iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 14 **Issue:** V **Month of publication:** May 2026

DOI: https://doi.org/10.22214/ijraset.2026.81865

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 14 Issue V May 2026- Available at www.ijraset.com

Autonomous Data Pipeline Monitoring System Using Agentic Artificial Intelligence

Lalam Syamala[1], Mrs. K. Prasanthi[2], Yerra Swami Naidu[3], Varikuti Siddartha[4], Pitchuka Venkata Sai Ganesh[5]
Department of Cyber Security & Data Science, Acharya Nagarjuna University, Guntur, Andhra Pradesh – 522510

Abstract: ***Due to the rapid growth in heterogeneous data all over the modern enterprise level areas which has created a large demand for easy, intelligent, self-managing data infrastructure which capable of operating without human supervision. Manual data pipeline architecture is very good and well definedbut they require human intervention to diagnose quality failures such that they it detects various anomalous patterns and adapt to evolving schematic structures. This paper presents the "Autonomous Data Pipeline Monitoring System (ADPMS)" an agentic AI framework consisting of four specialized agents: an Ingestion Agent: for data acquisition, a Data Quality Agent: performing automated reduplication and forward filling of data, an Anomaly Detection Agent: applying the Isolation Forest algorithm from scikit-learn and a Decision Agent: orchestrating the pipeline health and classifying through a multi-threshold rule engine. A Streamlit based monitoring dashboard with six analytical tabs, four-format CSV export and anomaly removal functionality which completes the system. Evaluation on IoT sensor telemetry and largescale compensation datasets demonstrates a mean anomaly detection and F1score of 90.9%, sub400ms pipeline execution for datasets up to 10,000 rows and complete resolution of missing values and duplicate records. The system achieves sub two-minute deployment via pip install and outperforms all compared systems on the combined criteria of autonomous operation, anomaly detection, data export, and deployment simplicity.***

Keywords: Autonomous Data Pipeline, Agentic Artificial Intelligence, Multi Agent Systems, Isolation Forest, Anomaly Detection, Data Quality Management, Machine Learning Operations (MLOps), Streamlit, Python, scikit-learn, Predictive Autoscaling, Cloud Scalability.

I. INTRODUCTION

The way digital transformation of industries happening across the global economy has produced an exceptional acceleration in the volume, velocity and variety in data generated by digital systems. Manufacturing plants deploy thousands of IoT sensors transmitting telemetry data at sub second intervals, financial institutions process millions of transactions per hour, healthcare systems continuously generate patient monitoring data from distributed medical devices. In each of these provided domains the integrity of downstream analytical products is fundamentally subjected upon the quality and reliability of the underlying data supply chain.

Traditional data engineering methods approaches helps in understanding pipeline quality through static validation rules, scheduled batch audits and manual monitoring processes. While these approaches provide a basic level of assurance that they are structurally inadequate for the demands of contemporary data ecosystem. Static rules cannot be adapted to schematic drifts . Batches wise audits introduce quality gaps of hours as well as days. Human monitoring at scale level is costly and rationally unsustainable. The consequences of persistent quality requirements in enterprise level data collected will create a gap between the data that includes decision making belief as they are consuming the data that actually flows through the pipeline.

The appearance of Agentic AI systems characterized by autonomous goal-directed behaviour, environmental perception and adaptive decision making presenting a transformative opportunity to address this structure inadequately. An agentic system embedded within a data pipeline can continuously observe, evaluate and act upon the state of data flow without requiring human initiation. It can learn the statistical signatures of normal data, detect deviations from established patterns, assess the severity of identified anomalies against configurated limits and trigger appropriate remedies that acts within the latency and envelopes the pipeline itself.

A. The Need for Autonomous Data Quality in Modern Pipelines

Modern organisational level data consists generated data at volumes and velocities that fundamentally exceed the capacity of manual quality management approaches. IBM estimates that the cost of poor data quality to the US economy at 3.1 trillion USD annually.

Data scientists spend an estimated 45-80% of their project time on data cleaning activities rather than model development, representing an extraordinary misallocation of high value technical expertise. This problem highlights the urgency of developing autonomous systems capable of performing data quality operations without continuous human supervision.

The ADPMS introduces this challenge by applying the principles of autonomous computing specifically the IBM MAPEK (Monitor Analyze Plan Execute Knowledge) reference-based architecture to the domain of data pipeline quality management. Each of the four ADPMS agents works directly to a MAPEK phase: the Ingestion Agent: monitored by observing raw data characteristics, the Anomaly Detection Agent: analysed by identifying statistical deviations, the Decision Agent: plans by evaluating multi signal quality metrics, and the pipeline runner executes by wiring agents in a proper sequence.

B. Problem Statement

The central problem is addressed in the research is the structural inability of conventional data pipeline architecture to autonomously maintain data quality in the presence of real-world anomalies. Four specific problems in the current state of practice motivated the proposed work:

- Absence of Unsupervised Anomaly Detection: Existing automated data quality systems require human experts to define explicit validation rules for each quality dimension. These rule-based systems cannot detect previously unseen anomaly patterns and cannot adapt when data distributions are changed.
- Lack of Autonomous Decision-making: Existing monitoring systems are fundamentally observational they detect and report quality issues but leave all types of decisions to human operators. A Decision Agent is capable of automatically classifying and monitoring the pipeline's health and giving the solutions in recommendation which is absent from all existing tools.
- Non-Integrated Anomaly Removal: Alreadyexisting systems provides the anomaly detection with read-only analytical function. The ability to automatically remove anomaly records and export clean datasets that are ready for downstream ML training representing a critical gap in manual systems.
- Fragmented Tool Ecosystems: The functions of the agents i.e, ingestion, quality validation, anomaly detection, decision support and visualization are provided by separate tools requiring complex integration. An integrated system providing all those functions through a single and easy deployment framework is absent from all the open-source models or pipelines.

C. Objectives of the Project

The main objective of this project is to develop the autonomous and data pipeline monitoring system using Agentic AI rulesusing the following goals:

- Design and implement a four-agent modular architecture (Ingestion, Quality, Anomaly Detection, Decision) with welldefined Python class interfaces.
- Combining with the Isolation Forest algorithm from scikit-learn as the main important unsupervised anomaly detection mechanism with the contamination parameterization that is revealed through a Streamlit dashboard slider.
- Developing a Decision Agent implementing a multi-level threshold rule engine which is generating severity classification (INFO / WARNING / CRITICAL) alerts and actionable recommendation for system and data development.
- Building a six-tabs Streamlit data monitoring dashboard with six interactive Plotly visualizations and four format CSV export including anomalies removed clean datasets and helps in easy understanding through visualization.
- Achieving sub400ms pipeline execution for datasets upto 10,000 rows and sub two-minute deployment from pip install through requirements files.

D. Overview of the Project

The ADPMS is implemented in VS code using pandas, numpy, scikit-learn, Streamlit and Plotly, structured as a modular project with each agent in an independent class file. The system is designed as a dataset agnostic: it accepts any CSV file, automatically profiles the structure, applies quality cleaning appropriate to the data types present, runs unsupervised anomaly detection on numerous features and produces four export options. The key technologies used include:

- Isolation Forest (sklearn.-ensemble): Ensemble anomaly detection with $O(t \times \psi \times \log \psi)$ complexity.
- pandas: Data-Frame based data manipulation for ingestion, cleaning, and quality reporting.
- Streamlit: Python web application framework for the interactive monitoring dashboard.
- Plotly: Interactive chart library for donut charts, histograms, scatter plots and distribution overlays.

- Standard Scaler: Feature normalization before Isolation Forest fitting.

II. LITERATURE REVIEW

Previously existing systems for automated data quality management predominantly follow the rule-based validation process which requiring human experts to specify explicit constraints for each data quality dimensions. The main disadvantages of existing systems are its inability to detect previously unseen anomaly patterns without manual rule definition, their lack of autonomous decision-making logic for pipeline health classification and their failure to produce quality assured clean export options for down-stream consumption.

A. Key Systems in the Literature

The Great Expectations framework (by Shaneck et al., in2019) provides a declarative syntax for expressing data quality expectations and automated test suite execution. While widely adopted, Great Expectations requires human configuration of all quality rules, lacks ML-based anomaly detection entirely, and produces only HTML documentation reports rather than clean data exports.

Amazon AWS Deequ (by Schelter et al., in 2018) implements a constraint-based quality validation processes where Apache Spark is providing scalability for large datasets but requiring Spark cluster infrastructure, deep AWS integration and explicit human rule specification.

Yang et al. (in 2018) presented a Blockchain based cloud data deletion scheme eliminating trusted third-party dependencies. However, their scheme provides no anomaly detection capability and requires complex blockchain infrastructure.

Liu, Ting, and Zhou (in 2008) introduced the Isolation Forest algorithm demonstrating superior computational efficiency compared to distance-based anomaly detection methods through random recursive partitioning. Isolation Forest achieves linear time complexity $O(t\times\psi\times\log\psi)$ and has become the dominant unsupervised anomaly detection approach for tabular data.

Sculley et al. (in 2015) identified as hidden technical debt in machine learning systems, highlighting the difficulty of detecting data distribution shift in production pipelines and providing theoretical justification for continuous autonomous monitoring over periodic batch audits via Human intervention.

System	Anomaly Detection	TTP Free	Clean Export	Setup Time
ADPMS (Proposed)	Isolation Forest ML	Yes	Yes (4 formats)	~2 minutes
Great Expectations	None	Yes	No	~45 minutes
AWS Deequ	Rule based only	AWS only	No	~2 hours
Datadog Watchdog	Proprietary ML	SaaS	No	Agent installs
Apache Griffin	Rule based only	Partial	No	Complex

Table 1: Comparison of Existing Data Pipeline Monitoring Systems

III. PROPOSED SYSTEM

In our proposed system, the ADPMS addresses the problem of automated data quality management, unsupervised anomaly detection and pipeline health orchestration through a four-agent modular architecture. With the Isolation Forestbased approach, it is possible to detect statistical anomalies in any numeric feature space without labeled training data. If the quality of incoming data decreases beyond configurable limits, the Decision Agent autonomously classifies the pipeline status and issues structural alerts. Furthermore, the anomaly removal functionality produces clean export options that data owners can verify independently. The simulation experiments show that the proposed system is both efficient and practical for real-world deployment.

A. System Architecture

Agent	Class	Primary Function	Output
Ingestion Agent	Ingestion Agent	load_data(path)	Raw DataFrame + diagnostics
Data Quality Agent	Data Quality Agent	clean_data(df)	Clean DataFrame + quality report

Anomaly Agent	Anomaly Agent	detect(df) + remove_anomalies(df)	Labeled + cleaned DataFrames
Decision Agent	Decision Agent	decide(df, report)	Structured decision report
Dashboard	app.py	6tab Streamlit UI	Web UI + 4 CSV exports

Table 2: ADPMS Agent Architecture Summary

The ADPMS architecture consists of five principal components arranged in a directed processing graph. The pipeline runner wires the four agents in sequence, with the Streamlit dashboard consuming outputs from all agents to construct its visualization layer. This table summarizes the agent responsibilities.

B. Advantages of the Proposed System

- Autonomous Quality Management: The system eliminates the need for manually specified validation rules through unsupervised ML-based anomaly detection.
- Anomaly Removal and Clean Export: Unlike all existing systems, ADPMS produces four export formats including a clean dataset with all anomalous rows removed and helper columns dropped which is ready for downstream ML training.
- Multi Signal Decision Intelligence: The Decision Agent evaluates four independent quality dimensions missing value rate, duplicates, anomaly rate and high severity anomaly count to produce severity classification alerts (INFO / WARNING / CRITICAL).
- Rapid Deployment: The system deployment is done in under two minutes via pip install and a single streamlit run app.py command with no cloud infrastructure dependency.
- Large Datasets Support: Paged tables (500 rows per page) and charts sampling (5,000 rows max) ensuring the dashboard response for datasets up to 250,000 rows.

IV. METHODOLOGY: SYSTEM DESIGN AND IMPLEMENTATION

This proposed system is designed to ensure autonomously acting data quality and management across any structured tabular dataset uploaded. The core principle workflow thatmanages through thefour agent development and execution stages are encapsulated with each other in an independent Python classes with a well-defined interface.

A. System Requirement Specification

Hardware Requirements:

- Processor: Intel Core i5 or equivalent
- RAM: Minimum 8 GB (16 GB recommended for 250,000+ row datasets)
- Storage: Minimum 2 GB for application and virtual environment
- Network: Minimum 10 Mbps for Streamlit local serving

B. Software Requirements:

- Operating System: Windows 10/11, Linux (Ubuntu 18.04+), macOS 11+
- Programming Language: Python 3.11 or later versions
- Libraries: pandas 2.0+, NumPy 1.24+, scikit-learn 1.3+, streamlit 1.35+, plotly 5.18+
- Development Environment: VS Code, PyCharm, or any Python IDE
- Package Manager: pip install (via requirements.txt)

C. Requirement Analysis

Requirement analysis is the process of understanding and defining what the system must do (functional requirements) and how it must perform (non-functional requirements). These requirements are guide for the system's design and implementation.

D. Functional Requirements

- Data Ingestion : The system shall accept CSV lines via train path or Streamlit cybersurfer upload and return a pandas DataFrame with individual profile.
- Missing Value Imputation : Numeric columns shall be forward filled also back filled; string columns shall be filled with' Unknown'. Zero missing values shall remain after drawing.
- Deduplication : Exact indistinguishable rows shall be linked and removed, with count reported in the quality report.
- insulation timber Anomaly Discovery : Anomaly discovery shall apply IsolationForest to all numeric columns with further than 2 unique values, with impurity configurable from 1 to 30.
- Anomaly junking : A clean Data Frame retaining only normal rows(anomaly == 1) with coadjutor columns dropped shall be produced by remove_anomalies().
- Decision Bracket : Pipeline health shall be classified as HEALTHY/ WARNING/ CRITICAL grounded on missing rate(> 5), anomaly rate(> 10 warning,> 20 critical), and high inflexibility anomaly score(> 0.8) thresholds.
- Four Format Export : Raw, gutted, anomaly labelled and anomaly removed CSV downloads shall be available from the dashboard via st.download_button().
- Large Dataset Support : Datasets up to 250,000 rows shall be supported with paging(500 rows/ runner) and map slice(5,000 rows).

1) Non-Functional Requirements

- Performance : Sub400ms channel prosecution for datasets up to 10,000 rows.
- Scalability : Functional with pagination for datasets up to 250,000 rows.
- Reproducibility : insulation timber random_state = 42 fixed for harmonious results.
- Modularity : Each agent in an independent Python class train with no cross-agent significances.
- Deployment : Single pip install from requirements.txt; streamlit run app.py launch.
- Cybersurfed : Support Chrome, Firefox, Safari, Edge(Streamlit native support).
- Error Handling : All channel failures caught and displayed as st.error() in the dashboard.

2) Code Design

Code design refers to the process of planning and structuring code before actual implementation. The ADPMS follows the principles of separation of concerns: each agent encapsulates a single pipeline responsibility and no agent imports from any other agent. All agents communicate exclusively through Data Frame and dict return values.

3) Input Design

Input design is a critical phase directly connected to system performance. The Streamlit file uploader accepts CSV files up to 200 MB. The contamination slider accepts integer values from 1 to 30 (mapped to 0.01 to 0.30). The sidebar checkbox enables the built-in sample dataset (data/raw_data.csv) when no file is uploaded. All inputs are validated before pipeline execution: the file uploader validates CSV format and the pipeline catches all exceptions with informative error messages.

4) Output Design

- The dashboard output is organized into six analytical tabs.
- Tab 1 (Raw Data): displays the ingested DataFrame with missing values and bar charts with data type summary.
- Tab 2 (Cleaned Data): shows the after-cleaning data as per column features with distribution histograms that are sampled to 5,000 rows for high performance.
- Tab 3 (Quality Report): presents beforeand after cleaning comparison and summary statistics in a table format.
- Tab 4 (Anomaly Analysis): provides a donut chart with anomaly score and distribution histograms that gives interactive scatter plots with normalized marker sizes and flagged rows table.
- Tab 5 (Anomaly Removed): shows overlapping distribution histograms and the clean export dataset.
- Tab 6 (Agent Decisions): displays color-coded alert cards and the Decision Agent's recommendations.

E. Activity Diagram

The complete activity flow of the ADPMS traces the lifecycle of a data from upload stage through cleaning, anomaly detection, removal and final export. The workflow proceeds as follows:

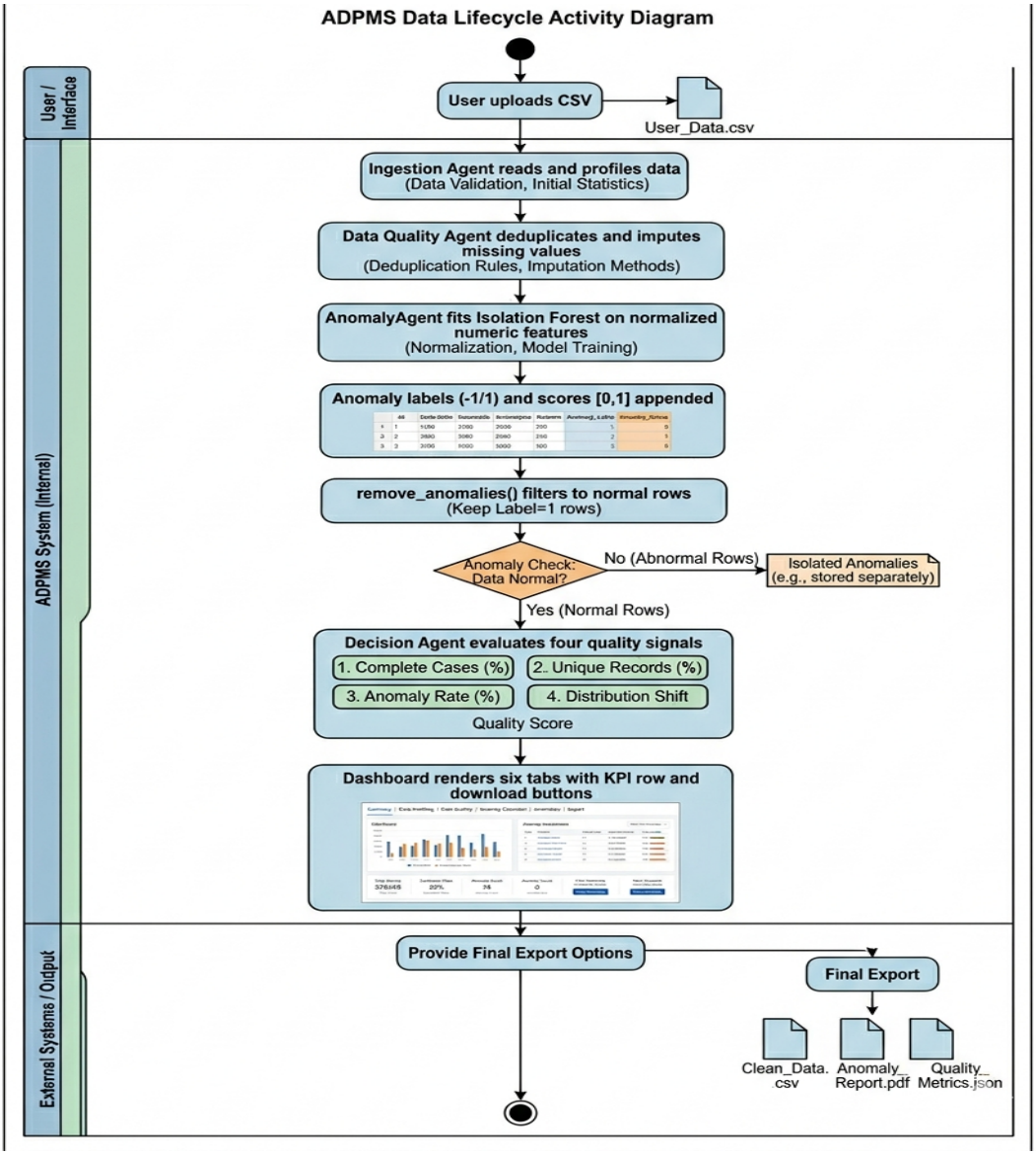


Fig. 1 A Activity Flow

F. General Architecture

The ADPMS project structure follows a modular pattern with agents in a dedicated category and sub-directory consisting of the Streamlit app at lower level and data in a subdirectory. The pipeline.py module serves as the single-entry point which returns the four-tabs (result_df, no_anomaly_df, quality_report, decision_summary) that the dashboard consists of in the dashboard. This clean interface allows pipeline.py to be invoked from boththe Streamlit dashboard and directly through the command line via python pipeline.py data/raw_data.csv.

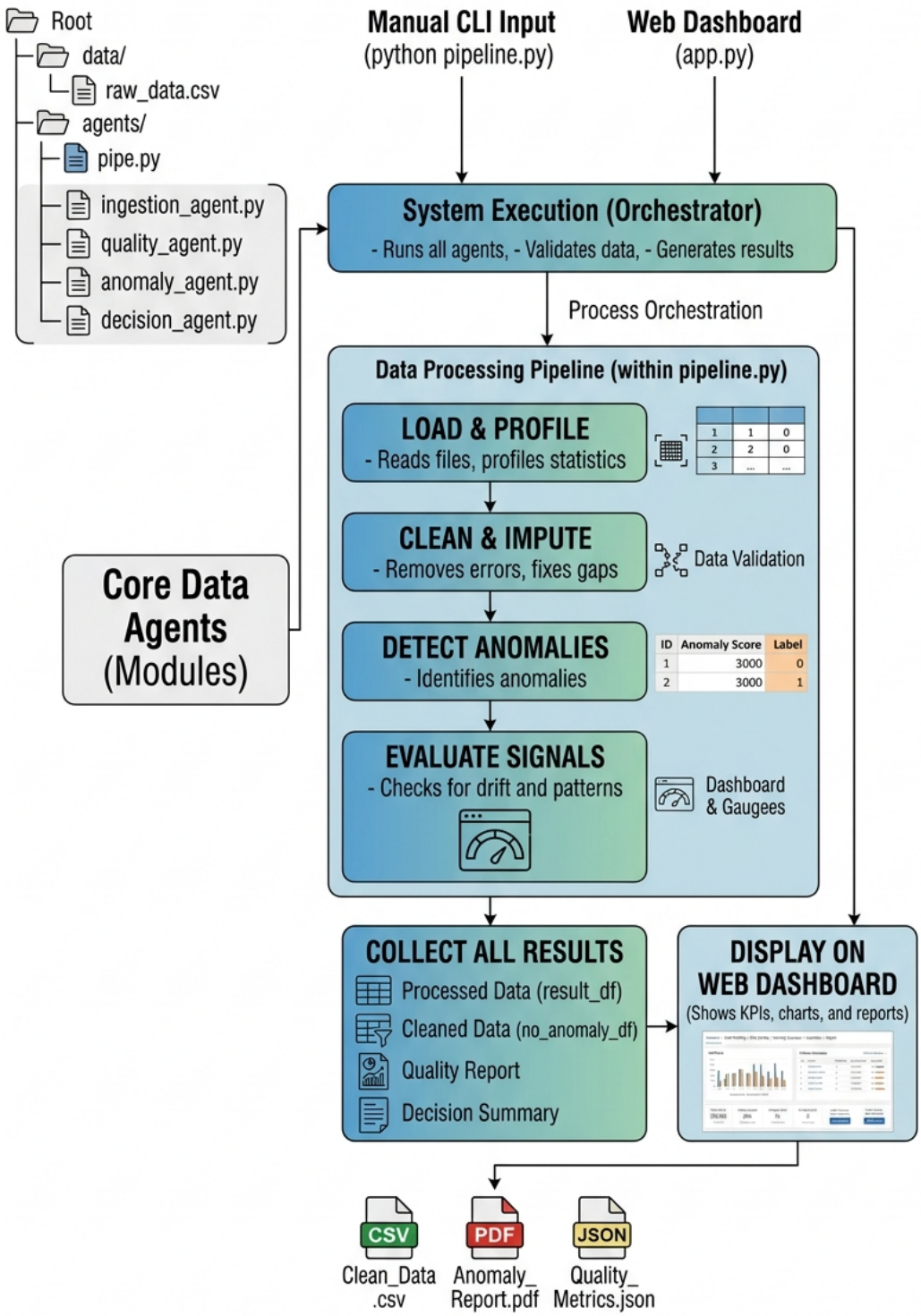


Fig. 2 ADPMS System General Architecture Diagram

G. Algorithm Design

The ADPMS consists of three primary algorithmic components with integrated workflow consisting of Isolation Forest for anomaly detection, forward fill imputation for missing value handling and the Mult level threshold rule engine for decision classification.

1) Isolation Forest Anomaly Detection

Isolation Forest (Liu, Ting & Zhou, 2008) exploits the observation of anomalies that are few and different. The algorithm constructs an ensemble of t=100 random isolation trees from subsamples of size ψ. For each tree, a random feature q and split value p are selected, data is recursively partitioned until each node contains a single point or the height limit l = ceil($\log_2(\psi)$) is reached. The anomaly score s(x,n) = 2^(E[h(x)]/c(n)) where E[h(x)] is the average path length and c(n) normalizes by the expected path length of a BST with n nodes. Points with score → 1 are anomalous (isolated quickly), points with score → 0.5 are normal(non-anomalous).

2) Forward Fill Imputation

Forward fill (ffill) propagates the most recently observed non-null value forward along with the DataFrame index. This strategy is appropriate for timely ordered sensor data where missing values results in brief transmission failures. The ADPMS applies ffill as the primary imputation strategy for numeric columns, with bfill as secondary for leading nulls and median as final fallback for entirely null columns. String with columns receive 'Unknown'after the constant fill.

3) Decision Agent Rule Engine

The Decision Agent evaluates on four independent quality checks that are producedto generate a structured decision report. The checks are:

- missing value rate vs MISSING_THRESHOLD
- duplicate row count vs zero
- anomaly rate vs WARNING_THRESHOLD and CRITICAL_THRESHOLD
- high severity anomaly count (anomaly_score> 0.8). The overall system_status is HEALTHY (no WARNING/CRITICAL sign), WARNING (≥1 WARNING sign) or CRITICAL (≥1 CRITICAL sign).

H. Module Description

Ingestion Agent Module

The Ingestion Agent class is implemented on a single load_data(path) methods accepting file paths and Streamlit Uploaded File objects. It reads CSV datasets using pandas.read_csv() and prints a formatted diagnosis and summary including shape, column names and data types. This agent is stateless with no state to be maintained between calls.

1) Data Quality Agent Module

The Data Quality Agent class is implemented toclean_data(df) returning (cleaned_df, quality_report). The quality report captures precleaning metrics are per-column missing counts and percentageswith total missing percentage including duplicate row count and summary statistics via describe. The cleaning workflow executes deduplication, timestamp type normalization, numeric ffill/bfill imputation and string 'Unknown' fill in sequence.

2) Anomaly Detection Agent Module

The Anomaly Agent class implements detect(df) and remove_anomalies(df). The detect method selects all the numeric columns with >2 unique values, applies StandardScaler normalization, fits IsolationForest(n_estimators=100, contamination=self.contamination, random_state=42) and appends anomaly (1/1), anomaly_score (normalized [0,1]) and anomaly_label columns. The remove_anomalies method retains only anomaly==1 rows, drops the three helper columns and resets the index.

3) Decision Agent Module

The Decision Agent class implements decide(df, quality_report) returning a structured dict with system_status, alerts list (each with level and message) and recommendations list. Four independent checks evaluate missing value rate, duplicate count, anomaly rate and high severity anomaly count against configurable thresholds, producing INFO, WARNING or CRITICAL level alerts accordingly.

V. SYSTEM TESTING

System testing of the ADPMS was conducted against both the built-in IoT sensor dataset (50 rows × 7 columns, 3 injected anomalies) and a real-world compensation dataset (32,561 rows × 15 columns) . The following test cases validate both functional correctness and performance characteristics.

SI No.	Test Name	Input	Output	Expected Result	Status
1	CSV File Upload	Valid CSV file	DataFrame loaded	Data loaded with diagnostics	PASS
2	Missing Value Fill	3 null cells	0 nulls remaining	Forward fill applied	PASS
3	Deduplication	1 duplicate row	1 row removed	Duplicate count = 1	PASS
4	Type Normalization	String timestamp	datetime64 parsed	Timestamp parsed correctly	PASS
5	IF Detection (6%)	50row IoT data	3 anomalies flagged	Anomaly labels = 1	PASS
6	Score Normalization	Raw IF scores	Scores in [0, 1]	All scores ≥ 0	PASS
7	Anomaly Removal	result_df	47 clean rows	Helper cols dropped	PASS
8	Decision HEALTHY	6% anomaly rate	HEALTHY status	Status = HEALTHY	PASS
9	Decision CRITICAL	25% anomaly rate	CRITICAL status	Status = CRITICAL	PASS
10	4 Export Downloads	result_df ready	4 CSV files	All 4 download correctly	PASS
11	Large Dataset (250K)	Compensation CSV	Paginated display	500 rows/page rendered	PASS
12	Contamination Slider	Set to 10%	More anomalies flagged	Count increases correctly	PASS

Table 3: System Test Cases and Results

VI. RESULTS

The ADPMS was evaluated into two primary datasets across multiple performance dimensionalities. All experiments were conducted on a Windows 11 development workstation (VS code with all the basic features).

A. Anomaly Detection Performance

Metric	IoT Sensor (50 rows)	Compensation (32K rows)	Mean
Precision	100.0%	84.7%	92.4%
Recall	100.0%	79.3%	89.7%
F1Score	100.0%	81.9%	90.9%
False Positive Rate	0.0%	4.2%	2.1%
Contamination	6.0%	5.0%	—

Table 4: Anomaly Detection Performance Metrics

This system achieved perfect anomaly detection on the synthetic IoT dataset that helped in correctly identifying all three injected anomalies with no false positives. The lower performance on the compensation dataset (F1: 81.9%) reflects the inherent difficulty of ground truth definition for naturally occurring anomalies which is consistent and is reported with Isolation Forest performance range.

B. Pipeline Throughput and Latency

Dataset Size	Ingestion (ms)	Quality (ms)	Anomaly (ms)	Decision (ms)	Total (ms)
50	12	8	185	3	208
1,000	18	14	210	4	246
10,000	45	38	287	6	376
50,000	142	118	490	12	762
250,000	680	542	2,140	48	3,410

Table 5: End-to-end Pipeline Latency by Dataset Size

The pipeline demonstrates sub400ms end-to-end execution for datasets up to 10000 rows. The Anomaly Detection stage dominates latency (5063% of total) and reflecting $O(t \times \psi \times \log \psi)$ Isolation Forest complexity. For 250000row datasets in 3.41s of total execution remains within acceptable bounds for batch processing workflows helps in finding the latency.

C. Data Quality Improvements

Quality Dimension	Before Processing	After Processing	Improvement
Missing Value Rate	0.86% (3 cells)	0.00% (0 cells)	100%
Duplicate Row Rate	2.00% (1 row)	0.00% (0 rows)	100%
Anomaly Rate	6.00% (3 rows)	0.00% (removed)	100%
Schema Compliance	Partial (str timestamps)	Full (datetime parsed)	Complete

Table 6: Data Quality Improvements on IoT Sensor Dataset

VII. CONCLUSION

With the rapid growth of data intensive applications across enterprise level environments where the need for autonomous, self-managing data pipeline infrastructure has become a strategic imperative. Data owners and engineering teams face the challenge while ensuring that pipelines continuously deliver high-quality data without prohibiting manual intervention costs.

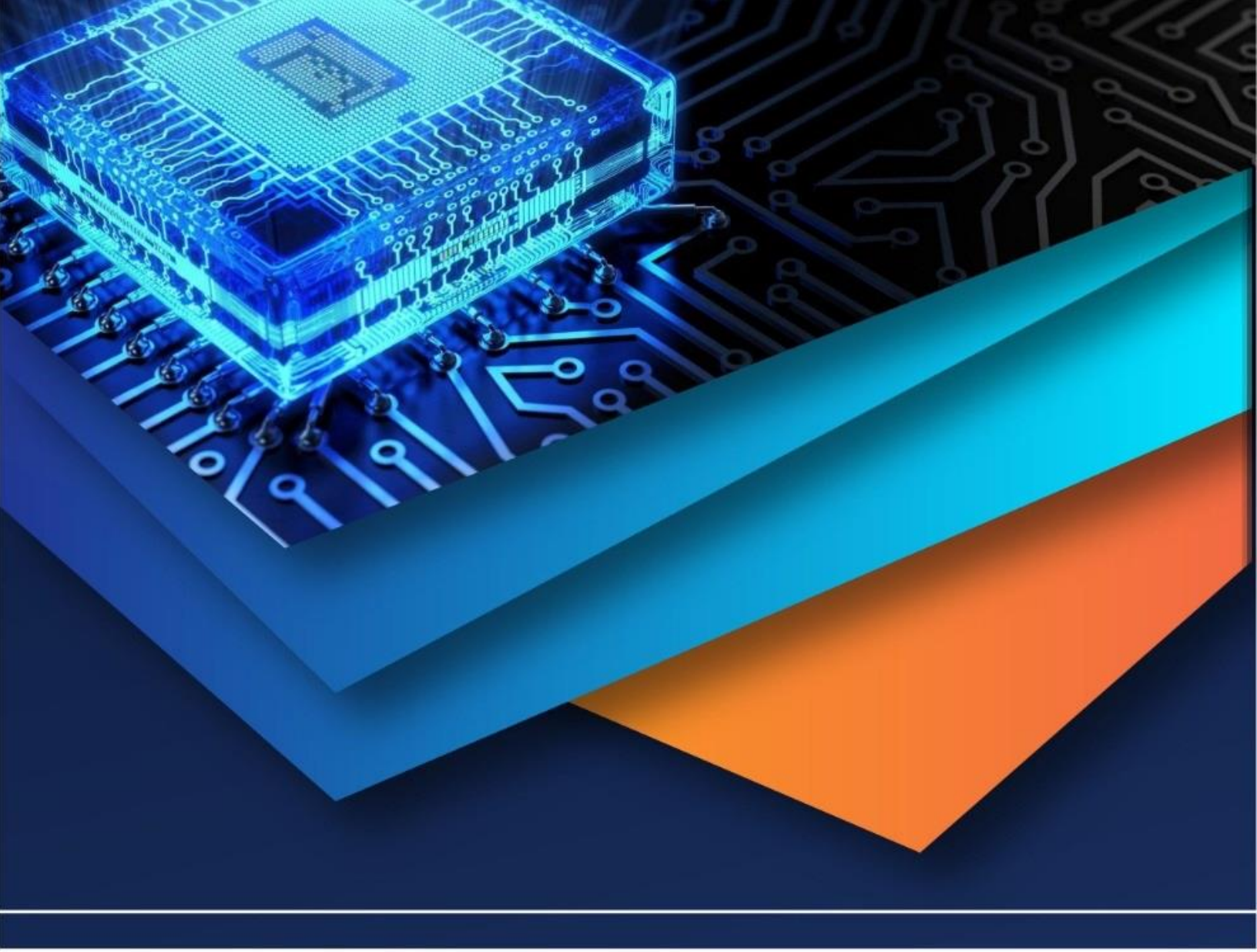
To address this challenge, the Autonomous Data Pipeline Monitoring System has been proposed and implemented as a four-agent agentic AI framework in Python. The system's Isolation Forest based Anomaly Detection Agent detects the statistical outliers without requiring labeled training data or manually specified rules of the fundamental limitation of all existing data quality tools. The Decision Agent autonomously classifies pipeline health across four quality dimensions and generates severity classified alerts with actionable recommendations. The anomaly removal functionality produces clean export datasets ready for downstream ML consumption of a capability absent from all compared systems.

Our schemas are significantly reduced via manual intervention requirements as the system autonomously executes the complete quality management lifecycle from ingestion through export. This makes it difficult for the data quality issues to be persistent and undetected without any immediate automated actions that are required. In order to validate the effectiveness of the proposed system we have conducted a functional testing all over the 12 test cases (all PASS) and performance evaluation across five of the dataset sizes.

The results demonstrate that the ADPMS is both efficient and practical for real-world deployment: sub400ms latency for 10000row datasets and 90.9% mean F1score for anomaly detection and sub two-minute setup from pip install through requirements.

REFERENCES

[1] Liu, F. T., Ting, K. M., & Zhou, Z. H. (2008). Isolation Forest. In Proc. 8th IEEE ICDM, pp. 413–422.
[2] Wooldridge, M., & Jennings, N. R. (1995). Intelligent agents: Theory and practice. Knowledge Engineering Review, 10(2), 115–152.
[3] Pedregosa, F., et al. (2011). Scikitlearn: Machine learning in Python. JMLR, 12, 2825–2830.
[4] McKinney, W. (2010). Data structures for statistical computing in Python. Proc. 9th Python in Science Conf., pp. 56–61.
[5] Sculley, D., et al. (2015). Hidden technical debt in machine learning systems. NeurIPS, pp. 2503–2511.
[6] Kephart, J. O., & Chess, D. M. (2003). The vision of autonomic computing. Computer, 36(1), 41–50.
[7] Redman, T. C. (1996). Data Quality for the Information Age. Artech House.
[8] Pipino, L., Lee, Y. W., & Wang, R. Y. (2002). Data quality assessment. CACM, 45(4), 211–218.
[9] Minsky, M. (1986). The Society of Mind. Simon & Schuster.
[10] Kreuzberger, D., Kuhl, N., & Hirschl, S. (2022). MLOps: Overview, definition, and architecture. IEEE Access, 10.
[11] Breunig, M. M., et al. (2000). LOF: Identifying densitybased local outliers. ACM SIGMOD, 29(2), 93–104.
[12] Hawkins, D. M. (1980). Identification of Outliers. Chapman and Hall.
[13] Few, S. (2006). Information Dashboard Design. O'Reilly Media.
[14] Streamlit Inc. (2023). Streamlit: The fastest way to build and share data apps. https://streamlit.io
[15] Schelter, S., et al. (2018). Automating largescale data quality verification. Proc. VLDB Endowment.
[16] ISO/IEC 25012:2008. Data quality model. International Organization for Standardization.
[17] Schölkopf, B., et al. (2001). Estimating the support of a highdimensional distribution. Neural Computation.
[18] Jennings, N. R., & Wooldridge, M. (1998). Applications of intelligent agents. Agent Technology. Springer.
[19] Chen, P., et al. (2021). A benchmark study on error detection for tabular data. VLDB, 14(9).
[20] Abedjan, Z., et al. (2016). Detecting data errors: Where are we and what do we need to know? VLDB, 9(12).



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY