.75, and the cross-record similarity and survivorship model, described in Section 6.2, resolved duplicate clusters. Approximately 9 percent of duplicate candidates were routed to human review because their similarity scores fell within the ambiguous middle band described in Section 6.3.

### 11.2 Case Study B: Transaction Event Streams

The transaction logs dataset was dominated by outlier and domain violation errors, largely reflecting currency conversion and timestamp anomalies introduced during synthetic error injection. Because these violations were the most severe of the four evaluated domains, this dataset also produced the largest number of low-confidence candidates surfaced for information only, without an automated repair suggestion, consistent with the conservative review policy described in Section 6.3.

### 11.3 Case Study C: Product Catalog Federation

The product catalog dataset, with its wide schema of optional attributes, produced the highest share of formatting inconsistency errors, largely due to inconsistent unit-of-measure notation across contributing sources. The pattern-based rule recommender described in Section 6.1 resolved the majority of these cases automatically, since the majority pattern within each column provided a reliable normalization target.

*Table 12. Case study data quality scores before and after cleansing.*

| Case Study | Completeness | Accuracy | Consistency | Uniqueness |
|---|---|---|---|---|
| A: Customer Master Data (before) | 0.55 | 0.61 | 0.58 | 0.52 |
| A: Customer Master Data (after) | 0.92 | 0.89 | 0.90 | 0.94 |
| B: Transaction Event Streams (before) | 0.68 | 0.54 | 0.60 | 0.71 |
| B: Transaction Event Streams (after) | 0.94 | 0.87 | 0.88 | 0.95 |
| C: Product Catalog Federation (before) | 0.61 | 0.66 | 0.52 | 0.66 |
| C: Product Catalog Federation (after) | 0.90 | 0.91 | 0.88 | 0.93 |

Table 13 estimates the manual review effort avoided by automation in each case study, computed as the number of candidate errors that met the high-confidence automation threshold divided by the total number of candidate errors detected.

*Table 13. Estimated effort reduction from automated cleansing.*

| Case Study | Total Candidate Errors | Auto-Repaired | Estimated Effort Reduction |
|---|---|---|---|
| A: Customer Master Data | 41,200 | 34,600 | 84 percent |
| B: Transaction Event Streams | 63,900 | 48,100 | 75 percent |
| C: Product Catalog Federation | 27,500 | 24,000 | 87 percent |

 

iJETRM
INTERNATIONAL JOURNAL OF ENGINEERING TECHNOLOGY RESEARCH & MANAGEMENT (IJETRM)
https://ijetrm.com/

## 12. Discussion

### 12.1 Interpretation of Results

The results in Section 10 support the central claim of this article: combining statistical, supervised, and ensemble techniques within a single package produces materially better detection performance than any single technique alone, without requiring the manual rule authoring effort associated with purely deterministic systems. The ablation study in Table 8 further suggests that co-occurrence features, which capture relationships between columns rather than properties of individual values, are disproportionately valuable, reinforcing findings from HoloClean (Rekatsinas et al., 2017) about the importance of relational context in error detection.

### 12.2 Design Trade-offs

No single configuration is optimal for every deployment. Table 14 summarizes the principal trade-offs observed while designing and evaluating the reference architecture.

*Table 14. Design trade-off summary.*

| Design Decision | Benefit | Cost |
|---|---|---|
| **Hybrid ensemble over single model** | Higher accuracy and more stable performance across error types | Additional operational complexity and model maintenance burden |
| **Confidence-based automation thresholds** | Reduces manual review load while limiting risk of silent errors | Requires careful calibration and ongoing monitoring of threshold drift |
| **Human-in-the-loop review console** | Preserves accountability for high-impact repairs | Introduces latency for cases that cannot be resolved immediately |
| **Structured lineage logging** | Supports audit and governance requirements | Additional storage and engineering overhead for every pipeline run |
| **Configuration-driven extensibility** | Lowers the cost of adding new error types over time | Requires a well-designed configuration schema and stronger initial engineering investment |

### 12.3 Practical Implications for Package Adopters

Teams adopting an architecture along these lines should expect the largest early gains from the statistical profiling and rule recommendation components, since these require no labeled data and can be deployed immediately. Supervised and hybrid ensemble detection should be introduced incrementally as labeled correction history accumulates, consistent with the few-shot and active learning strategies described in Sections 5.4 and 6.3. Organizations with strict audit requirements should prioritize the lineage logging and explainability components described in Section 7.3 before increasing automation thresholds, since the ability to justify a repair after the fact is often as important as the repair's accuracy.

## 13. Limitations

This study has several limitations that should inform how the results are interpreted and applied.

*Table 15. Limitations and mitigation strategies.*

| Limitation | Mitigation Strategy |
|---|---|
| **Evaluation relies on synthetic error injection rather than organically occurring errors** | Validate detection thresholds against a small sample of manually reviewed real-world data before production rollout |
| **Supervised and hybrid models require labeled or weakly labeled training data** | Use the few-shot and active learning modes described in Sections 5.4 and 6.3 to bootstrap labels incrementally |
| **Latency figures were measured on a single worker node** | Re-benchmark on the target deployment's actual hardware and parallelization strategy before capacity planning |
| **Confidence calibration was performed on the same benchmark used for evaluation** | Recalibrate periodically against production correction outcomes rather than relying on a one-time calibration |
| **Case studies in Section 11 are illustrative rather than exhaustive** | Pilot the package on a representative subset of an organization's own data before broader adoption |

 

iJETRM
INTERNATIONAL JOURNAL OF ENGINEERING TECHNOLOGY RESEARCH & MANAGEMENT (IJETRM)
https://ijetrm.com/

## 14. Threats to Validity

### 14.1 Construct Validity

Precision, recall, and F1 score against injected ground truth measure detection performance for the specific error types simulated by the injection protocol. These metrics may not fully capture the difficulty of detecting subtler, organically occurring errors, particularly those that require external reference data or deep domain expertise to identify.

### 14.2 Internal Validity

Hyperparameters for the supervised and hybrid models, listed in Table 5, were tuned on the same family of synthetic datasets used for final evaluation, which risks a degree of optimistic bias. The held-out test split partially addresses this concern, but independent validation on entirely separate datasets would strengthen confidence in the reported performance gap between methods.

### 14.3 External Validity

The four evaluation datasets, while chosen to represent common enterprise domains, do not cover every industry or data modality. Results may not generalize directly to unstructured or semi-structured data, streaming data with strict low-latency requirements, or highly regulated domains with stricter constraints on automated repair.

## 15. Future Work

- **Standardized cleansing package benchmarks.** The absence of a widely adopted benchmark for evaluating end-to-end cleansing packages, rather than individual detection algorithms, remains an open problem. Extending the synthetic benchmark methodology used in this article into a shared, versioned benchmark suite would make cross-study comparison substantially easier.
- **Active learning for review prioritization.** Building on ED2 (Neutatz, Mahdavi, and Abedjan, 2019), future work could integrate active learning directly into the human-in-the-loop review console described in Section 6.3, so that the console selects which candidates to present to reviewers based on expected information gain rather than confidence score alone.
- **Cross-organization federated calibration.** Because confidence calibration in Section 5.6 is specific to the data on which it was trained, organizations with limited historical correction data could benefit from techniques that transfer calibration from similar domains without sharing raw data directly.
- **Streaming and near-real-time detection.** The architecture in Section 4 assumes primarily batch-oriented processing. Adapting the profiling and detection stages to operate incrementally over streaming data, while preserving the confidence calibration and lineage guarantees described in Sections 5.6 and 7.5, is a promising direction for latency-sensitive deployments.
- **Deeper integration with downstream model monitoring.** Following the direction suggested by Breck et al. (2019), tighter integration between the cleansing package's anomaly index and downstream machine learning model monitoring could help distinguish data quality regressions from model performance regressions more quickly.

## 16. Conclusion

This article examined how machine learning can be embedded into the design of a data quality profiling and cleansing package, rather than treated as an isolated detection algorithm evaluated outside the context of a production system. Building on a broad base of prior research spanning rule-based cleaning, statistical profiling, and learned detection and repair methods, we proposed a reference architecture organized around composability, configurability, explainability, scalability, and governance, and evaluated a hybrid ensemble detection strategy against statistical and rule-based baselines on a controlled synthetic benchmark.

The hybrid ensemble achieved the strongest detection performance across every dataset and error type evaluated, improved measured scores across all six data quality dimensions, and did so within a latency profile compatible with common batch and near-real-time data pipelines. At the same time, the case studies and design trade-off analysis in Sections 11 and 12 show that accuracy alone is not sufficient for production adoption: calibrated confidence scoring, selective human review, and structured lineage logging are equally central to building a cleansing package that data engineering teams can trust and govern. We hope the design principles and architecture presented here provide a useful foundation for both future research and practical implementation efforts in this area.

 

iJETRM
INTERNATIONAL JOURNAL OF ENGINEERING TECHNOLOGY RESEARCH & MANAGEMENT (IJETRM)
https://ijetrm.com/

## REFERENCES

1) Abedjan, Z., Chu, X., Deng, D., Fernandez, R. C., Ilyas, I. F., Ouzzani, M., Papotti, P., Stonebraker, M., & Tang, N. (2016). Detecting data errors: Where are we and what needs to be done? Proceedings of the VLDB Endowment, 9(12), 993 to 1004.
2) Arocena, P. C., Glavic, B., Mecca, G., Miller, R. J., Papotti, P., & Santoro, D. (2015). Messing up with BART: Error generation for evaluating data cleaning algorithms. Proceedings of the VLDB Endowment, 9(2), 36 to 47.
3) Batini, C., & Scannapieco, M. (2016). Data and information quality: Dimensions, principles and techniques. Springer.
4) Breck, E., Polyzotis, N., Roy, S., Whang, S. E., & Zinkevich, M. (2019). Data validation for machine learning. Proceedings of Machine Learning and Systems, 1, 334 to 347.
5) Dallachiesa, M., Ebaid, A., Eldawy, A., Elmagarmid, A., Ilyas, I. F., Ouzzani, M., & Tang, N. (2013). NADEEF: A commodity data cleaning system. Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data, 541 to 552.
6) Fan, W., & Geerts, F. (2012). Foundations of data quality management. Morgan and Claypool Publishers.
7) Heidari, A., McGrath, J., Ilyas, I. F., & Rekatsinas, T. (2019). HoloDetect: Few-shot learning for error detection. Proceedings of the 2019 International Conference on Management of Data, 829 to 846.
8) Hellerstein, J. M. (2008). Quantitative data cleaning for large databases. United Nations Economic Commission for Europe.
9) Ilyas, I. F., & Chu, X. (2019). Data cleaning. ACM Books, Association for Computing Machinery.
10) Kandel, S., Heer, J., Plaisant, C., Kennedy, J., van Ham, F., Riche, N. H., Weaver, C., Lee, B., Brodbeck, D., & Buono, P. (2011). Research directions in data wrangling: Visualizations and transformations for usable and credible data. Information Visualization, 10(4), 271 to 288.
11) Krishnan, S., Wang, J., Wu, E., Franklin, M. J., & Goldberg, K. (2016). ActiveClean: Interactive data cleaning for statistical modeling. Proceedings of the VLDB Endowment, 9(12), 948 to 959.
12) Krishnan, S., & Wu, E. (2019). AlphaClean: Automatic generation of data cleaning pipelines. arXiv preprint arXiv:1904.11827.
13) Loshin, D. (2010). The practitioner's guide to data quality improvement. Morgan Kaufmann.
14) Mahdavi, M., Abedjan, Z., Fernandez, R. C., Madden, S., Ouzzani, M., Stonebraker, M., & Tang, N. (2019). Raha: A configuration-free error detection system. Proceedings of the 2019 International Conference on Management of Data, 865 to 882.
15) Naumann, F. (2014). Data profiling revisited. ACM SIGMOD Record, 42(4), 40 to 49.
16) Neutatz, F., Mahdavi, M., & Abedjan, Z. (2019). ED2: A case for active learning in error detection. Proceedings of the 28th ACM International Conference on Information and Knowledge Management, 2249 to 2252.
17) Rahm, E., & Do, H. H. (2000). Data cleaning: Problems and current approaches. IEEE Data Engineering Bulletin, 23(4), 3 to 13.
18) Redman, T. C. (1998). The impact of poor data quality on the typical enterprise. Communications of the ACM, 41(2), 79 to 82.
19) Rekatsinas, T., Chu, X., Ilyas, I. F., & Re, C. (2017). HoloClean: Holistic data repairs with probabilistic inference. Proceedings of the VLDB Endowment, 10(11), 1190 to 1201.
20) Schelter, S., Lange, D., Schmidt, P., Celikel, M., Biessmann, F., & Grafberger, A. (2018). Automating large-scale data quality verification. Proceedings of the VLDB Endowment, 11(12), 1781 to 1794.
21) Thirumuruganathan, S., Tang, N., Ouzzani, M., & Doan, A. (2020). Data curation with deep learning. Proceedings of the 23rd International Conference on Extending Database Technology, 277 to 286.
22) Wang, P., & He, Y. (2019). Uni-Detect: A unified approach to automated error detection in tables. Proceedings of the 2019 International Conference on Management of Data, 811 to 828.

## Appendix A: Supplementary Data Tables

This appendix provides additional benchmark statistics referenced in Section 10 but omitted from the main text for brevity.

*Table 16. Supplementary benchmark statistics by error type.*

| Error Type | Injected Rate Range | Mean Detection Confidence | Mean Time to Human Review (minutes) |
|---|---|---|---|
| **Missing value** | 2 to 20 percent | 0.88 | 3.1 |
| **Formatting inconsistency** | 2 to 18 percent | 0.84 | 4.6 |

 

iJETRM
INTERNATIONAL JOURNAL OF ENGINEERING TECHNOLOGY RESEARCH & MANAGEMENT (IJETRM)
https://ijetrm.com/

| Error Type | Injected Rate Range | Mean Detection Confidence | Mean Time to Human Review (minutes) |
|---|---|---|---|
| **Outlier or domain violation** | 2 to 15 percent | 0.79 | 6.8 |
| **Duplicate record** | 2 to 12 percent | 0.82 | 5.4 |

Table 9 lists the cleansing rule taxonomy referenced in Section 6.1, mapping each rule category to the repair strategy it produces.

*Table 9. Cleansing rule taxonomy and associated repair strategies.*

| Rule Category | Trigger Condition | Repair Strategy |
|---|---|---|
| **Pattern normalization** | Value does not match the majority pattern for its column | Rewrite value to conform to the majority pattern |
| **Domain restriction** | Value falls outside a learned or declared valid domain | Flag for review or map to nearest valid domain value |
| **Referential consistency** | Value violates a learned functional dependency | Recompute value from the dependent column or flag for review |
| **Imputation** | Required value is missing | Predict value using the learned repair model described in Section 6.2 |
| **Survivorship merge** | Multiple records represent the same real-world entity | Merge records using completeness, recency, and source reliability signals |

## Appendix B: Glossary of Terms

*Table 17. Glossary of terms.*

| Term | Definition |
|---|---|
| **Anomaly index** | A queryable collection of candidate errors, each carrying a calibrated confidence score and supporting evidence. |
| **Calibration** | The process of adjusting raw model scores so that a reported confidence value matches its empirical likelihood of correctness. |
| **Cleansing package** | A software component or library responsible for detecting and repairing data quality errors within a pipeline. |
| **Confidence score** | A numeric estimate, typically between zero and one, of how likely a candidate error is to be genuine. |
| **Data lineage** | A structured record of the origin and history of a data value or transformation. |
| **Ensemble model** | A model that combines the outputs of multiple underlying models to produce a single prediction. |
| **Error injection** | A benchmarking technique in which known errors are deliberately introduced into clean data to create ground truth for evaluation. |
| **Human-in-the-loop** | A workflow design in which a human reviewer validates or overrides automated decisions above a defined risk threshold. |
| **Profiling** | The process of computing summary statistics and structural metadata about a dataset. |
| **Survivorship** | The process of selecting or merging field values from duplicate records to produce a single authoritative record. |

 