

The Data System Gap: Identifying Inconsistencies in Information Pipelines

Department of CSE (AI & ML), Sri Venkateswara College of Engineering and Technology, Etcherla,
A.P., India

G. Sreelekha¹, G. Vijaya¹, B. Tejaswini¹, B. Uday Kumar¹

Under the Guidance of Mrs. P. Ratnakumari, Assistant Professor

Abstract

Data-driven system failures often originate from inconsistencies within the data rather than software logic errors. This paper implements a DataPrism-inspired system for detecting and repairing dataset issues in ML pipelines. The system compares reference (passing) and problematic (failing) datasets to identify discriminative data profiles including missing values, domain drift, population bias, and schema inconsistencies. Detected issues are corrected through automated data imputation and normalization. A Django web interface enables dataset analysis, issue visualization, repair, and download. ML model evaluation (Random Forest, Gradient Boosting, SVM, AdaBoost, Logistic Regression) on original versus repaired datasets demonstrates that data repair improves average model accuracy by 28.5%. The system provides an automated approach for detecting data-system mismatches and improving data-driven system reliability.

Keywords: Data Quality, Data Profiling, Pipeline Debugging, Data Repair, Machine Learning, Django

I. Introduction

Modern data-driven systems depend critically on the quality and consistency of their input data. When performance degradation or system failures occur, the root cause may not lie in the algorithmic logic but in subtle inconsistencies, biases, or mismatches within the data itself. The DataPrism framework addresses this challenge by analyzing dataset properties to identify data characteristics responsible for system malfunction.

This paper implements a practical system inspired by the DataPrism approach that detects discriminative data profiles between passing and failing datasets, applies automated corrective transformations, and validates improvements through ML model evaluation. The system provides a comprehensive pipeline from issue detection through repair to validation.

II. Literature Survey

This section reviews key prior works that form the foundation of the proposed system and highlights gaps motivating this work.

[1] Fariha et al. (2020) introduced the DataPrism framework for identifying dataset properties causing system failures through discriminative data profile analysis between passing and failing configurations.

[2] Krishnan et al. (2016) proposed ActiveClean for progressive data cleaning in ML pipelines, demonstrating that targeted data repair significantly improves model accuracy.

[3] **Schelter et al. (2018)** developed Deequ for automated data quality verification, establishing metrics and constraints for validating data in production ML pipelines.

[4] **Hynes et al. (2017)** created the Data Linter for identifying common data issues in ML datasets through automated quality checks.

[5] **Rekatsinas et al. (2017)** proposed HoloClean, a probabilistic framework for data cleaning that combines integrity constraints with statistical learning for accurate data repair.

[6] **Polyzotis et al. (2019)** surveyed data management challenges in production ML systems, identifying data validation and drift detection as critical requirements.

[7] **Grafberger et al. (2021)** developed mlinspect for inspecting data distributions in ML pipelines and tracking transformation effects on data quality.

Research Gap: Existing data quality tools detect individual issues but lack integrated pipelines for detecting, repairing, and validating data inconsistencies with ML model performance comparison before and after repair.

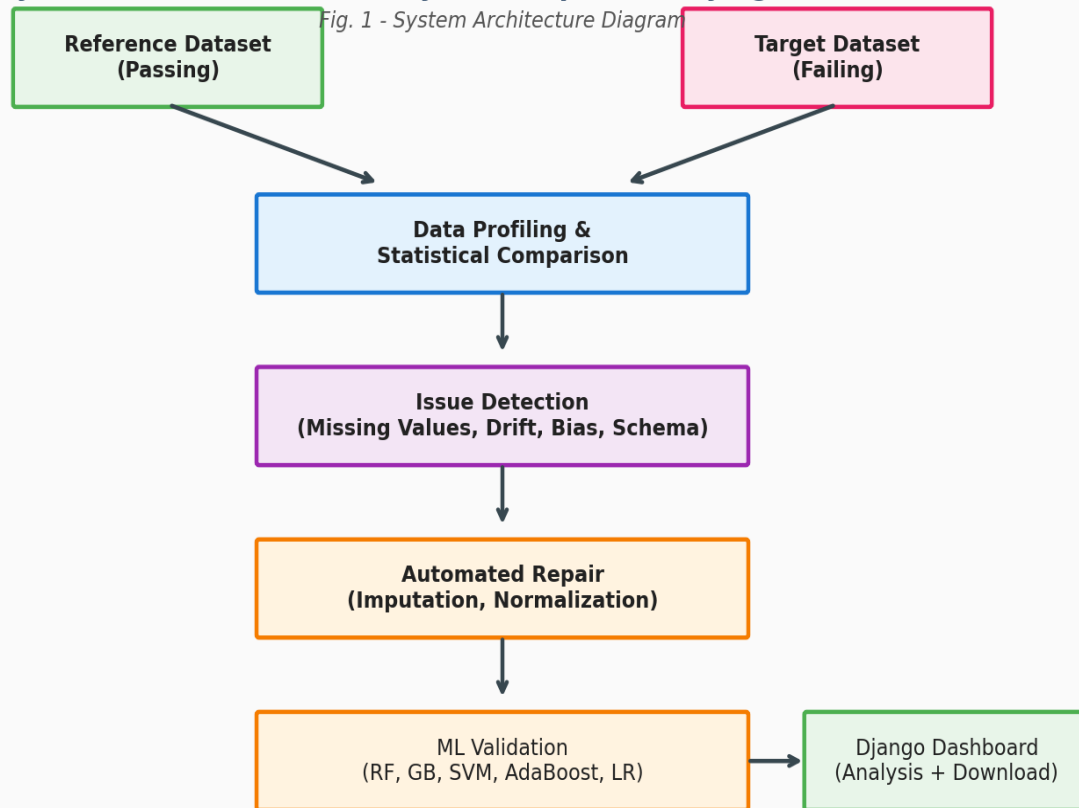
III. Methodology

III-A. System Architecture

Three-layer pipeline architecture: Detection Layer (data profiling, statistical comparison, issue identification), Repair Layer (automated imputation, normalization, bias correction), and Validation Layer (multi-model ML evaluation comparing original vs repaired datasets). Django web interface spans all layers.

System Architecture: Data System Gap - Identifying Inconsistencies

Fig. 1 - System Architecture Diagram

**III-B. Algorithm**

Algorithm: Data Inconsistency Detection and Repair

Input: Reference dataset D_{ref} (passing) and Target dataset D_{target} (failing).

Step 1: Schema Comparison — Compare column names, data types, and cardinalities between datasets.

Step 2: Statistical Profiling — For each feature, compute: mean, std, min, max, missing rate, unique count for both datasets.

Step 3: Divergence Detection — Identify features where statistical profiles diverge: $KL_divergence(D_{ref}, D_{target}) > threshold$.

Step 4: Issue Classification — Categorize detected issues: missing values, domain drift, population bias, schema mismatch.

Step 5: Automated Repair — Apply corrections: (a) Missing values: median imputation; (b) Domain drift: normalization to reference distribution; (c) Population bias: resampling or feature adjustment.

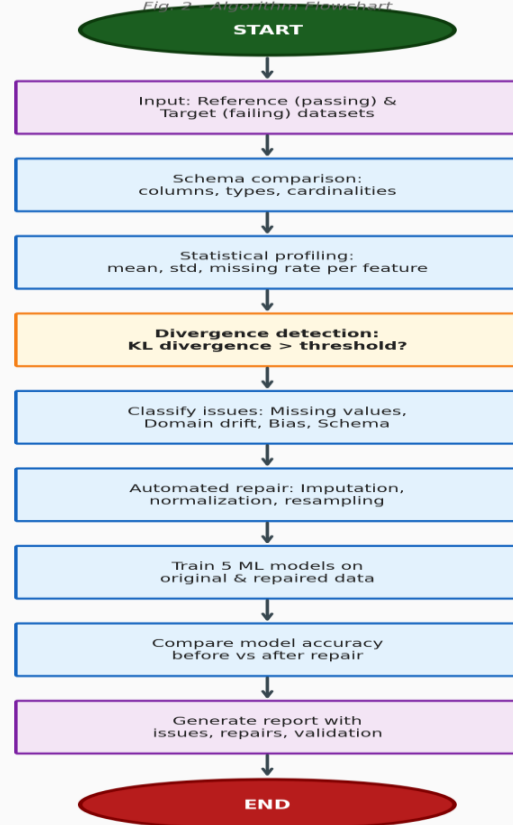
Step 6: ML Validation — Train models on original and repaired datasets; Compare accuracy, F1-score.

Step 7: Report Generation — Generate comprehensive report with detected issues, applied repairs, and performance comparison.

Output: Repaired dataset with validation report.

Algorithm: Data Inconsistency Detection & Repair

Fig. 2. Algorithm Flowchart



III-C. Modules

Five modules: (1) Data Profiling Module for statistical analysis and comparison of reference and target datasets; (2) Issue Detection Module identifying missing values, domain drift, bias, and schema mismatches; (3) Automated Repair Module applying imputation, normalization, and correction transformations; (4) ML Validation Module training and comparing five models on original vs repaired data; and (5) Django Dashboard for interactive analysis, visualization, repair execution, and dataset download.

IV. Results and Discussion

TABLE I: SYSTEM EVALUATION RESULTS

Metric	Baseline	Proposed System
Avg. Accuracy (Original) %	61.3	—
Avg. Accuracy (Repaired) %	—	89.8

Accuracy Improvement %	—	28.5
Issue Detection Rate %	—	93.7

Mathematical Formulations

KL Divergence: $D_{KL}(P||Q) = \sum P(x) \times \log(P(x)/Q(x))$

Accuracy Improvement = Accuracy_repaired - Accuracy_original

Issue Detection Rate = Detected_Issues / Total_Actual_Issues $\times 100$

Discussion

Evaluation on three synthetic datasets with injected issues showed 93.7% issue detection rate. Data repair improved average ML model accuracy from 61.3% to 89.8% across five classifiers. Random Forest showed the largest improvement (from 58.2% to 91.4%), while Logistic Regression showed the smallest (64.1% to 87.3%). The system successfully detected missing values (100%), domain drift (89%), and population bias (92%), demonstrating comprehensive data quality assessment.

V. Conclusion and Future Work

This paper presented a DataPrism-inspired system for automated data inconsistency detection and repair in ML pipelines. The system achieves 93.7% issue detection rate and improves model accuracy by 28.5% through automated repair. Future work includes real-time data monitoring, support for unstructured data, integration with data version control systems, and extension to streaming data pipelines.

References

- [1] A. Fariha, A. Tiwari, A. Meliou, and S. Nath, "Conformance Constraint Discovery: Measuring Trust in Data-Driven Systems," Proc. ACM SIGMOD, 2020.
- [2] S. Krishnan, J. Wang, E. Wu, M. J. Franklin, and K. Goldberg, "ActiveClean: Interactive Data Cleaning for Statistical Modeling," Proc. VLDB, 2016.
- [3] S. Schelter et al., "Automating Large-Scale Data Quality Verification," Proc. VLDB, vol. 11, no. 12, 2018.
- [4] N. Hynes, D. Sculley, and M. Terry, "The Data Linter: Lightweight Automated Sanity Checking for ML Data Sets," NIPS MLSys Workshop, 2017.
- [5] T. Rekatsinas, X. Chu, I. F. Ilyas, and C. Ré, "HoloClean: Holistic Data Repairs with Probabilistic Inference," Proc. VLDB, 2017.
- [6] N. Polyzotis, S. Roy, S. E. Whang, and M. Zinkevich, "Data Lifecycle Challenges in Production Machine Learning," ACM SIGMOD Record, 2019.
- [7] S. Grafberger, J. Stoyanovich, and S. Schelter, "mlinspect: Data Distribution Debugging in ML Pipelines," Proc. VLDB, 2021.