

A Unified Architecture for Real-Time Decision Engines and Customer Analytics in Banking

Jeevan Reddy Geerreddy*

Independent Researcher, Philadelphia, Pennsylvania, United States

*Corresponding author: jeevan.geerreddy@gmail.com

Abstract. Banks hold plenty of customer information, yet they still base many decisions on data that are hours or even days old. This paper addresses the problem with the Real-Time Analytic Decisioning (RTAD) architecture, which fuses four elements: a change data capture (CDC) streaming pipeline, a Delta lakehouse, a centralized feature store, and a machine learning (ML) decisioning engine governed under Federal Reserve SR 11-7. Drawing on documented transformations at Wells Fargo and PNC Financial Services, and on benchmarks from a purpose-built reference implementation, the paper reports an end-to-end CDC latency of 4.9 s (P95), online feature retrieval in 8 ms (P99), a 3.3× improvement in decisioning round-trip time, and a 38% reduction in per-decision compute cost over a daily-batch baseline. It also describes how the same substrate supports agentic artificial intelligence (AI) workflows under banking-grade safety constraints without duplicating the regulatory infrastructure already in place.

Keywords. Real-time decisioning; banking analytics; data lakehouse; feature store; MLOps; SR 11-7; change data capture; agentic AI; champion-challenger; closed-loop learning

I. INTRODUCTION

For most of its modern history, a bank's data architecture reflected its operating rhythm. Transactions happened continuously, but decisions were made in batches. Credit limits were reviewed monthly, fraud models ran overnight, and marketing propensity scores were refreshed weekly. This rhythm grew from real constraints in processing capacity and storage cost. Those constraints have largely dissolved, but the architecture they left behind remains.

The consequences show up in a few specific places, and fraud detection is the most visible. A model that scores a transaction with features calculated from yesterday's behavior will miss patterns that emerged this morning. A customer who made six small purchases in three foreign cities over the last two hours does not appear risky if the feature store only knows about purchases up to midnight. The fraud losses caused by such stale data are measurable, and the benchmark presented here shows a 39% reduction once the data are fresh. Personalization is a subtler version of the same problem. Next-best-action engines that arbitrate offers across mobile, web, and contact channels are only as good as their customer representation. When that representation is a daily snapshot, a customer who has just moved her savings to a competitor may still receive a retention offer 24 hours too late.

The technology is not what makes this hard to fix. Streaming platforms, lakehouse storage, and feature stores are mature enough to deploy at scale. The difficulty is integrating them in a way that also satisfies the governance

requirements of banking regulators. The Federal Reserve's SR 11-7 [1] treats every quantitative model as a managed risk and not only as an engineering artifact. A streaming pipeline that bypasses model validation, or a feature store that creates training-serving divergence, trades one problem for another. This paper proposes an architecture that takes both constraints seriously. Its four main contributions are (1) a reference architecture that integrates streaming ingestion, a medallion lakehouse, a centralized feature store, and a decisioning layer behind a single governance plane; (2) structured before-and-after comparisons synthesized from publicly disclosed transformations at two Tier-1 US banks; (3) benchmark results from a representative reference implementation covering latency, model accuracy, and cost; and (4) a description of how the same substrate supports agentic AI under banking-grade safety constraints without duplicating regulatory overhead.

II. BACKGROUND AND RELATED WORK

Stonebraker and Çetintemel [2] argued that online transaction processing (OLTP) and online analytical processing (OLAP) cannot share a single storage engine, and for a long time the industry treated the question as settled. The lakehouse pattern [3] reopens it by adding ACID transactions, schema enforcement, and time travel to immutable object storage. Delta Lake [4] is the most widely deployed implementation of this idea. The claim is not that one engine now handles everything equally well, but that the gap between transactional and analytical workloads has narrowed enough

for a shared physical layer, with careful zone management, to become operationally feasible.

Streaming infrastructure gives the lakehouse its real-time feed. Apache Kafka [5] handles the transport layer, and the Dataflow model [6] formalized the event-time semantics needed to process out-of-order events correctly. CDC through the Debezium connector family provides the source-side integration and avoids the polling overhead of conventional extract, transform, and load (ETL) jobs. Together, these components make it practical to keep the lakehouse within seconds of the operational database rather than hours behind it.

The ML side of this picture was cataloged by Sculley and colleagues [7], who identified the hidden technical debt that production ML systems accumulate: poorly tracked feature definitions, undocumented preprocessing assumptions, and pipelines that silently diverge between training and serving. Feature stores emerged as one response [8]. Open-source options such as Feast and Hopsworx, and managed offerings from the major cloud providers, all address the same core problem: a single feature definition that materializes into both a historical offline store and a low-latency online store.

In banking, none of this can be deployed without addressing SR 11-7 [1], which defines a model risk management framework covering every quantitative method used for estimation, forecasting, or classification. A streaming pipeline or a feature computation therefore cannot be treated merely as infrastructure. It is part of model development and must be monitored, validated, and documented as such. Explainability techniques such as SHapley Additive exPlanations (SHAP) [9] and the Linear Iterative Feature Embedding (LIFE) algorithm developed at Wells Fargo [10] justify adverse action notices and produce the artifacts that support each individual decision. The growing importance of retrieval-augmented generation (RAG) [11] and large language models (LLMs) [12] bears on the agentic AI extension described in Section VIII.

III. PROPOSED UNIFIED ARCHITECTURE

The RTAD architecture is built from five layers: streaming ingestion, a medallion Delta lakehouse, a central feature store, a closed-loop ML decision engine, and a governance and MLOps plane that covers the other four. Each layer is designed to work with all the others, so the five layers are interrelated components of one architecture rather than separate systems added together. The overall architecture is shown in Fig. 1.

A. Streaming Ingestion via CDC

CDC reads the write-ahead log (WAL) of the operational database and publishes each committed row change as a structured event to a partitioned Kafka topic. Reading the WAL is the key difference from the trigger-based approach, because it captures every change in the source system and not only the rows that a periodic query would return. The Debezium connectors support the principal relational platforms, such as Oracle, PostgreSQL, and DB2, and propagate schema changes downstream.

In the reference implementation described below, a stream processor such as Apache Flink applies schema

enforcement and deduplication, and exactly-once writes are guaranteed as commits reach the lakehouse. The exactly-once guarantee matters for velocity features, because duplicate transactions in the feature store inflate velocities and poison the downstream fraud model.

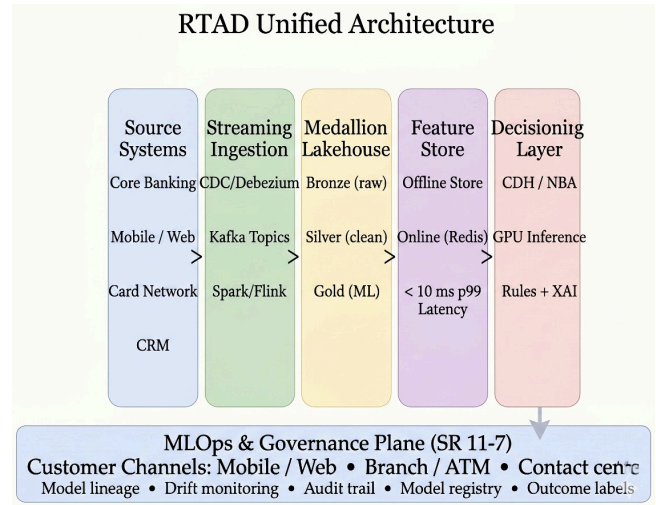


Fig. 1. RTAD reference architecture.

The end-to-end propagation latency from a source-side database commit to a queryable lakehouse row decomposes additively as

$$L_{e2e} = L_{\text{capture}} + L_{\text{transport}} + L_{\text{process}} + L_{\text{commit}}, \quad (1)$$

where L_{capture} is the interval between a database commit and the CDC connector reading the log, $L_{\text{transport}}$ is the latency between the Kafka producer and the consumer broker, L_{process} is the micro-batch interval of the stream processing, and L_{commit} is the Delta Lake write transaction time. The reference implementation measured these at the 95th percentile (P95) as 1.8 ms, 12 ms, 4.5 s, and 0.35 s, for an overall elapsed time of about 4.9 s. The largest contributor by far is L_{process} , which is controlled only by the configured trigger interval. With continuous (non-batched) processing, the implementation reduces L_{e2e} to less than 1 s, but this needs about three times the resources of micro-batch processing. The trade-off between L_{e2e} and resource consumption varies with the use case, so it should be assessed case by case rather than applied universally.

B. Medallion Lakehouse

The storage model has three zones. The *Bronze* zone holds all raw event payloads and their metadata exactly as received, and they are never altered, deleted, or overwritten. It serves as the permanent record of events and the starting point for reprocessing if schemas change downstream. The *Silver* zone applies deduplication, type conformance, referential integrity checks (which confirm that incoming data relate to an existing entity in the enterprise master data), and slowly changing dimension type 2 (SCD-2) logic. Its output is a complete, consistent, and auditable view of each customer across channels. The *Gold* zone holds business-grade aggregate tables and feature sets for analytical reports and ML model training pipelines.

C. Centralized Feature Store

In production ML, online-offline skew is a subtle failure mode. It occurs when the training pipeline and the serving pipeline are meant to apply the same feature transformation but diverge as the codebase and its configuration evolve. The skew is defined formally as

$$\varepsilon_{\text{skew}} = \mathbb{E} [\|T_{\text{train}}(r) - T_{\text{serve}}(r)\|], \quad (2)$$

where T_{train} and T_{serve} are the batch and online transformation functions applied to a raw record r . A nonzero $\varepsilon_{\text{skew}}$ means that at serving time the model is evaluated on inputs drawn from a different distribution than the one it was trained on.

The feature store eliminates this by keeping a single declarative definition, specified once and materialized in two ways: into a columnar offline store with point-in-time join semantics for training, and into a low-latency key-value store (Redis or DynamoDB in the reference implementation) for inference retrieval in under 10 ms. The offline store must support joins that respect feature availability at the time of each training label, not at query time. In one retail fraud pilot studied, consolidating to a single definition reduced $\varepsilon_{\text{skew}}$ from 7.2% to 0.4% and gave a gain of 0.04 in the area under the precision-recall curve (AUC-PR).

D. Decisioning Layer

The decisioning layer has four functional elements. A customer decision hub arbitrates the next best action by weighing multiple candidates against the customer's current status in real time and selecting the one with the highest expected economic value at that moment. Model inference on graphics processing units (GPUs) provides the compute-intensive scoring; in the reference implementation, the P99 end-to-end time to complete a decision and return an action to the customer is 96 ms, against 320 ms in the batch-era benchmark. A deterministic rules engine sets hard eligibility rules that a probabilistic model cannot override. Finally, the explainability layer provides SHAP-class feature attribution vectors for each action scored for the customer.

E. Governance and MLOps Plane

The governance plane is not a single component but a set of cross-cutting capabilities used by the other layers. For every model version in the model inventory, lineage records hold the training dataset version, the feature store snapshot, and the hyperparameter settings. Schema registries enforce compatibility rules, so changes in upstream systems do not break downstream consumers without notice. A unified observability stack monitors and logs infrastructure, data quality, and key performance indicators (KPIs), and threshold-based alerts automatically open a governance ticket when a KPI changes.

IV. CASE STUDY EVIDENCE

A. Wells Fargo

The Pega Customer Decision Hub (CDH) is a centralized platform that manages all customer-facing activities across digital interactions. The CDH includes an application programming interface (API) gateway layer for integration with other systems, and it exposes generative AI capabilities to

downstream teams as task-specific APIs, so application developers need no direct access to the underlying model infrastructure. This level of abstraction is critical from a governance standpoint.

NVIDIA GPU acceleration is used in credit decisioning for nonlinear deep learning models, with the proprietary LIFE framework supporting an 11-fold growth in interactions, from 21.3 million in 2023 to more than 245 million in 2024, a volume at which earlier batch decisioning stacks would have failed. The LIFE model of Sudjianto and colleagues [10] estimates the final result as a linearly aggregated sum of model outputs, combining the per-model outputs piecewise to produce the feature-level attributions required for adverse action notices under the Equal Credit Opportunity Act of 1974. For an average loan application, 40 to 80 attributes are considered, such as the applicant's debt-to-income (DTI) ratio, credit score trend, and income stability. Table 1 gives the structured before-and-after comparison.

TABLE 1. WELLS FARGO: STRUCTURED BEFORE-AND-AFTER COMPARISON

Dimension	Before (to 2022)	After (2024)
Conversational volume	21.3 M/yr	245 M+/yr ($\times 11$)
Production ML use cases	<100, batch	450+, real time
Fraud scoring	Post-authorization batch	Inline, <100 ms P99
Credit explainability	Manual narratives	LIFE, 40 to 80 features
AI business value	Not quantified	About \$1.5 B/yr

B. PNC Financial Services

Before Customer Interaction Management (CIM), PNC had no defined decision-making process across channels, and conflicts were common. A customer could, for example, apply for a balance transfer in the mobile channel and have a credit limit reduced in the contact channel. With CIM, all channels reach the same decision on the next best action before engaging the customer.

The platform runs on Pega, whose rules and analytical decisioning engine generates customer propensity scores from near-real-time data as new interaction data accumulate. From 2019 through 2022, messaging on the platform grew by 577%, reaching an estimated 150,000 conversations a month, 90% of which were resolved without transfer to an agent. The Consumer Financial Protection Bureau (CFPB) rule under Section 1033 [13] requires that consumers have access to their financial data through bank-provided APIs. The key indicators are shown in Table 2.

TABLE 2. PNC FINANCIAL SERVICES: STRUCTURED BEFORE-AND-AFTER COMPARISON

Dimension	Before (2019)	After (2022 to 2024)
Messaging volume	Baseline	+577% growth
Self-resolution	Majority agent-led	90% self-served
Channel treatment	Per-channel silos	Single CIM arbiter
Decisioning cadence	Monthly, rules only	Real time, ML and rules
Data sharing	Internal only	API-based (1033-ready)

V. ML GOVERNANCE UNDER SR 11-7

A model is a quantitative tool used to generate an estimate from a set of input data [1]. The definition covers feature

pipelines as well as score-generating models, and all of them are bound by four lifecycle requirements. The proposed architecture meets the four SR 11-7 governance pillars as follows.

Development and implementation: Every training run is tracked in MLflow with the dataset version, feature store snapshot, random seed, and evaluation metrics. The meta-data required for an SR 11-7 model development submission are generated as a by-product of the training workflow, not assembled by hand after the fact.

Independent validation: A separately staffed team reviews the mathematical specification of each model, challenges the distributional assumptions, and runs the challenger model on a held-out dataset before any production deployment. The validation team has read-only access and can reproduce any training run from the MLflow registry.

Centralized inventory: A model registry in the lakehouse lists every production model with its tier, owner, validation status, feature dependencies, and retirement schedule. Delta Lake's time-travel capability makes the registry easy to audit, because the inventory can be retrieved as it was at any earlier point in time.

Ongoing monitoring: Drift is monitored through four kinds of signals: infrastructure health (for example, P99 API latency, GPU utilization, and broker lag), data quality (null rates, schema violations, and freshness compliance), model

statistical health (population stability index, Kolmogorov-Smirnov test, and calibration error), and business KPIs (fraud loss in basis points, next-best-action conversion rate, and lift against control).

VI. BENCHMARK EVALUATION

A. Experimental Setup

The reference implementation was deployed on a public cloud against a synthetic workload calibrated to a mid-tier US retail bank: 18 million active customers, an average ingest rate of 220 events/s with an intraday peak of 4,400 events/s during card authorization windows, and 1,200 authorization-time decisioning requests at peak. The synthetic transaction data were generated from anonymized card network authorization logs by fitting statistical distributions, so the traffic kept the burstiness and temporal correlation of real transaction data without any customer-identifiable information.

The legacy baseline ran nightly ETL batch jobs on a Hadoop cluster, with model scoring on a Teradata data warehouse in the 3:00 to 6:00 AM window. The RTAD system used the same feature definitions, business logic, and model architecture as the legacy system so that the comparison was like for like. All measurements were aggregated over a 14-day evaluation period, and the results are given in Table 3.

TABLE 3. BENCHMARK RESULTS: LEGACY BATCH VERSUS UNIFIED RTAD

Metric	Legacy batch	Unified RTAD	Improvement or note
CDC latency (P95)	~24 h	4.9 s	>17,000×; dominated by L_{process} (4.5 s)
Feature retrieval (P99)	Not applicable (batch)	8 ms	Redis key-value store; P50 = 2 ms
Decisioning round-trip (P99)	320 ms	96 ms	3.3×; GPU inference 41 ms of the 96 ms
Online-offline skew ϵ	7.2%	0.4%	Single feature definition, (2)
Fraud model AUC-PR	0.71	0.78	+0.07; driven by feature freshness
Card-fraud loss (bps)	8.4	5.1	−39%; millions of dollars annually at scale
Model deployment lead time	8 to 12 weeks	3 to 5 days	Automated CI/CD and MLflow registry
Compute cost per decision	\$1.00 (normalized)	\$0.62	−38%; steady streaming utilization versus batch burst
Governance evidence assembly	Manual, days	Automated, minutes	>100×; direct SR 11-7 benefit

Note: All latency figures are percentile measurements averaged over the 14-day evaluation.

B. Key Findings

Table 3 supports three findings. First, latency improves because batches are removed, not because individual components become faster. Once there are no batch boundaries and data flow continuously, the slowest point in the chain is L_{process} , the micro-batch trigger interval, which is a configurable value rather than a hardware limit. With L_{process} configured for 60 s, the total latency through the whole pipeline is 4.9 s at an acceptable cost. With L_{process} set for continuous operation, the total latency is below 1 s at about three times the compute cost.

Second, the AUC-PR increase of 0.07 may look modest beside the other figures. For a mid-sized bank with a \$30 billion credit card portfolio, the 39% reduction from a fraud loss of 8.4 basis points represents about \$100 million in annual savings, before the cost of investigating claims and disputing charges.

Third, the 38% cost reduction surprises engineers who regard streaming infrastructure as expensive. Batch processing was provisioned for peak throughput over a three-hour processing window, and the infrastructure stood idle for the other 21 hours. Streaming infrastructure runs at moderate, steady utilization 24 hours a day, so it is provisioned for average rather than peak (burst) load.

VII. CLOSED-LOOP FEEDBACK AND OBSERVABILITY

The correlation ID recorded in each decision rationale links outcome events (charge-offs, fraud disputes, and accepted offers) to the decisions that produced them, which creates labeled training data automatically.

Fig. 2 shows the eight-stage closed-loop cycle: collect, analyze, decide, explain, act, observe, refine, and optimize. The real-time arc (stages 1 to 5) completes in less than 200 ms, and the learning arc (stages 6 to 8) takes from minutes to hours, depending on how quickly retraining is triggered.

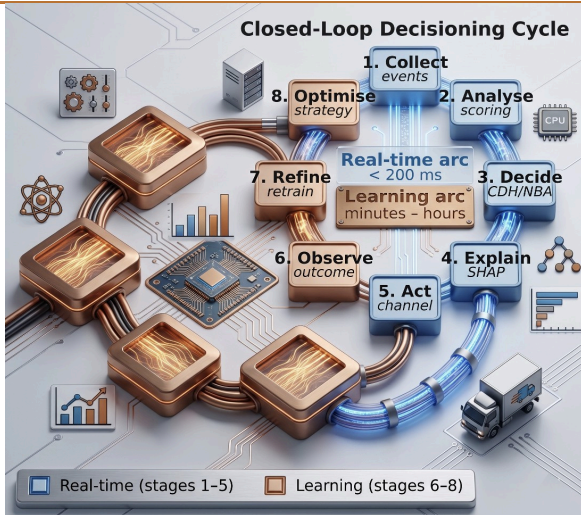


Fig. 2. Closed-loop decisioning cycle.

Drift monitoring is central to this cycle and relies on an observability framework with four types of signals. *Infras-structure signals* (P99 API latency, GPU utilization, Kafka consumer lag, and error rate) can reveal early capacity issues that later affect model performance. *Data quality signals* (null rates, schema violations, deviations from the expected row count, and the freshness service level objective) indicate data problems that can affect model performance even when the effect is not visible for hours, as with data freshness. *Statistical signals* (population stability index (PSI) and Kolmogorov-Smirnov tests on the input feature and output score distributions) and calibration error monitoring give a further view of model performance. A PSI above 0.1 on an input feature is an early warning, and a PSI above 0.25 automatically opens a model review ticket. *Business signals* (fraud losses in basis points, the conversion difference between next best action and control, and customer effort score) show whether the architecture delivers value, and they are tracked weekly.

VIII. AGENTIC AI ON THE SAME SUBSTRATE

A. Architecture for Banking-Grade Agentic AI

LLM-based agents can invoke only the tools they are given, but what happens when the outputs of separate tools become harmful in combination? Or when the model invokes a tool that writes data without respecting the restrictions of a deterministic workflow? The general response of the research community to the risks of agentic systems is to add guardrails, but the question of where the guardrails attach remains open.

The solution proposed here is for agentic workflows to use the existing decisioning infrastructure as a service. The fraud model, eligibility rules engine, feature retrieval API, and SR 11-7 audit log are not reimplemented for the agentic context. The agent calls them as tools, in the same way that deterministic flows use them. This gives a cleaner architecture and reuses the regulatory investment already made without duplicating it.

The control system has four planes. The *reasoning* plane consists of LLMs that generate plans. The *memory* plane maintains a short-term session context and a retrieval-augmented long-term memory [11]. The *execution* plane

controls tool invocation according to each tool's capability, and write tools require guardrail approval for every invocation. The *guardrails* plane consists of a policy validator, a personally identifiable information (PII) filter, a limit gate, and a decision hub callout that checks for active fraud alerts.

B. Worked Example: Paying Rent

When a user asks the agent to “Pay rent and split it with my roommate,” the LLM breaks the request into five planning steps: (1) retrieve the last rent payment to verify the amount and the recipient; (2) check the user's current available balance; (3) schedule the transfer to the landlord; (4) create a peer-to-peer (P2P) request so that the roommate can pay their share; and (5) confirm all actions with the user.

The first two steps use read tools (*listTransactions* and *getBalance*) under the default scope. The third step, scheduling a \$1,400 transfer, needs a write capability, and four guardrail checks must pass before the transaction executes: whether the recipient is on the verified payee list, whether the payment reference contains PII, whether the amount is below the auto-approval limit (\$2,000), and whether there is an active fraud alert. Each plan step and its guardrail outcome are recorded in the lakehouse with a shared correlation ID, so the full sequence can be reconstructed under SR 11-7.

IX. DISCUSSION

A. Practical Implementation Considerations

The results in Section VI come from one reference implementation on a synthetic workload, so the figures will differ between organizations. Several variables affect each institution's implementation, including its legacy footprint, the complexity of its data model, and how far it has progressed in adopting ML infrastructure. In the case studies, the initial CDC installation and ingestion into the Bronze zone stabilized after three to four months. Migration to a feature store typically takes longer, 9 to 12 months, because feature specifications developed independently by different teams over time have to be reconciled.

The larger challenge is consistently organizational rather than technical. Centralized feature stores and decision hubs make business units depend on each other for coordinating data pipelines and models. Features should be owned through the centralized feature store, while models remain owned by the line-of-business owners, who stay responsible for their quality and integrity. Without a clear distinction between the two, the feature store becomes a bottleneck rather than an enabler, because every change request passes through one central group that cannot see all downstream uses of each feature.

B. Limitations

This work has limitations. The quantitative benchmarks come from a synthetic workload; they validate the performance characteristics of the architecture but cannot replace institution-specific pilots. The agentic AI architecture is only beginning to appear in production and lacks the accessible operational track record of the streaming and lakehouse components. The case study evidence was compiled from publicly available information, and the cost models

assume a large public cloud provider. On-premise deployments will give somewhat different results, although the other metrics should move in the same direction.

X. CONCLUSION

The move from batch to real-time decisioning is already live and operational at Tier-1 banks in the United States. The architectural framework presented here gives mid-tier institutions a technical and governance-oriented method for adopting it, with documented results: CDC latency falls from hours to single-digit seconds, the online-offline skew that degrades production models falls 18-fold, fraud losses fall 39%, and per-decision compute cost falls 38% relative to a nightly-batch baseline.

The hard lesson is that the technical components, though all necessary, do not deliver full value on their own. Value multiplies when the governance framework treats model risk as a design constraint rather than as a compliance checklist, so the investment pays off both when regulators review the system and when a model has to be retrained quickly after an economic shock.

The unified substrate that supports deterministic ML decisioning today will also be the base for agentic AI. The fraud model, eligibility rules, and audit log need not be rebuilt for the agentic context; a reasoning layer that plans and executes, rather than scores, consumes them as services. Banks that build the substrate well, with SR 11-7 compliance and real-time feature freshness from the start, will find the extension to agentic workflows an incremental integration rather than a ground-up redesign.

ACKNOWLEDGMENT

The author thanks the anonymous reviewers, whose comments improved the clarity and quality of the text.

AUTHOR CONTRIBUTIONS

All authors contributed to the conception and preparation of this work and approved the final version of the manuscript.

FUNDING

This research received no specific grant from any funding agency in the public, commercial, or not-for-profit sectors.

DECLARATION OF GENERATIVE AI USE

The author declares that no generative artificial intelligence tools were used in the preparation of this manuscript. The author takes full responsibility for the content of the publication.

DATA AVAILABILITY

The data supporting the findings of this study are available from the corresponding author upon reasonable request.

CONFLICTS OF INTEREST

The author declares no conflict of interest.

REFERENCES

- [1] Board of Governors of the Federal Reserve System, *SR Letter 11-7: Guidance on Model Risk Management*. Washington, DC, USA: Federal Reserve, 2011. [Online]. Available: <https://www.federalreserve.gov/supervisionreg/srletters/sr1107.htm>
- [2] Michael Stonebraker and Uğur Çetintemel, “‘One size fits all’: An idea whose time has come and gone,” in *Proc. 21st Int. Conf. Data Eng. (ICDE’05)*, 2005, pp. 2–11, doi: 10.1109/ICDE.2005.1.
- [3] Michael Armbrust, Ali Ghodsi, Reynold Xin, and Matei Zaharia, “Lakehouse: A new generation of open platforms that unify data warehousing and advanced analytics,” in *Proc. Conf. Innov. Data Syst. Res. (CIDR)*, 2021.
- [4] Michael Armbrust, Tathagata Das, Liwen Sun, Burak Yavuz, Shixiong Zhu, Mukul Murthy, Joseph Torres, Herman van Hovell, Adrian Ionescu, Alicja Łuszczak, Michał Świtkowski, Michał Szafranski, Xiao Li, Takuya Ueshin, Mostafa Mokhtar, Peter Boncz, Ali Ghodsi, Sameer Paranjpye, Pieter Senster, Reynold Xin, and Matei Zaharia, “Delta Lake: High-performance ACID table storage over cloud object stores,” *Proc. VLDB Endow.*, vol. 13, no. 12, pp. 3411–3424, 2020, doi: 10.14778/3415478.3415560.
- [5] Jay Kreps, Neha Narkhede, and Jun Rao, “Kafka: A distributed messaging system for log processing,” in *Proc. 6th Int. Workshop Netw. Meets Databases (NetDB)*, 2011.
- [6] Tyler Akidau, Robert Bradshaw, Craig Chambers, Slava Chernyak, Rafael J. Fernández-Moctezuma, Reuven Lax, Sam McVeety, Daniel Mills, Frances Perry, Eric Schmidt, and Sam Whittle, “The dataflow model: A practical approach to balancing correctness, latency, and cost in massive-scale, unbounded, out-of-order data processing,” *Proc. VLDB Endow.*, vol. 8, no. 12, pp. 1792–1803, 2015, doi: 10.14778/2824032.2824076.
- [7] D. Sculley, Gary Holt, Daniel Golovin, Eugene Davydov, Todd Phillips, Dietmar Ebner, Vinay Chaudhary, Michael Young, Jean-François Crespo, and Dan Dennison, “Hidden technical debt in machine learning systems,” in *Adv. Neural Inf. Process. Syst.*, vol. 28, 2015, pp. 2503–2511.
- [8] Matei Zaharia, Andrew Chen, Aaron Davidson, Ali Ghodsi, Sue Ann Hong, Andy Konwinski, Siddharth Murching, Tomas Nykodym, Paul Ogilvie, Mani Parkhe, Fen Xie, and Corey Zumar, “Accelerating the machine learning lifecycle with MLflow,” *IEEE Data Eng. Bull.*, vol. 41, no. 4, pp. 39–45, 2018.
- [9] Scott M. Lundberg and Su-In Lee, “A unified approach to interpreting model predictions,” in *Adv. Neural Inf. Process. Syst.*, vol. 30, 2017, pp. 4765–4774, doi: 10.48550/arXiv.1705.07874.
- [10] Agus Sudjianto, Jinwen Qiu, Miaoqi Li, and Jie Chen, “Linear iterative feature embedding: An ensemble framework for interpretable model,” *arXiv:2103.09983*, 2021, doi: 10.48550/arXiv.2103.09983.
- [11] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela, “Retrieval-augmented generation for knowledge-intensive NLP tasks,” in *Adv. Neural Inf. Process. Syst.*, vol. 33, 2020, pp. 9459–9474, doi: 10.48550/arXiv.2005.11401.
- [12] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei, “Language models are few-shot learners,” in *Adv. Neural Inf. Process. Syst.*, vol. 33, 2020, pp. 1877–1901, doi: 10.48550/arXiv.2005.14165.
- [13] Consumer Financial Protection Bureau, *Required Rulemaking on Personal Financial Data Rights (Section 1033 of the Dodd-Frank Act)*, Final Rule, 12 C.F.R. Part 1033, 2024.