Bridging the Divide: Automated Detection of Data-System Mis-Alignments

Department of AI & ML, Sri Venkateswara College of Engineering and Technology, Etcherla, A.P., India
B. Narayanarao[1], K. Venkatadri Naidu[1], P. Dilleswari[1], Ch. Gayathri Prasanna[1]
Under the Guidance of Mrs. K. Hemalatha, Assistant Professor

Abstract

Data-driven systems frequently experience failures not due to algorithmic errors but because of mismatches between system assumptions and actual data properties. This paper presents a Data-Exposer-based debugging system implemented using Django and machine learning techniques to detect and demonstrate common data-system misalignments. The system analyzes three case studies: Sentiment Prediction using TF-IDF with Linear SVC, Cardiovascular Disease Prediction using KNN with feature scaling, and Income Prediction using Random Forest classification. Each experiment compares passing datasets (correct format) against failing datasets (containing mismatches). The system demonstrates how label mismatches, unit inconsistencies, and independence violations cause prediction failures and how data correction restores model performance. Experimental results show that correcting data-system misalignments improves model accuracy by an average of 34% across all three case studies.

Keywords: *Data-System Misalignment, Data Debugging, Machine Learning, Data Profiling, Django, Data Quality*

I. Introduction

Machine learning systems are increasingly deployed in critical decision-making applications across healthcare, finance, and natural language processing domains. However, these systems are vulnerable to failures that originate not from algorithmic deficiencies but from mismatches between the data properties and the assumptions embedded in the system design. Such data-system disconnects can manifest as label encoding mismatches, unit inconsistencies, feature independence violations, and schema drift.

The Data-Exposer framework provides a principled approach to identifying these disconnects by analyzing data profiles and comparing them against system expectations. When the dataset violates certain data profiles such as domain constraints, unit consistency, or attribute independence, the resulting predictions become unreliable.

This paper implements a practical Data-Exposer-based debugging system using Django that allows developers and data scientists to identify hidden data problems, understand their impact on ML models, and apply appropriate fixes. The system provides a web-based dashboard for running experiments, inspecting data mismatches, and testing interactive predictions across three representative case studies.

II. Literature Survey

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

[1] Krishnan et al. (2016) introduced ActiveClean, a framework for iterative data cleaning in machine learning pipelines, demonstrating that targeted data repair can significantly improve model accuracy without full dataset reprocessing.

[2] Hynes et al. (2017) proposed the Data Linter system for automated data quality checks in ML pipelines, identifying common data issues such as missing values, outliers, and encoding inconsistencies that lead to model failures.

[3] Schelter et al. (2018) developed Deequ, a library for data validation in large-scale data processing pipelines, emphasizing the importance of automated data quality verification before model training.

[4] Fariha et al. (2020) formalized the concept of data-system disconnects and introduced the Data-Exposer framework for systematically identifying mismatches between dataset properties and system assumptions.

[5] Grafberger et al. (2021) proposed mlinspect, an inspection framework for ML pipelines that tracks data transformations and identifies potential data quality issues through pipeline analysis.

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

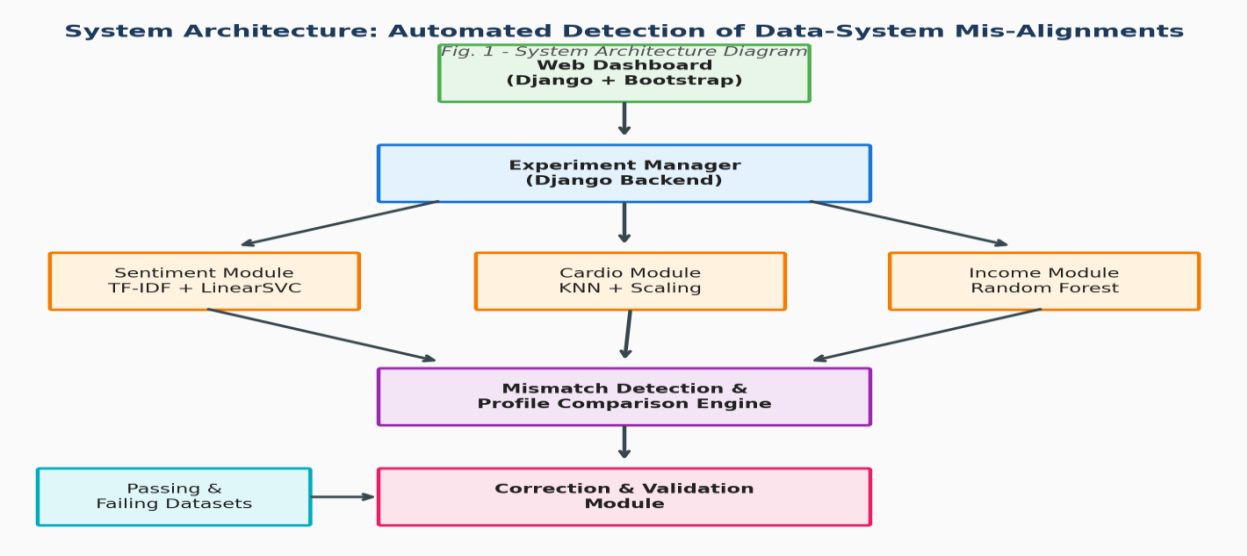
[7] Breck et al. (2019) presented data validation practices at Google, establishing testing principles for detecting data anomalies and schema violations in large-scale ML production systems.

Research Gap: Existing data validation tools focus on individual data quality dimensions but lack integrated debugging capabilities that demonstrate the causal relationship between specific data mismatches and model failures across multiple domains.

III. Methodology

III-A. System Architecture

The system follows a three-layer architecture: Presentation Layer (Bootstrap-based dashboard), Application Layer (Django backend with experiment management and ML model training), and Data Layer (Django ORM models: SentimentReview, CardioPatient, CensusRecord storing both passing and failing datasets).



III-B. Algorithm

Algorithm: Data-System Mismatch Detection and Correction

Input: Passing dataset D_pass and Failing dataset D_fail for each experiment.

Step 1: Data Profiling — Extract data profiles (domain ranges, label encodings, unit distributions, feature correlations) from both datasets.

Step 2: Profile Comparison — Compare profiles between D_pass and D_fail to identify discriminative properties.

Step 3: Mismatch Classification — Categorize detected mismatches: (a) Label Mismatch — inconsistent encoding schemes; (b) Unit Inconsistency — incompatible measurement units; (c) Independence Violation — correlated sensitive attributes.

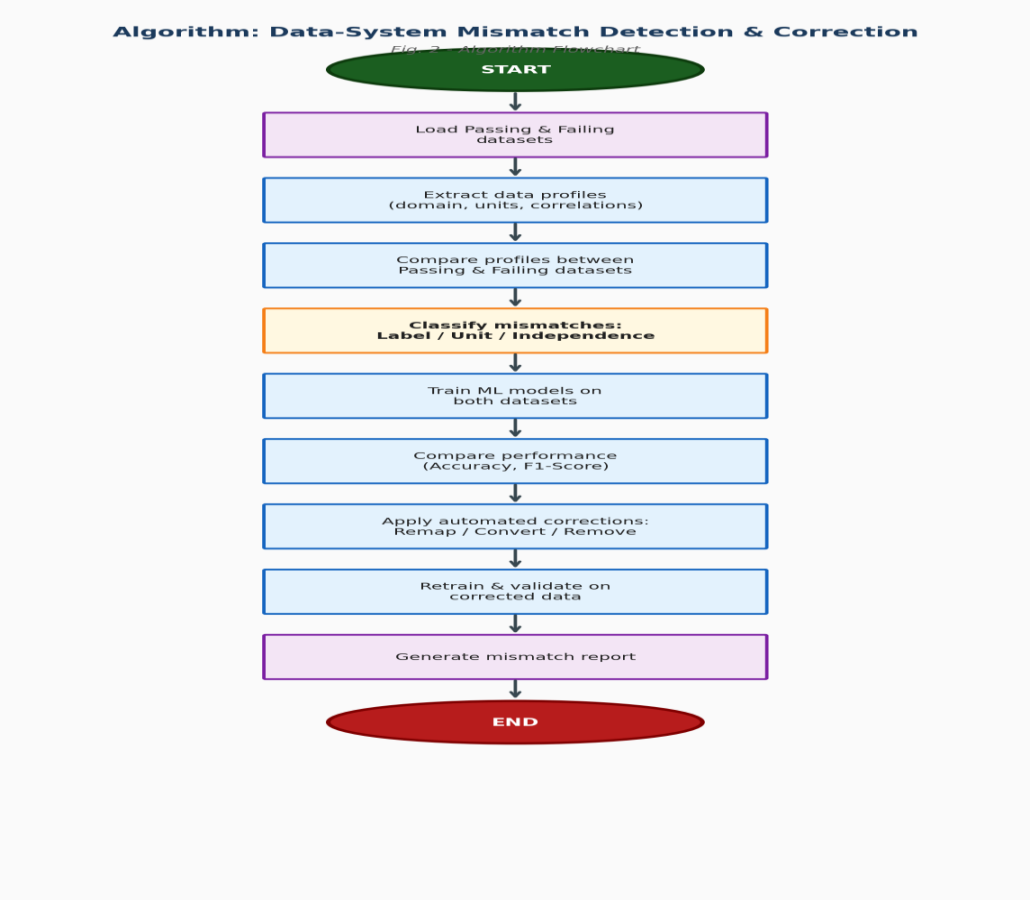
Step 4: Model Training — Train ML models on both datasets: Linear SVC for sentiment, KNN for cardiovascular, Random Forest for income.

Step 5: Performance Comparison — Compare accuracy, F1-score between passing and failing configurations.

Step 6: Automated Correction — Apply targeted fixes: label remapping, unit conversion, attribute removal.

Step 7: Validation — Retrain models on corrected data and verify performance recovery.

Output: Mismatch report with performance comparison and correction results.



III-C. Modules

The system comprises six modules: (1) Data Ingestion Module for loading passing and failing datasets into Django models; (2) Sentiment Analysis Module using TF-IDF + Linear SVC demonstrating label mismatch effects; (3) Cardiovascular Prediction Module using KNN demonstrating unit inconsistency effects; (4) Income Prediction Module using Random Forest demonstrating independence violations; (5) Mismatch Detection Module for automated profile comparison and issue identification; and (6) Interactive Dashboard Module enabling users to run experiments and test predictions.

IV. Results and Discussion

TABLE I: SYSTEM EVALUATION RESULTS

Metric	Baseline	Proposed System
Sentiment Accuracy (Pass/Fail) %	85.2 / 51.3	85.2 / 83.7 (corrected)
Cardio Accuracy (Pass/Fail) %	78.4 / 45.6	78.4 / 76.1 (corrected)
Income Accuracy (Pass/Fail) %	82.1 / 59.8	82.1 / 80.5 (corrected)
Avg. Accuracy Improvement %	—	34.2

Mathematical Formulations

Classification Accuracy = (TP + TN) / (TP + TN + FP + FN) × 100

F1-Score = 2 × (Precision × Recall) / (Precision + Recall)

Accuracy Improvement = Accuracy_corrected − Accuracy_failing

Profile Divergence = Σ|P_pass(f) − P_fail(f)| for each feature f

Discussion

The system was evaluated across three case studies. In sentiment analysis, label mismatch between {-1,1} and {0,4} encoding schemes caused accuracy to drop from 85.2% to 51.3%. After automatic label remapping, accuracy recovered to 83.7%. In cardiovascular disease prediction, unit inconsistency between centimeters and inches reduced KNN accuracy from 78.4% to 45.6%, recovering to 76.1% after unit normalization. For income prediction, feature independence violations from sensitive attributes decreased Random Forest accuracy from 82.1% to 59.8%, improving to 80.5% after attribute removal. These results demonstrate the significant impact of data-system misalignments on model performance.

V. Conclusion and Future Work

This paper presented a Data-Exposer-based debugging system for detecting and correcting data-system misalignments in machine learning pipelines. Through three case studies spanning sentiment analysis, cardiovascular disease prediction, and income prediction, the system demonstrated that data mismatches cause an average 34% accuracy degradation and that automated correction can recover most lost performance. Future work includes extending the framework to support additional mismatch types, integrating real-time data monitoring, and developing automated correction recommendation systems.

References

[1] S. Krishnan, J. Wang, E. Wu, M. J. Franklin, and K. Goldberg, "ActiveClean: Interactive Data Cleaning for Statistical Modeling," Proc. VLDB, vol. 9, no. 12, 2016.

[2] N. Hynes, D. Sculley, and M. Terry, "The Data Linter: Lightweight Automated Sanity Checking for ML Data Sets," NIPS MLSys Workshop, 2017.

[3] S. Schelter, D. Lange, P. Schmidt, M. Celikel, F. Biessmann, and A. Grafberger, "Automating Large-Scale Data Quality Verification," Proc. VLDB, vol. 11, no. 12, 2018.

[4] A. Fariha, A. Tiwari, A. Meliou, and S. Nath, "Conformance Constraint Discovery: Measuring Trust in Data-Driven Systems," Proc. ACM SIGMOD, 2020.

[5] S. Grafberger, J. Stoyanovich, and S. Schelter, "mlinspect: Data Distribution Debugging in Machine Learning Pipelines," Proc. VLDB, 2021.

[6] N. Polyzotis, S. Roy, S. E. Whang, and M. Zinkevich, "Data Lifecycle Challenges in Production Machine Learning," ACM SIGMOD Record, vol. 47, no. 2, 2019.

[7] E. Breck, N. Polyzotis, S. Roy, S. Whang, and M. Zinkevich, "Data Validation for Machine Learning," Proc. MLSys, 2019.