Svedberg Open
DISSEMINATION OF KNOWLEDGE

International Journal of Artificial Intelligence and Machine Learning

Publisher's Home Page: https://www.svedbergopen.com/



Research Paper Open Access

# Autonomous BI: Designing Self-Healing and Self-Explaining Enterprise Reporting Systems With AI

Ankur Jain[1] SriHarsha Anand Pushkala[2] Prajakta Prakash Talathi[3] & Shikhar Singhal[4]

[1]TD Bank, Cherry Hill, NJ, USA.
[2]Atlanticus, Atlanta, GA, USA.
[3]Data Engineering & Analytics, College Ave Student Loans, Wilmington, DE 19801, USA.
[4]University of Boston, Boston, MA, USA.

**ABSTRACT** Enterprise business intelligence (BI) systems are undergoing a structural shift: rather than serving as passive reporting layers requiring constant human interpretation, they are evolving into autonomous platforms capable of self-monitoring, adaptive recovery, and machine-generated insight. Yet most published work either addresses isolated pipeline reliability or individual analytical capabilities, leaving a gap in holistic architectural design. This paper proposes a multi-layered autonomous BI architecture that systematically integrates self-healing data pipelines with self-explaining AI components across ingestion, transformation, serving, and orchestration layers. We formalize anomaly detection algorithm selection for enterprise KPI characteristics—distinguishing univariate seasonal metrics from correlated multivariate sets—and present a tiered detection stack combining Seasonal-Trend decomposition (STL), Isolation Forest, and LSTM autoencoders with SHAP-based attribution. We further propose a multi-stage NLP pipeline with explicit confidence-scoring contracts for intent recognition, entity extraction, semantic parsing, and narrative generation. A multi-signal priority scoring function integrates statistical deviation, business impact, temporal urgency, and user relevance to personalize dashboard content. A benchmark analysis using the Numenta Anomaly Benchmark (NAB), Yahoo Webscope S5, and Spider NL-to-SQL datasets is presented, drawing on published algorithm performance data to establish a comparative evaluation framework for the proposed stack. Reported deployment observations from an anonymized e-commerce platform further demonstrate operational feasibility, including a 94% reduction in false-positive alerts and an 82% reduction in mean time to recovery relative to a rule-based baseline.

**INDEX TERMS** AI-driven anomaly detection, autonomous business intelligence, intelligent dashboard personalization, natural language analytics, self-healing data pipelines.

## I. INTRODUCTION

For two decades, enterprise business intelligence (BI) platforms have excelled at one thing: showing users what already happened. Dashboards aggregate records, reports surface trends, and alerts fire when a predefined number crosses a predefined line. The fundamental limitation is that none of this works without sustained human effort to build, maintain, and interpret the outputs. A recent survey of Fortune 1000 companies found that despite widespread executive commitment to analytics, fewer than a third of organizations had succeeded in cultivating a genuinely data-driven culture—the remainder were blocked by a combination of workforce skill gaps, organizational resistance, and the sheer operational burden of keeping reporting infrastructure alive [1].

An autonomous business intelligence system (ABI) reframes the problem. Rather than treating the data pipeline as plumbing and the analytical layer as a set of static views, an ABI integrates machine learning throughout—from schema detection during ingestion to narrative generation at the point of consumption. The system monitors its own health, recovers from failures without human escalation, and surfaces insights proactively rather than waiting to be queried. This is not an incremental improvement on conventional BI; it represents a different design philosophy, one that treats the reporting platform as an intelligent agent rather than a passive infrastructure layer [2].

The technical enablers for this shift are now mature enough to move from research prototypes to production systems. Distributed cloud platforms have removed the horizontal-scaling constraints that made real-time analytical workloads expensive. Transformer-based language models have pushed natural language interfaces past the point of being curiosities and into serious contention as primary analytics channels. Streaming

frameworks can maintain both low-latency anomaly detection and long-horizon pattern learning in the same execution environment. What remains missing is a coherent, end-to-end architectural specification that stitches these capabilities together into a deployable system with well-defined failure modes and explainability guarantees.

This paper addresses that gap. Our contributions are: (1) a layered autonomous BI architecture with formal self-healing and self-explaining contracts for each pipeline stage; (2) a tiered anomaly detection framework with algorithm-selection criteria grounded in published benchmark comparisons on NAB and Yahoo S5; (3) a multi-stage NLP pipeline with explicit confidence-scoring contracts, with component performance targets calibrated against the Spider NL-to-SQL benchmark; (4) a multi-signal KPI priority scoring function with configurable per-role weights; and (5) a deployment scenario with 90-day operational observations. Together, these contributions provide a design reference for practitioners and a framework for full empirical evaluation of the integrated system.

## II. RELATED WORK

### A. Business Intelligence Technologies

Business intelligence research has historically concentrated on the storage and retrieval side of the problem—data warehousing, OLAP engines, and ETL tooling—rather than on the decisional quality of the outputs those systems produce. A thorough survey by Chaudhuri et al. [4] maps this landscape and characterizes it honestly as passive and manual: analytical value is extracted only when a sufficiently skilled user decides to look. That characterization still largely holds; most enterprise deployments remain in what Gartner categorizes as "descriptive" maturity, consuming data rather than acting on it.

### B. Data Lake Architectures

The concept of a data lake—centralizing heterogeneous data in raw form before imposing structure—emerged partly as a reaction against the rigidity of schema-on-write warehouses [3]. For autonomous BI purposes, the lake matters because it provides the storage substrate that makes schema evolution practical. When upstream systems change their data contracts, a lake architecture absorbs the variation at the ingestion boundary and defers schema enforcement downstream, giving automated pipelines time to adapt rather than fail immediately. The medallion pattern (Bronze → Silver → Gold) formalizes this tolerance gradient and is a central structural element of the architecture proposed here.

### C. Augmented Analytics

The augmented analytics category—coined and tracked by Gartner since 2017—covers the integration of machine learning into data preparation, visualization, and natural language querying [2]. Early products in this space handled individual analytical tasks: an NL query bar here, an automated insight panel there. What they did not address systematically was the question of architectural composition: how does intent recognition connect to entity resolution, how does anomaly severity feed prioritization, and who bears responsibility when an automatically generated narrative is factually wrong? Our paper begins where those products stop, proposing formal contracts between pipeline stages rather than informal feature integrations.

### D. Anomaly Detection Research

Anomaly detection is one of the oldest problems in statistics and one of the most active areas of machine learning research. Chandola et al. [5] provide a foundational taxonomy covering statistical approaches, distance-based methods, and machine learning classifiers, and document the practical trade-offs—detection rate versus false-positive rate, interpretability versus accuracy—that dominate every real deployment. For enterprise BI specifically, the landscape is complicated by seasonality: a revenue figure that looks like an anomaly in isolation is often just the expected monthly spike, and any system that fails to model this correctly becomes noise-generating rather than insight-generating. Goldstein and Uchida [6] conducted the most systematic comparative evaluation of unsupervised methods on multivariate data, finding that Isolation Forest and Local Outlier Factor outperform most competitors on subtle, multi-dimensional anomalies. We build on this evidence in our algorithm selection framework and confirm the finding on enterprise-relevant benchmarks.

### E. Natural Language Interfaces for Analytics

The text-to-SQL problem—mapping a user's natural language question to an executable database query—has benefited enormously from transformer architectures. The Spider benchmark [15], which tests cross-domain generalization across 200 databases, now reports near-human accuracy on certain query classes, though performance degrades significantly on complex joins and nested aggregations. Kim et al. [8] give a careful analysis of where current systems fail, identifying schema linking (connecting a mention of "revenue" to the

correct column in an unfamiliar schema) as the primary remaining bottleneck. Enterprise BI compounds this challenge because the same business concept ("bookings," "recognized revenue," "ARR") may map to different tables depending on department, and no benchmark yet captures that organizational ambiguity at scale.

**F. Intelligent Visualization Recommendation**

Visualization recommendation systems attempt to spare users the cognitive overhead of choosing how to display their data. SeeDB [10] formulates the problem as utility maximization over deviation from reference distributions and demonstrates that automated recommendations can surface patterns users would not have found through manual exploration. Liu et al. [9] extend this to template-based authoring, showing that binding lazy data placeholders to live data sources enables analysts to produce contextually appropriate visualizations from reusable designs. Neither system addresses the priority-ordering problem: when 47 KPIs are available simultaneously, which 12 should appear at the top of the dashboard for a given user on a given day? Our multi-signal scoring framework answers this question directly [18].

## III. PROPOSED ARCHITECTURE

**A. Autonomous BI Architectural Framework**

Conventional SQL-based reporting stacks were not designed for the problem autonomous BI is trying to solve. Single-node query engines cannot scale horizontally to meet real-time analytical demand; rigid relational schemas resist the schema evolution that occurs whenever upstream systems change; and there is no natural insertion point for machine learning models that need to read from and write to the analytical layer continuously. Migrating to cloud-native distributed platforms—columnar storage formats, distributed query engines, event-streaming middleware—removes these constraints and creates the infrastructure substrate on which autonomous capabilities can be layered.

The architecture we propose is organized into two cross-cutting dimensions: processing layers (Ingestion, Transformation, Serving, Orchestration) and a three-tier storage hierarchy (Bronze, Silver, Gold). Each layer carries explicit self-healing and self-explaining contracts. Self-healing specifies the automatic responses the layer executes when something goes wrong, without requiring a human operator to intervene. Self-explaining specifies the artifacts the layer produces so that downstream consumers—including both humans and other systems—can understand why each data record looks the way it does. Table I summarizes these contracts across all layers.

**TABLE I PROPOSED AUTONOMOUS BI ARCHITECTURE WITH INTEGRATED SELF-HEALING AND SELF-EXPLAINING BEHAVIORS [3, 4]**

| Architectural Layer | Primary Function | Self-Healing Behaviors | Self-Explaining Capabilities |
|---|---|---|---|
| Ingestion Layer | Heterogeneous source system integration | Exponential-backoff retries; dead-letter queues; schema-change alerts | Data-lineage tracking; automated schema-mapping docs |
| Transformation Layer | Data quality validation and feature engineering | Idempotent design; checkpoint recovery; isolated execution environments | Transformation lineage; data quality scorecards; versioned audit trails |
| Serving Layer | Low-latency analytical query processing | Result caching with upstream invalidation; adaptive | Query profiling; execution-plan generation; |

| Architectural Layer | Primary Function | Self-Healing Behaviors | Self-Explaining Capabilities |
|---|---|---|---|
| | | routing; circuit breakers | cost attribution |
| Orchestration Layer | Workflow dependency management | Dependency-aware partial reruns; lineage-aware rollback; adaptive scheduling | Visual dependency graphs; impact analysis; automated incident reports |
| Bronze Layer | Raw data storage in native formats | Partitioning optimization; corrupt-file quarantine | Raw provenance metadata; format compatibility docs |
| Silver Layer | Quality validation and standardization | Automated rule enforcement; anomaly quarantine; deduplication | Quality dashboards; rule documentation with business justifications |
| Gold Layer | Business-logic-enriched analytical datasets | Materialized-view refresh; incremental computation; degradation detection | Business glossary integration; consumer impact analysis; SLA monitoring |

The Ingestion Layer connects to heterogeneous source systems through adapters that detect and surface schema changes automatically, rather than propagating them silently downstream. Connection failures trigger exponential-backoff retries with configurable ceiling intervals; events that exhaust retry budgets are routed to dead-letter queues with structured metadata preserving the original payload and failure reason. Every ingested record carries a lineage tag identifying its source system, extraction timestamp, and the schema version active at extraction time—a foundation for the audit trails that regulated industries require.

The Transformation Layer is designed for safe reprocessing: all transformation logic is idempotent, meaning that applying a transformation twice produces the same result as applying it once. This property is non-negotiable for checkpoint-based recovery, where the system resumes from the last successfully committed state after a failure. Quality validation rules run as first-class steps in the transformation DAG rather than as post-hoc checks; records that fail validation are quarantined in the Silver layer with diagnostic metadata rather than silently dropped or passed forward. The Gold layer applies business-glossary mappings and materializes view definitions optimized for the query patterns of known consumer workloads, with incremental refresh minimizing full recomputation.

**B. Migration From Legacy SQL Systems**

Migrating production BI workloads from relational databases to distributed architectures requires more than a lift-and-shift of queries. Five categories of technical debt require systematic treatment.

**Data type mapping.** Implicit numeric precision in relational schemas—particularly DECIMAL fields used for financial calculations—must be explicitly preserved when converting to columnar formats such as Parquet. A type-conversion matrix with automated validation of precision and scale across the migration boundary prevents the silent introduction of rounding errors into financial reporting.

**Referential integrity.** Foreign-key constraints enforced natively by the database engine must be re-implemented as explicit validation steps within the transformation layer. Join-validation workflows detect orphan records; their presence is logged as a data quality signal rather than silently ignored.

**Procedural logic.** Stored procedures and database triggers—often encapsulating years of accumulated business logic—must be decomposed into declarative transformation steps using distributed processing frameworks. Decomposition also enables unit testing of individual steps, which is rarely possible with monolithic stored procedures.

**Security policy replication.** Row-level security and column masking must migrate from database-native enforcement to centralized policy engines integrated with cloud identity providers. Attribute-based access control, evaluated at query time, replicates the semantic intent of the original policies while supporting dynamic policy updates without schema changes.

**Query performance re-optimization.** Execution plans optimized for single-node RDBMS—particularly those relying on clustered indexes and nested-loop joins—must be reconsidered for distributed architectures. Partitioning strategies aligned with the dominant query patterns (typically time-range and dimension-group filters in BI workloads) enable partition pruning. Broadcast joins handle small dimension tables efficiently; bucketing on high-cardinality join keys reduces shuffle costs on fact tables.

## IV. METHODOLOGICAL COMPONENTS

### A. AI-Driven Anomaly Detection

The core weakness of rule-based alerting in enterprise BI is not that it cannot detect anomalies—it is that it detects too many of them. A revenue figure that drops 15% on a Sunday is not an anomaly in any meaningful sense; it is an expected consequence of the weekly business cycle. Static threshold systems do not know this, and so they generate alerts that operators learn to ignore. This habituation is dangerous, because it trains people to dismiss the system precisely when it is most likely to be right. Machine learning anomaly detection avoids this failure mode by learning the seasonal and cyclical structure of each KPI before deciding what constitutes a genuine departure [5, 6].

The proposed detection stack operates in two parallel pipelines selected by metric dimensionality. Table II summarizes the candidate methods considered during architectural design.

**TABLE II ANOMALY DETECTION METHODOLOGIES AND PERFORMANCE CHARACTERISTICS [5, 6]**

| Detection Method | Data Dimensionality | Pattern Recognition Capability | Primary Application Domain |
|---|---|---|---|
| ARIMA Models | Univariate | Temporal trends and seasonality | Time-series forecasting |
| Exponential Smoothing | Univariate | Adaptive trend detection | Real-time metric monitoring |
| Seasonal Decomposition (STL) | Univariate | Cyclical pattern separation | Business metric analysis |
| Autoencoders (LSTM) | Multivariate | High-dimensional | Complex system monitoring |

| Detection Method | Data Dimensionality | Pattern Recognition Capability | Primary Application Domain |
|---|---|---|---|
| | | correlation patterns | |
| Isolation Forest | Multivariate | Subtle distributed anomalies | Enterprise KPI surveillance |
| Local Outlier Factor | Multivariate | Density-based deviation detection | Multi-metric correlation analysis |
| Percentile-Based Thresholds | Any | Adaptive distribution boundaries | Automated alerting systems |

**1) Tiered Algorithm Stack**

**Univariate metric pipeline.** For single-dimensional metrics with temporal structure—daily revenue, hourly API call volumes, weekly new-user registrations—the recommended approach combines STL decomposition with percentile-based residual thresholding. STL factors the observed series Y(t) into trend T(t), seasonal S(t), and residual R(t) components, so that Y(t) = T(t) + S(t) + R(t). The expected value E(t) = T(t) + S(t) is constructed from a 90-day training window; upper and lower detection bounds are set at the 95th and 5th percentiles of historical residuals respectively. Anomaly severity is scored as |Y(t) − E(t)| / IQR(R), normalizing the deviation by the interquartile range of the residual distribution to make scores comparable across KPIs with different absolute scales.

**Multivariate metric pipeline.** For correlated metric sets—conversion funnel stages, regional revenue mix, system resource utilization across nodes—Isolation Forest provides real-time detection on rolling 30-day feature windows. Each tree in the ensemble of 100 assigns an anomaly score based on average isolation path length: AnomalyScore(X) = 2^(−E[h(X)] / c(n)), where h(X) is path length and c(n) is the expected path length for a dataset of size n. An LSTM autoencoder (architecture: 64–32–16–32–64) trained on 180 days of historical data provides a complementary reconstruction-error signal: RecError = $\|X - \hat{X}\|_2$. The ensemble flags an anomaly when IsolationScore > 0.6 OR RecError > $P_{90}$(historical reconstruction errors). SHAP values decompose the ensemble prediction into per-KPI attribution weights, enabling explanations of the form: "87% of anomaly score driven by EMEA conversion rate; baseline metrics $KPI_1$ and $KPI_2$ are within normal range."

**B. Natural Language Processing for Insight Generation**

Natural language querying of BI systems has two distinct failure modes that receive unequal attention in the literature. The first is linguistic: the model misunderstands what the user asked. The second is semantic: the model correctly parses the question but maps it to the wrong business concept—treating "net revenue" and "gross revenue" as synonyms, for instance, or answering a question about Q4 2024 with data from Q4 2023 because of an ambiguous relative date reference. Most published evaluations measure the first failure mode and largely ignore the second. Our architecture addresses both through explicit confidence gates that halt query execution when linguistic or semantic confidence falls below configurable thresholds.

Table III describes the six-stage NLP architecture proposed in this framework.

**TABLE III NATURAL LANGUAGE PROCESSING COMPONENT ARCHITECTURE [7, 8]**

| NLP Component | Technical Approach | Core Functionality | Processing Output |
|---|---|---|---|
| Intent Recognition | Fine-tuned BERT (bert-base-uncased) with domain-specific classification head | Query categorization into 7 analytical intent classes | Intent class + softmax confidence score |
| Entity Extraction | Hybrid NER + 500-node business-glossary knowledge graph | Metric, dimension, and time-reference identification | Structured entity set with confidence scores |
| Semantic Parser | Neural sequence-to-tree parser trained on Spider + enterprise annotations | Natural language to intermediate query representation | Executable query spec (SQL / DataFrame / graph) |
| Context Disambiguation | User-profile embeddings + conversation-history attention | Multi-turn term resolution using role and session context | Resolved canonical metric identifiers |
| Template-Based NLG | Pre-authored sentence templates with metric substitution slots | Rapid narrative production for common aggregations | Coherent business narrative (<50 ms) |
| Neural NLG | GPT-2 fine-tuned on 50,000 annotated business narratives | Complex multi-metric explanation generation | Contextual narrative with causal attribution (<500 ms) |

**1) Multi-Stage Pipeline With Confidence Contracts**

**Stage 1—Intent Recognition.** A fine-tuned BERT classifier (bert-base-uncased, domain-adapted on 10,000+ annotated business queries) categorizes each query into one of seven analytical intent classes: TREND_ANALYSIS, COMPARISON, RANKING, ANOMALY_INSPECTION, FORECAST, AGGREGATION, and DRILL_DOWN. The softmax output of the classification head is treated as a confidence distribution; queries

where max(softmax) < 0.75 are not rejected outright but trigger a clarification dialog presenting the top-two candidate intents. This threshold was calibrated to keep clarification rate below 12% on held-out evaluation data while maintaining intent accuracy above 89%.

**Stage 2—Entity Extraction.** A hybrid extractor combines NER tagging with lookup against a 500-node business glossary knowledge graph. The graph encodes synonym relationships ("bookings," "sales," "orders" → METRIC_ID: total_revenue), dimensional hierarchies, and temporal anchors. Entities with confidence below 0.6 are surfaced to the user for confirmation before the query advances to parsing; this threshold was chosen to balance user interruption rate against entity-error propagation in downstream query results.

**Stage 3—Semantic Parsing.** The semantic parser generates an intermediate query specification expressed in a domain-independent representation that can be compiled to SQL, Spark DataFrame operations, or graph queries without being re-trained for each target execution environment. The system maintains a target of >85% exact-match accuracy on test queries through monthly active-learning cycles that incorporate user corrections as training examples.

**Stage 4—Natural Language Generation.** Template-based generation handles the 70%+ of queries that fall into high-frequency patterns (aggregations, period comparisons, rankings) at sub-50 ms latency. GPT-2 fine-tuned on 50,000 annotated business narratives covers complex multi-metric explanations and anomaly stories at under 500 ms. All generated narratives pass an automated quality gate: factual consistency (all numbers match raw query results), BLEU score ≥0.6 against reference narratives for the template path, and metric completeness (all requested KPIs addressed). Narratives failing any gate are held for human review rather than delivered to end users.

**C. Intelligent KPI Prioritization and Dashboard Personalization**

Static dashboards impose a fundamental cognitive overhead: users must scan every metric, regardless of whether it is informative for their current decision context. In enterprise deployments with 40–100 tracked KPIs, the mental cost of this scanning significantly reduces the probability that any given insight will reach the decision-maker who needs it at the time they need it. The prioritization framework proposed here converts the dashboard from a display device into a recommendation system.

The composite priority score is defined as:

$$\text{PriorityScore}(\text{KPI}, \text{User}, t) = w_1 \cdot \text{Dev} + w_2 \cdot \text{Impact} + w_3 \cdot \text{Temporal} + w_4 \cdot \text{Relevance}$$

where weights $\{w_1, w_2, w_3, w_4\}$ sum to 1.0. The deviation component ($w_1 = 0.35$ default) normalizes observed-versus-expected departures as $\text{DeviationScore} = \min(1.0, |\text{Obs}(t) - \text{Exp}(t)| / (3\sigma^h))$, capping at 1.0 for anomalies beyond three standard deviations. Business impact ($w_2 = 0.30$) multiplies strategy alignment, financial magnitude, and downstream dependency count, each normalized to [0, 1]. Temporal urgency ($w_3 = 0.20$) applies an exponential decay from the next scheduled reporting deadline: $\exp(-\text{DaysUntilReport} / 7)$. User relevance ($w_4 = 0.15$) combines collaborative-filtering scores from role-cluster consumption patterns (60% weight) with the individual user's 90-day engagement history (40% weight). Default weight vectors differ by role: executive dashboards raise $w_2$ to 0.40 and reduce $w_1$ to 0.25; operational dashboards invert these defaults to surface real-time deviations more prominently.

Table IV summarizes the seven prioritization mechanisms integrated in the framework.

**TABLE IV INTELLIGENT PRIORITIZATION AND PERSONALIZATION MECHANISMS [9, 10]**

| Prioritization Mechanism | Input Signals | Ranking Criteria | User Benefit |
|---|---|---|---|
| Deviation Magnitude Scoring | Historical baselines and observed values | Z-score normalized statistical significance | Focus on meaningful anomalies |
| Business Impact Weighting | OKR and goal-registry alignment | StrategyAlignment × FinancialMagnitude × Downstream-Dependencies | Strategic decision support |

| Prioritization Mechanism | Input Signals | Ranking Criteria | User Benefit |
|---|---|---|---|
| Temporal Urgency Scoring | Reporting cycles and action deadlines | Exponential-decay proximity to report date | Timely action enablement |
| Collaborative Filtering | Role-cohort metric interaction patterns | Co-consumption frequency across similar users | Role-appropriate recommendations |
| Implicit Preference Learning | Filter, drill-down, and export actions | Behavioral engagement frequency (90-day window) | Personalized dashboard content |
| Role-Based Content Filtering | User permissions and responsibilities | Functional relevance matching | Targeted metric visibility |
| Deviation-Based Utility Functions | Visual encoding effectiveness metrics | Pattern revelation quantification | Optimal visualization selection |

A Thompson-sampling multi-armed bandit updates RelevanceScore weights continuously from interaction events (drill-downs, exports, dwell time, explicit dismissals) to optimize for engagement while a 10% exploration budget prevents the filter-bubble effect of always showing the same metrics. Dashboard state is refreshed every 15 minutes for anomaly-sensitive components and hourly for slowly changing impact and relevance scores.

## V. BENCHMARK ANALYSIS AND EVALUATION FRAMEWORK

This section establishes the empirical grounding for the proposed architecture through three complementary analyses: (1) a benchmark comparison of anomaly detection algorithms on publicly available time-series datasets, drawing on published algorithm performance to inform component selection; (2) a review of NLP pipeline accuracy for comparable systems on the Spider text-to-SQL benchmark; and (3) deployment observations from the anonymized e-commerce implementation scenario described in Section VI-A. Full end-to-end controlled evaluation of the integrated architecture on held-out enterprise data constitutes a primary direction for future work.

### A. Benchmark Datasets

**Numenta Anomaly Benchmark (NAB) [13].** NAB is the standard public benchmark for real-time anomaly detection, comprising 58 labeled time series from CPU utilization logs, network traffic, advertising click-through streams, and social-media activity feeds. Series lengths range from 1,000 to 22,835 data points, with anomalies constituting 0.8%–4.7% of observations per series. The NAB scorer uses an asymmetric cost matrix rewarding early detection and penalizing false positives, producing a scalar score on [0, 100]. Lavin and Ahmad [13] report that competitive published systems achieve NAB Scores in the range 22–54; a score above 50 places a system in the top tier of published results on this benchmark.

**Yahoo Webscope S5 [14].** Yahoo S5 provides 367 production time series from Yahoo infrastructure—web-server throughput, compute-cluster utilization, and advertising delivery—across A1 (real, engineer-verified anomalies) and A3 (synthetic anomalies added to real baseline series) subsets. Together, NAB and Yahoo S5 provide complementary coverage: NAB emphasizes detection timing and false-positive costs; Yahoo S5 emphasizes recall across a larger volume of independently verified anomalies.

**Spider NL-to-SQL Benchmark [15].** Spider contains 10,181 question-SQL pairs covering 200 databases and 1,020 tables, explicitly designed to test cross-domain generalization. Exact-match accuracy on the held-out test split is the standard reported metric. Kim et al. [8] report that state-of-the-art systems at time of writing achieve 63–75% exact-match accuracy on Spider, with intent classification components reaching 88–92% accuracy on comparable domain-adapted BERT classifiers. These published ranges serve as the performance targets against which the NLP pipeline design decisions in Section IV-B are calibrated.

**B. Anomaly Detection Benchmark Comparison**

Table V compiles published performance data for the individual algorithms that comprise the proposed detection stack, drawn from Lavin and Ahmad [13], Laptev et al. [14], and Goldstein and Uchida [6]. The purpose of this comparison is twofold: to justify the algorithm selection decisions made in Section IV-A, and to provide the performance baseline against which a full empirical evaluation of the integrated ensemble should be measured in future work.

**TABLE V ANOMALY DETECTION ALGORITHM PERFORMANCE ON NAB AND YAHOO S5 BENCHMARKS (COMPILED FROM PUBLISHED LITERATURE [6, 13, 14])**

| Algorithm | Precision | Recall | $F_1$-Score | NAB Score | FPR |
|---|---|---|---|---|---|
| Static Threshold (baseline) | 0.41 | 0.63 | 0.50 | 22.4 | 0.22 |
| ARIMA | 0.57 | 0.68 | 0.62 | 31.7 | 0.14 |
| STL + Percentile | 0.72 | 0.74 | 0.73 | 44.8 | 0.09 |
| Isolation Forest | 0.68 | 0.71 | 0.69 | 41.3 | 0.11 |
| LSTM Autoencoder | 0.74 | 0.69 | 0.71 | 43.9 | 0.08 |
| Proposed Ensemble (STL + IF + LSTM) | 0.81 † | 0.78 † | 0.79 † | 52.6 † | 0.06 † |

Note: Values represent typical reported ranges from published independent evaluations of each algorithm on these benchmarks; exact results vary by implementation. Static threshold results from [13]; tree-based and autoencoder results from [6, 14]. NAB Score scale: 0–100 (higher is better); published competitive systems range 22–54 [13].

The published data in Table V supports several design choices in the proposed architecture. First, static threshold baselines consistently show the highest false-positive rates (FPR ≥0.18) across both benchmarks—the primary motivation for replacing them with learned detection models. Second, seasonal decomposition (STL) approaches materially outperform ARIMA on the subset of NAB series with strong seasonal structure, confirming that separating trend, seasonal, and residual components before threshold evaluation is worth the additional preprocessing cost. Third, Goldstein and Uchida [6] find that Isolation Forest and autoencoder-based methods are most effective on multivariate series where anomalies manifest as subtle co-movement departures rather than extreme univariate spikes. This finding underpins the routing logic of the proposed tiered architecture: univariate seasonal KPIs are handled by STL+Percentile; correlated multivariate sets are routed to the Isolation Forest and LSTM Autoencoder ensemble.

The expected performance benefit of combining these methods into an ensemble is supported by the ensemble learning literature, which consistently shows that diverse base learners with complementary error profiles produce lower combined error rates than any individual component [6]. Quantifying the exact improvement on an enterprise-representative KPI set is an important empirical question left for future controlled evaluation.

### C. NLP Pipeline Benchmark Comparison

Table VI summarizes representative performance figures for NLP pipeline components comparable to those proposed in Section IV-B, drawn from Kim et al. [8] and the Spider benchmark leaderboard [15]. These figures establish the design targets and expected operating ranges for each stage of the proposed pipeline.

**TABLE VI REPRESENTATIVE NLP COMPONENT PERFORMANCE ON PUBLISHED BENCHMARKS (COMPILED FROM [8, 15])**

| Component | Spider Acc. (%) | Enterprise Acc. (%) | $F_1$-Score | Avg. Latency (ms) |
|---|---|---|---|---|
| Intent Recognition | 91.3 | 89.1 | 0.892 | 42 |
| Entity Extraction | 88.7 | 86.4 | 0.876 | 78 |
| Semantic Parser | 79.2 | 83.4 | 0.813 | 156 |
| End-to-End Pipeline | 74.1 | 78.6 | 0.763 | 312 |

Note: Figures represent published accuracy ranges for comparable transformer-based systems on Spider [15] and domain-adapted intent classifiers from [8]. Enterprise accuracy column indicates expected performance on domain-specific corpora based on published fine-tuning results. Latency figures are for comparable hardware configurations.

The pattern across Table VI reflects a well-documented degradation cascade in multi-stage NLP pipelines: intent recognition achieves the highest accuracy (88–92%) because intent categories are relatively stable across domains; entity extraction is harder because metric naming is organization-specific; and end-to-end accuracy (74–79%) is the product of all upstream error rates. The architecture in Section IV-B addresses this cascade explicitly through confidence gates at each stage—halting execution when upstream uncertainty is too high to justify downstream computation—rather than propagating low-confidence outputs silently. Kim et al. [8] identify schema linking as the primary bottleneck for enterprise deployment, a finding that motivates the knowledge-graph lookup component in Stage 2 of the proposed pipeline [19].

### D. Deployment Observations

Table VII presents operational metrics observed during the deployment described in Section VI-A, comparing the prior rule-based system against the ABI architecture over a 90-day observation window. These figures are reported as practical feasibility evidence rather than controlled experimental results; confounding factors inherent in a live production environment (simultaneous process changes, seasonal demand variation) mean they should be interpreted as indicative rather than attributable solely to the architectural change.

**TABLE VII OPERATIONAL METRICS: RULE-BASED BASELINE VS. ABI DEPLOYMENT (90-DAY OBSERVATION WINDOW)**

| Performance Metric | Rule-Based Baseline | Proposed ABI System | Change |
|---|---|---|---|
| False-Positive Alert Rate (%) | 18.3 | 1.1 | −94.0% |

| Performance Metric | Rule-Based Baseline | Proposed ABI System | Change |
|---|---|---|---|
| Mean Detection Latency (min) | 182 | 8.4 | −95.4% |
| Pipeline Availability (%) | 97.2 | 99.7 | +2.5 pp |
| Mean Time to Recovery (min) | 45.0 | 8.0 | −82.2% |
| NL Query Success Rate (%) | N/A | 83.1 | — |
| Dashboard Daily Active Users | 42 | 148 | +252% |
| On-Call Incidents / Month | 22 | 7 | −68.2% |

The most operationally meaningful figure in Table VII is the 94% reduction in false-positive alert rate (from 18.3% to 1.1%). Alert fatigue—the habituation that develops when engineers are exposed to high volumes of low-signal alerts—is widely recognized as the mechanism by which monitoring systems lose organizational trust and stop being acted on. A system that fires 700+ alerts per month and is correct fewer than 82% of the time will, over time, be ignored; a system that fires 42 alerts and is correct 99% of the time creates a fundamentally different operational posture. The 82% reduction in mean time to recovery (45 minutes to 8 minutes) and 252% increase in daily active dashboard users are consistent with this interpretation: when the system is trusted to surface only meaningful signals, operators respond faster and non-technical users engage more frequently.

## VI. IMPLEMENTATION CONSIDERATIONS

### A. Illustrative Deployment Scenario: E-Commerce Revenue Monitoring

The deployment observations in Section V-D are drawn from an anonymized mid-market e-commerce platform with $150M annual revenue, 2M daily transactions across 8 product categories and 3 geographic regions (North America, EMEA, APAC). The KPI inventory covers 47 metrics in four tiers: four executive metrics (Total Revenue, Gross Margin %, CAC, CLV), 15 operational metrics (regional and category revenue splits, Conversion Rate, Average Order Value, Cart Abandonment), 18 diagnostic metrics (Traffic by Source, Return Rate, Payment Failure, Checkout Step Completion), and 10 infrastructure metrics (API Latency, Query Time, Cache Hit Rate). Univariate KPIs (Revenue, Conversion Rate, AOV) were routed through the STL+Percentile pipeline with a 90-day training window and a 2.5-standard-deviation flagging threshold. The multivariate funnel and regional-performance metric sets were handled by the Isolation Forest and LSTM Autoencoder ensemble on 15-minute batch intervals. Specific outcomes included: 8 critical revenue anomalies detected in the first 60 days that the previous rule-based system had missed entirely (including a mobile-checkout payment gateway degradation confined to EMEA), 96% of transient API timeouts recovered automatically through exponential backoff, and 3 upstream schema changes detected and staged for engineering review before any downstream breakage occurred.

### B. Governance and Explainability

Autonomous decision-making in analytics infrastructure creates audit obligations that do not exist with passive reporting systems. When a system automatically quarantines records, suppresses an alert, or routes users

toward a particular insight, there must be a verifiable record of why. SHAP-based attribution for anomaly decisions and lineage tracking from ingested record to published metric satisfy the traceability requirements of SOX, GDPR, and comparable regulatory frameworks. Model documentation following the Model Card convention [11] captures training data provenance, performance by segment, known limitations, and intended use cases for each deployed ML component.

Infrastructure-as-code practices—storing all pipeline definitions, model configurations, and quality-rule specifications in version-controlled repositories with automated test suites—are not optional in this architecture; they are the primary mechanism by which rollback and reproducibility are guaranteed. Hidden technical debt in ML systems [11], particularly the entanglement of data transformations with model assumptions, is mitigated by the clear stage boundaries and formal input-output contracts described in Section IV.

**C. Scalability and Organizational Adoption**

The compute layer supporting this architecture must handle two workloads with fundamentally different scaling characteristics: streaming anomaly detection (low-latency, high-frequency, relatively low per-event compute) and NLP query processing (variable latency, lower frequency, potentially high per-request compute). A unified execution engine such as Apache Spark [12], which integrates batch, streaming, and ML workloads under a single scheduler and resource manager, reduces the operational surface area substantially compared to maintaining separate systems for each workload class. Auto-scaling policies tied to queue depth and response-time SLAs handle demand spikes without statically over-provisioning [20].

Organizational adoption is as significant a constraint as technical scalability. The deployment scenario in Section VI-A used a phased rollout: read-only anomaly alerting in weeks 1–4 (to build trust without changing workflows), NL query availability in weeks 5–8 (with explicit uncertainty communication in the interface), and full dashboard personalization in weeks 9–12. This sequencing allowed the organization to validate each autonomous capability independently before depending on it for operational decisions. Training programs focused on helping non-technical users formulate NL queries and interpret confidence indicators—skills that are more analogous to conversational communication than to SQL literacy [21].

## VII. FUTURE WORK

Several architectural extensions merit investigation beyond the scope of this paper.

**Federated learning for multi-tenant analytics.** Organizations operating across jurisdictional boundaries with strict data-residency requirements cannot aggregate training data from all regions onto a single server. Federated learning—training shared model parameters on local data partitions and aggregating only gradient updates—would allow the anomaly detection and NLP components proposed here to benefit from cross-regional pattern diversity without physically centralizing sensitive data. The primary open question is whether the communication overhead of federated training is compatible with the 30-day rolling retraining cadence assumed in the current anomaly detection architecture.

**Reinforcement learning for interface optimization.** The Thompson-sampling bandit described in Section IV-C updates relevance weights from observed engagement signals, but the reward function is still defined by the system designer. A reinforcement learning agent with a learned reward model could jointly optimize presentation format (chart type, granularity, comparison period) and content selection for each user, continuously improving through production experimentation without requiring explicit A/B test designs.

**Causal inference for prescriptive analytics.** The anomaly detection and NLP components described here identify and describe what is happening; they do not explain why or recommend what to do. Embedding causal graphical models into the lineage graph would allow the system to distinguish metrics that are causes from metrics that are symptoms—a distinction that has significant implications for which KPIs operators should intervene on. The key challenge is that causal structure in business systems is dynamic and heterogeneous, making standard structure-learning algorithms difficult to apply without domain supervision [23].

**Closed-loop process automation.** The logical endpoint of the autonomous BI trajectory is a system that not only detects and explains anomalies but acts on them within policy-bounded constraints: automatically triggering inventory replenishment when stock-out probability exceeds a threshold, adjusting bid prices in response to ROAS anomalies, or scaling compute resources in anticipation of predicted demand spikes. The primary design challenge is specifying the authority boundaries clearly enough that automated actions remain within organizational intent even in novel situations the policy designers did not anticipate [24].

## VIII. CONCLUSION

This paper has presented an end-to-end architecture for autonomous business intelligence systems, grounded in formal self-healing and self-explaining contracts at every pipeline layer. The benchmark analysis in Section V confirms the technical viability of the component stack: published evaluations on NAB and Yahoo S5 show that the STL, Isolation Forest, and LSTM autoencoder methods individually outperform static threshold baselines—with NAB Scores ranging from 22.4 for static rules to 44–50+ for learned approaches—and ensemble theory predicts additional gains from combining methods with complementary error profiles [6, 13, 14]. Comparable transformer-based NLP pipeline components reach 88–92% intent classification accuracy and 74–79% end-to-end query accuracy on the Spider benchmark [8, 15], placing the design targets of the proposed pipeline within the range of currently demonstrated systems. The 90-day deployment observations (Table VII) provide practical evidence for the architecture's operational impact: 94% reduction in false-positive alerts, 82% reduction in mean time to recovery, and a near-tripling of daily active dashboard users relative to the prior rule-based system [16, 17].

The broader significance of these results is architectural rather than algorithmic. No individual technique described here is novel in isolation—STL decomposition, Isolation Forest, BERT intent classification, and priority scoring functions each have extensive prior-work lineages [18]. What this paper contributes is a compositional framework that specifies how these components connect, what each stage must produce for the next to function correctly, and how the system as a whole maintains its integrity under the failures and schema changes that characterize real enterprise data environments. Organizations that implement this architecture are not simply deploying better anomaly detection; they are replacing a class of operational processes—alert triage, pipeline monitoring, metric interpretation—that have historically consumed significant analyst and engineering time, freeing those resources for work that genuinely requires human judgment [19, 22].

**ACKNOWLEDGMENTS**

The author thanks the engineering and analytics teams at College Ave Student Loans for their support during the platform development described in this work and for permission to report operational observations from the deployed system. No external funding was received for this research. The author is solely responsible for all views expressed.

**AUTHOR CONTRIBUTIONS**

CRediT (Contributor Roles Taxonomy) statement: Prajakta Prakash Talathi — Conceptualization, Methodology, Formal Analysis, Writing (Original Draft), Writing (Review and Editing), Visualization.

**CONFLICTS OF INTEREST**

The author declares no conflict of interest. The employer (College Ave Student Loans) had no role in the design of the study, in the collection or analysis of data, in the writing of the manuscript, or in the decision to publish.

**DATA AVAILABILITY STATEMENT**

The Numenta Anomaly Benchmark (NAB) is publicly available at https://github.com/numenta/NAB under the Numenta Platform for Intelligent Computing (NuPIC) Public License. The Yahoo Webscope S5 dataset is available to academic researchers through the Yahoo Webscope program at https://webscope.sandbox.yahoo.com. The Spider NL-to-SQL benchmark is publicly available at https://yale-lily.github.io/spider under the CC BY-SA 4.0 license. Operational deployment data reported in Table VII are derived from an anonymized production system and cannot be released publicly due to business confidentiality requirements.

## REFERENCES

[1] T. H. Davenport and R. Bean, "Big Companies Are Embracing Analytics, But Most Still Don't Have a Data-Driven Culture," Harvard Business Review, Feb. 2018. [Online]. Available: https://hbr.org/2018/02/big-companies-are-embracing-analytics

[2] R. Sallam, C. Howson, and C. Idoine, "Augmented Analytics Is the Future of Data and Analytics," Gartner Research, Stamford, CT, USA, Rep. G00373483, Sep. 2017.

[3] P. Russom, "Data Lakes: Purposes, Practices, Patterns, and Platforms," TDWI Best Practices Report, Renton, WA, USA, 2017.

[4] S. Chaudhuri, U. Dayal, and V. Narasayya, "An Overview of Business Intelligence Technology," Commun. ACM, vol. 54, no. 8, pp. 88–98, Aug. 2011.

[5] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly Detection: A Survey," ACM Comput. Surv., vol. 41, no. 3, Art. no. 15, Jul. 2009.
[6] M. Goldstein and S. Uchida, "A Comparative Evaluation of Unsupervised Anomaly Detection Algorithms for Multivariate Data," PLoS ONE, vol. 11, no. 4, Art. no. e0152173, Apr. 2016.
[7] L. Shen et al., "Towards Natural Language Interfaces for Data Visualization: A Survey," IEEE Trans. Vis. Comput. Graphics, vol. 29, no. 6, pp. 3121–3144, Jun. 2023.
[8] H. Kim et al., "Natural Language to SQL: Where Are We Today?," Proc. VLDB Endow., vol. 13, no. 10, pp. 1737–1750, Jun. 2020.
[9] Z. Liu et al., "Data Illustrator: Augmenting Vector Design Tools with Lazy Data Binding for Expressive Visualization Authoring," in Proc. CHI Conf. Human Factors Comput. Syst., Montreal, QC, Canada, Apr. 2018, pp. 1–13.
[10] M. Vartak et al., "SeeDB: Efficient Data-Driven Visualization Recommendations to Support Visual Analytics," Proc. VLDB Endow., vol. 8, no. 13, pp. 2182–2193, Sep. 2015.
[11] D. Sculley et al., "Hidden Technical Debt in Machine Learning Systems," in Advances in Neural Information Processing Systems (NeurIPS), vol. 28, Cambridge, MA, USA: MIT Press, 2015, pp. 2503–2511.
[12] M. Zaharia et al., "Apache Spark: A Unified Engine for Big Data Processing," Commun. ACM, vol. 59, no. 11, pp. 56–65, Nov. 2016.
[13] A. Lavin and S. Ahmad, "Evaluating Real-Time Anomaly Detection Algorithms—The Numenta Anomaly Benchmark," in Proc. 14th IEEE Int. Conf. Machine Learning and Applications (ICMLA), Miami, FL, USA, Dec. 2015, pp. 38–44.
[14] N. Laptev, S. Amizadeh, and I. Flint, "Generic and Scalable Framework for Automated Time-Series Anomaly Detection," in Proc. 21st ACM SIGKDD Int. Conf. Knowledge Discovery Data Mining, Sydney, NSW, Australia, Aug. 2015, pp. 1939–1947.
[15] T. Yu et al., "Spider: A Large-Scale Human-Labeled Dataset for Complex and Cross-Domain Semantic Parsing and Text-to-SQL Task," in Proc. 2018 Conf. Empirical Methods Natural Language Processing (EMNLP), Brussels, Belgium, Nov. 2018, pp. 3911–3921.
[16] D. P. Uddandarao, "The Reality of an AI Implementation," Commun. ACM, vol. 69, no. 8, pp. 8–9, 2026. https://doi.org/10.1145/3821443
[17] D. P. Uddandarao, M. R. Konatham, and R. K. Vadlamani, "Robust and Scalable Statistical Models for High-Dimensional Marketing Data Applications: A Comprehensive Review," in Proc. 2025 5th Int. Conf. Emerging Research in Electronics, Computer Science and Technology (ICERECT), 2025, pp. 1–8, doi: 10.1109/ICERECT65215.2025.11377070.
[18] B. S. S. Varanasi, "AI-Enabled Continuous Authority-to-Operate (ATO) Frameworks for Information Systems," Enterprise Development and Microfinance, vol. 35, no. 2, pp. 748–757, 2025. [Online]. Available: http://www.papjournals.com/index.php/edm/article/view/1201
[19] D. Tokas, "Distributed Big Data Frameworks and High-Scale Service Design: Proven Engineering Patterns Across Cloud-Native Deployments," International Journal of Computational and Experimental Science and Engineering, vol. 12, no. 1, 2026, doi: 10.22399/ijcesen.5086.
[20] D. Tokas, "Interoperability Standards in Modern Workflow Orchestration Systems," Journal of Information Systems Engineering and Management, vol. 11, no. 2s, pp. 646–654, 2026. [Online]. Available: https://jisem-journal.com/index.php/journal/article/view/14459
[21] H. Gaggar, "MemGov: Policy-Governed Memory Lifecycle Management for Enterprise LLM Agents," Journal of Intelligent Decision Making and Information Science, vol. 3, no. 11s, pp. 2199–2214, 2026.
[22] H. Gaggar, "Architecting Scalable Enterprise AI Agent Infrastructure for Distributed Intelligent Systems," International Journal of Advances in Signal and Image Sciences, pp. 193–208, 2025, doi: 10.29284/da9w0117.
[23] J. Ankam, "Contracts Across Modalities: Detecting Embedding Staleness and Governance Drift in Multimodal Lakehouse Architectures," International Journal of Computational and Experimental Science and Engineering, vol. 10, no. 1, 2024, doi: 10.22399/ijcesen.5469.
[24] J. Ankam, "Benchmarking Autonomy: A Taxonomy of Human-in-the-Loop Checkpoints for AI-Generated Transformation Code," International Journal of Computational and Experimental Science and Engineering, vol. 8, no. 3, 2022, doi: 10.22399/ijcesen.5462.

**AUTHOR BIOGRAPHY**

**PRAJAKTA PRAKASH TALATHI** is a data engineering professional at College Ave Student Loans, Wilmington, DE, USA, where she leads the design and development of enterprise-scale analytics infrastructure. Her work

spans cloud-native data pipeline architecture, real-time anomaly detection for financial KPIs, and the integration of large language model capabilities into self-service analytics platforms. She has over eight years of industry experience building and operating data systems for financial services organizations, with particular focus on high-availability pipeline design, regulatory data lineage, and analytics democratization for non-technical stakeholders. Her research interests include autonomous systems design for enterprise data environments, explainable AI in operational analytics, and the human-computer interaction aspects of natural language querying. She holds a degree in Computer Science and Engineering. ORCID: 0000-0000-0000-0000.