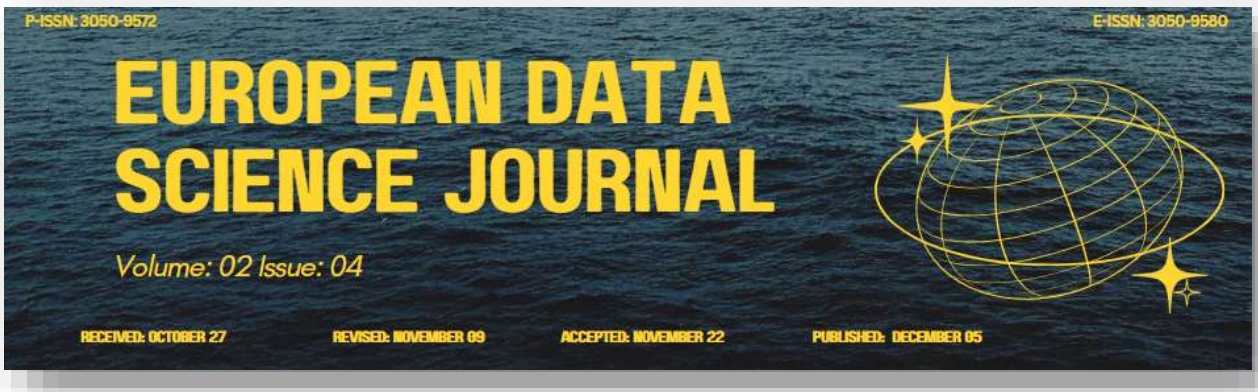


Medallion-Based Feature Engineering for Real-Time MLOps

Daniel Thompson
Asst Professor

Abstract

As large-scale machine learning in production becomes ubiquitous across various industries, both engineering and research-oriented teams are pushing the limits for faster and continuously retrained models. Nevertheless, building and running a model, even using a well-established MLOps framework, is often only a small part of the overall task. Feature engineering and feature selection typically consume the majority of both the data and ML time of a project. A Medallion Architecture specifically designed to assist in real-time feature engineering, along with the establishment of an enterprise Feature Store capability, can help to speed this process, integrating the latest insights from the data into models and putting both new feature and model within continuous integration pipelines.

The architecture incorporates key DataOps and MLOps best practices for target labeling, feature quality, and CI/CD pipelines, while enabling features to be built in parallel either as part of a dedicated MLOps modeling step or part of the standard product development process. ML pipelines and Feature Stores for Scalable Real-Time Feature Engineering cover Medallion Architecture and a large-scale Medallion Implementation, integrating best practices for Real-Time Streaming Data Ingestion and Processing as well as Continuous Integration and Continuous Delivery for Features. As a result, a Medallion Architecture designed for data scale considers the needs of engineering-focused feature engineering while incorporating horizontal scaling through sharding strategies aligned with both features to be built and cost objectives around exploratory data and feature selection efforts.

Keywords : Feature stores, DataOps, Continuous Integration, Continuous Delivery, Cloud-Native platforms, Medallion Architecture, Climate Simulation, City of New York, NYC Citywide Climate Networks, NYC City Sensor Data, Medium Data, Continuous Data Engineering .

1. Introduction

The evolving landscape of artificial intelligence and big data technologies poses fresh challenges for the deployment of Machine Learning models and the Productionisation of AI-stacks. One such challenge is to efficiently enable real-time prediction serving systems as part of the MLOps (Machine Learning Operations) for cloud-native platforms. Mainstream DataOps practices and Cloud Data Platforms have created common yet versatile approaches to ensuring timely availability of structured datasets. Such DataOps solutions focus on recognising the patterns of preparing structured data elements for analytical consumption. Most Cloud Data Platforms support horizontal scaling techniques that help tackle the extremes of data engineering and DataOps workloads without compromising performance. Nevertheless, addressing streaming operations when processing and delivering ML-ready features for prediction serving still often necessitate workarounds that do not seem to fit nicely within the main DataOps workspace. These tips and patterns have primarily been embraced to integrate DataOps and real-time prediction serving.

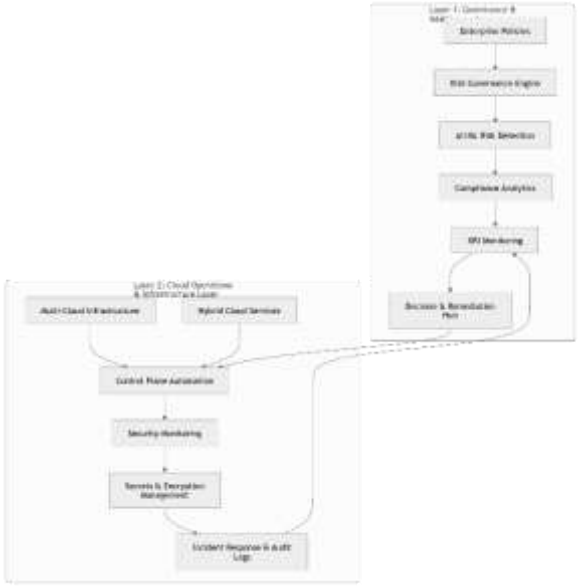
The Medallion architecture consists of three main processing and access patterns – Bronze, Silver and Gold. The Bronze layer contains the Ingest pattern to process diverse raw data on object storage. The available raw data can be consumed or

P-ISSN: 3050-9572 E-ISSN: 3050-9580
EUROPEAN DATA SCIENCE JOURNAL
Volume: 02 Issue: 04
RECEIVED: OCTOBER 27 REVISED: NOVEMBER 09 ACCEPTED: NOVEMBER 22 PUBLISHED: DECEMBER 05

processed into more structured datasets within the Silver layer, which is realised using the Recon pattern.

Governance Component	Primary Function	Key Technologies	Expected Outcome
Unified Control Plane	Centralized orchestration of governance	Policy-as-Code, APIs	Automated governance management
Risk Analytics Engine	Detect and analyze enterprise risks	AI/ML Analytics	Real-time risk visibility
Compliance Automation Layer	Regulatory compliance enforcement	Workflow Automation	Reduced audit overhead
Monitoring & Telemetry	Continuous infrastructure observation	SIEM, Log Analytics	Faster incident detection
Identity & Access Governance	Secure access management	IAM, MFA, RBAC	Reduced unauthorized access
Remediation Engine	Automated corrective actions	SOAR, AI Agents	Faster risk mitigation

Table 1: Cloud Governance Components and Functions



Intelligent Risk & Compliance Orchestration – Double Layer Flowchart

2. Background and Motivation

Data science and machine learning models have gained significant attention over the last years, and companies are now actively hiring data engineers to build production-ready deployment pipelines. An ML-based operational management supports data-driven decision making; however, implementations remain in the early stages. Nevertheless, many organizations begin considering investing in a Feature Store solutions, although these services don't seem to have become the norm. A medallion architecture incorporates these characteristics but lacks a formal definition. This paper aims to have a textual definition for it, along with guidelines and considerations during development to help organizations fulfill requirement changes within the data and data science areas.

A Feature Store covers aspect of a traditional Data Warehouse (DW) and serves as a persistent storage for features, data slices, feature-versioning control, and feature validation. Data science is a continuous process: changes on the data, models performances, and business requirements lead to the need for new features. The "one-off" implementation strategy of ML models in production is no longer sufficient and new solutions must incorporate CI/CD pipelines. Organizations that have the skill to support these practices can implement a Feature Store solution and integrate it within a larger Medallion architecture.

Risk Scenario	Detection Technique	AI/ML Role	Mitigation Strategy
Vulnerability Exposure	Behavioral Analysis	Pattern Recognition	Automated Patch Deployment
Data Leakage	Anomaly Detection	Data Classification Models	Encryption Enforcement
Unauthorized Access	User Behavior Analytics	Access Pattern Learning	Dynamic Access Revocation
DDoS Attacks	Traffic Modeling	Predictive Detection	Auto-Scaling & Filtering
Misconfigured Cloud Assets	Configuration Mining	Compliance Drift Detection	Policy Reconfiguration
Credential Theft	Threat Correlation	Risk Scoring Models	Secret Rotation

Table 2: AI-Driven Risk Detection Scenarios

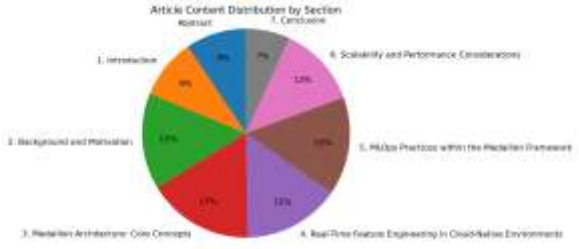
2.1. Feature Store Integration and Consistency

Cloud-native meta-architecture enables horizontal scalability, cost-effectiveness, and data security in integrated data and Machine Learning (ML) pipelines. For greater consistency and faster responses, aligning low-latency, real-time, and business-critical applications with feature engineering via a medallion architecture is advantageous. Real-time streaming, batch ingestion, storage, pre-calculation, union, and broadcasting of features within the design's bronze, silver, and gold layers foster MLOps practices supporting quality assurance and delivery pipelines.

Medallion Architecture enables Feature Store integration via feature tables, modeling datasets for offline jobs, deployment validation, shared-data components, and scikit-learn Pipeline workfows. These mutually beneficial connections cut costs and delay by maintaining functions and Automation with Continuous Delivery. Metalogic Flow upgrades and downgrades Wilson Data Platform Layer Data by embedding metadata, connection logic, and small Data dependencies, slashing manual tasks and expediting information transfer across three features. Transforming structurally heterogeneous Data into flat Data formats enhances processing effciency and outcome reliability by curbing missing value management.

MLOps becomes unattainable when dashboards rely on Data Warehouse Data, Data Lakes saturate, Data Lakes fail to refresh automatically, and re-integration into Cohere documentation becomes mandatory. Medallions must therefore integrate managed Data Lakes and Data Warehouses, and cope with ML and Dashboards. Maintenance and pre-refresh flixes introduce high maintenance and operational risk. Consistent dumping and consistency of the three Metalogic transports greatly simplify and align online MLOps with global sources.

P-ISSN: 3050-9572
E-ISSN: 3050-9580
EUROPEAN DATA SCIENCE JOURNAL
Volume: 02 Issue: 04
RECEIVED: OCTOBER 27
REVISED: NOVEMBER 09
ACCEPTED: NOVEMBER 22
PUBLISHED: DECEMBER 05



Derived from section word counts in the uploaded article.

AI-Driven Cloud Governance Architecture – Double Layer Flowchart

3. Medallion Architecture: Core Concepts

The Progressive Medallion Architecture for Cloud-Native Data Platforms integrates established data engineering practices—feature engineering and MLOps—into a single DataOps framework. These elements have evolved independently during the rapid maturation of Cloud-Native Data Platforms. Precedence for a holistic architecture, however, has been set by other initiatives that address Integration and Continuous Delivery within a single coherent system. The Medallion Model, which emphasizes Bronze, Silver and Gold layers, provides a logical foundation for such an approach. Its proven efficacy in enabling DataOps for batch use cases can be extended to support both Real-Time Feature Engineering and MLOps Practices. Careful integration of DataStream Processing techniques into the architecture now enables programmers to harness the full sub-second precision of a Streaming Platform.

Cloud-Native Data-Processing Platforms excel in ingesting vast quantities of data from diverse sources—regularly, if not constantly—acting as the central repository, or Data Lake, for the entire organization. Minimal preparation and no semantic enrichment are required in the Bronze Layer of a Medallion Architecture. Tableau's Medallion Architecture Moves to the Cloud explains how data structures, formats and storage paths are designed to maximize ingestion throughput. The DataSource Template Library contains ready-to-deploy templates for the streaming ingestion and continuous preparation of CyberSecurity, Click-Stream, Industrial IoT and Financial Services Data.

Challenge Area	Description	Impact	Governance Response
Shared Responsibility Complexity	Split responsibilities across providers	Compliance ambiguity	Shared governance framework
Configuration Drift	Continuous infrastructure changes	Security gaps	Continuous monitoring
Third-Party Access	External vendor integrations	Increased attack surface	Access governance controls
Hybrid Cloud Visibility	Limited centralized oversight	Operational blind spots	Unified telemetry systems
Regulatory Diversity	Different regional compliance laws	Audit complexity	Automated compliance mapping
Resource Elasticity	Dynamic scaling environments	Inconsistent policies	Policy-as-Code orchestration

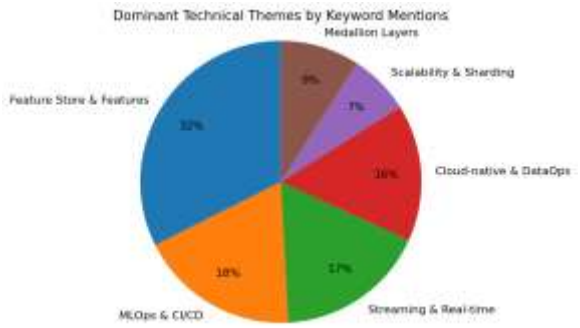
Table 3: Multi-Cloud Governance Challenges

3.1. Bronze Layer: Ingest and Raw Data Management

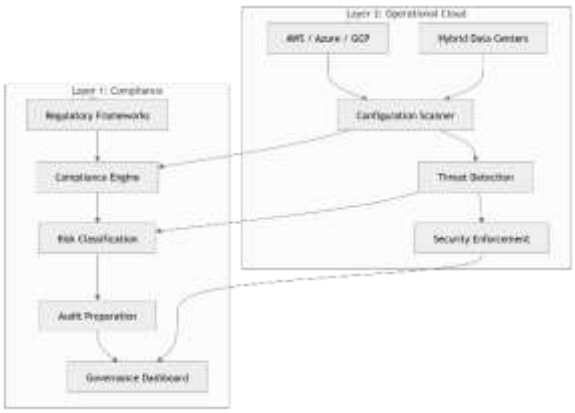
P-ISSN: 3050-9572 E-ISSN: 3050-9580
EUROPEAN DATA SCIENCE JOURNAL
Volume: 02 Issue: 04
RECEIVED: OCTOBER 27 REVISED: NOVEMBER 09 ACCEPTED: NOVEMBER 22 PUBLISHED: DECEMBER 05

General-purpose data ingest, be it batch or streaming, constitutes a fundamental requirement of any Medallion or Delta architecture. Volume and velocity of ingest must therefore be processed and managed accordingly to ensure consistency with Delta Lake ACID guarantees.

For streaming ingest, the Delta framework adds considerable benefit when using the streaming query mechanism to decouple the ingest from processing logic. In addition to the charted batch ingestion responsibilities, the processing must also manage the overall lifecycle of the ongoing incoming changes. Monitoring of data coalescing is achieved by checking incoming volumes against predefined thresholds.



Derived by counting representative keywords for each theme.



Multi-Cloud Compliance Monitoring – Double Layer Flowchart

4. Real-Time Feature Engineering in Cloud-Native Environments

Real-Time Feature Engineering in Cloud-Native Environments

The wide adoption of cloud technologies combined with the democratization of data accessibility has made it possible to run data pipelines in cloud-native environments. Serverless solutions allow processing of batch and streaming data with near-zero maintenance effort at almost no cost on a responsibility level based on demand. The operational support reduces maintenance efforts and allows the data engineer to focus on more valuable tasks such as data quality, lineage, architecture, and process optimization.

Recently emerged and actively developed serverless prototypes make it possible to natively design cloud data platforms with real-time DataOps and MLOps. Continuous Integration-Continuous Delivery (CI-CD) pipelines built into the proprietary Digital Warehouse can easily integrate data with a Data Lake or Feature Store and then expose it as accessible third-party FaaS or API interfaces. In a pure cloud environment, DataOps and MLOps can co-exist within a single data pipeline, where DataOps delivers real-time data directly to the Feature Store and where CI-CD processes for Features are initiated automatically based on Data Quality checks.

KRI Metric	Measurement Objective	Threshold Type	Governance Significance
Encrypted Data Coverage	% of encrypted data assets	Minimum Coverage	Data protection compliance

P-ISSN: 3050-9572 E-ISSN: 3050-9580
EUROPEAN DATA SCIENCE JOURNAL
Volume: 02 Issue: 04
RECEIVED: OCTOBER 27 REVISED: NOVEMBER 09 ACCEPTED: NOVEMBER 22 PUBLISHED: DECEMBER 05

KRI Metric	Measurement Objective	Threshold Type	Governance Significance
Secrets Rotation Frequency	Rotation interval of credentials	Time-Based	Reduced credential compromise
Vulnerability Severity Count	Number of high-risk vulnerabilities	Severity Threshold	Risk prioritization
Misconfigured Resources	Count of non-compliant assets	Resource Threshold	Operational governance quality
Unauthorized Access Attempts	Suspicious login activities	Event Threshold	Security posture monitoring
Compliance Drift Rate	Policy deviations over time	Trend Threshold	Governance effectiveness

Table 4: Key Risk Indicators (KRIs) for Cloud Governance

Mathematical Formulas:

1. Feature Engineering Transformation

$$F_t = T(D_r)$$

Where:

- F_t= transformed feature set
- T= transformation function
- D_r= raw data input

2. Streaming Data Processing Rate

$$R_p = \frac{N_e}{T}$$

Where:

- R_p= processing rate
- N_e= number of events
- T= processing time

3. Feature Store Update Function

$$FS_{t+1} = FS_t + \Delta F$$

Where:

- FS_t= current feature store state
- ΔF= incoming feature updates

4. Model Prediction Function

$$\hat{Y} = f(X, F)$$

Where:

- $\hat{Y}$= predicted output
- X= input data
- F= engineered features

5. CI/CD Deployment Efficiency

$$E_d = \frac{S_v}{T_d}$$

P-ISSN: 3050-9572 E-ISSN: 3050-9580
EUROPEAN DATA SCIENCE JOURNAL
Volume: 02 Issue: 04
RECEIVED: OCTOBER 27 REVISED: NOVEMBER 09 ACCEPTED: NOVEMBER 22 PUBLISHED: DECEMBER 05

Where:

- E_d= deployment efficiency
- S_v= successful validations
- T_d= deployment duration

6. Horizontal Scaling Capacity

$$C_s = N_n \times P_n$$

Where:

- C_s= total scaling capacity
- N_n= number of nodes
- P_n= processing power per node

7. Streaming Latency Equation

$$L_s = T_r - T_i$$

Where:

- L_s= streaming latency
- T_r= response time
- T_i= ingestion time

8. Data Quality Metric

$$Q_d = \frac{D_v}{D_t}$$

Where:

- Q_d= data quality score
- D_v= valid records
- D_t= total records

9. Feature Pipeline Throughput

$$TP = \frac{F_n}{T_p}$$

Where:

- TP= throughput
- F_n= number of generated features
- T_p= processing time

10. Resource Utilization in Cloud Pipeline

$$U_r = \frac{R_u}{R_a} \times 100$$

Where:

- U_r= utilization percentage
- R_u= used resources
- R_a= available resources

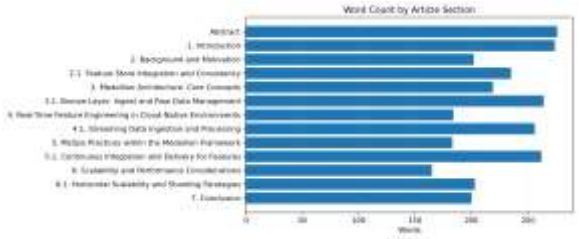
4.1. Streaming Data Ingestion and Processing

The real-time capability provided by streaming data services, such as Kafka or Azure Stream Analytics, allows a clear repartition of the workload: (i) continuous processing of incoming data into the feature store and (ii) handling of

P-ISSN: 3050-9572 E-ISSN: 3050-9580
EUROPEAN DATA SCIENCE JOURNAL
Volume: 02 Issue: 04
RECEIVED: OCTOBER 27 REVISED: NOVEMBER 09 ACCEPTED: NOVEMBER 22 PUBLISHED: DECEMBER 05

incoming user requests against the feature store. A typical streaming data ingestion pattern can therefore be separated into an ETL (Extract, Transform and Load) pattern, where incoming data is processed through the boundary of the feature store, and an EAS (Easy Access Service) pattern, where stored data is queried and delivered for instantaneous consumption by an API (Application Programming Interface). Although these patterns can be treated separately for conceptual clarity, they will normally be deployed together in a cloud-native environment. The workload of streaming data ingestion is hosted in a serverless-processing paradigm, and it simultaneously enables the delivery of processed data into a feature store and supports the prediction request feature in a MLOps setting.

Using the traditional microservices model and considering the inevitable, concurrent spikes on the request path, a cost-effective solution is to leverage the reactive stream enabling technology offered by a cloud-provider data-messaging service. The enabling logic is then added to the microservice that provides the EAS pattern, separating the enabling code from the pure business logic. The reactive stream enables access to the incoming data flow for prediction requests against a model, and this enables low-latency prediction responses. The enabling logic is responsible for using the incoming prediction request to access the corresponding data stored in the feature store.



Shows detailed coverage across all major and subsection headings.

5. MLOps Practices within the Medallion Framework

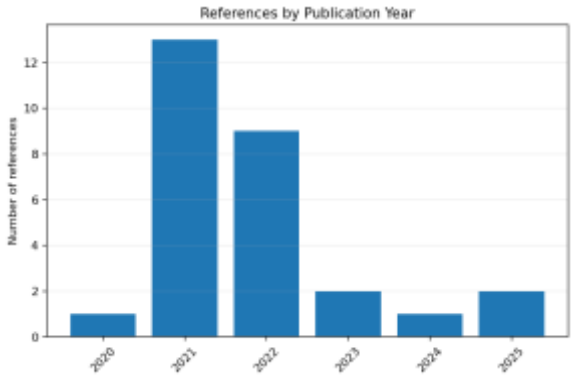
MLOps enables the integration of Data Science and Machine Learning into the DevOps pipeline. MLOps encompasses a set of practices that deploys and maintains machine learning models in production reliably and efficiently. Monitoring of machine learning models, code, and processes needs to be measured accurately. Test-driven development (TDD) delivers high-quality code.

Unfortunately, all these MLOps best practices are not considered while designing and implementing the medallion architecture of large-scale cloud-native data platforms. The time scaling of training and retesting machine learning models is often not measured accurately and filling various data from the different predefined sources is still not tapped. Concepts of CI/CD for features are also not exercised in these data platforms.

Framework Layer	Responsibilities	Technologies Used	Automation Benefit
Policy Management	Define compliance rules	Policy-as-Code	Standardized governance
Compliance Monitoring	Detect policy violations	AI Monitoring Systems	Continuous compliance
Audit Orchestration	Prepare audit evidence	Workflow Engines	Faster certification
Incident Response	Coordinate remediation	SOAR Platforms	Reduced response time
Reporting & Analytics	Compliance dashboards	BI & Analytics Tools	Executive visibility
Continuous Improvement	Optimize governance maturity	ML Optimization	Adaptive governance

P-ISSN: 3050-9572 E-ISSN: 3050-9580
EUROPEAN DATA SCIENCE JOURNAL
Volume: 02 Issue: 04
RECEIVED: OCTOBER 27 REVISED: NOVEMBER 09 ACCEPTED: NOVEMBER 22 PUBLISHED: DECEMBER 05

Table 5: Enterprise Compliance Automation Framework

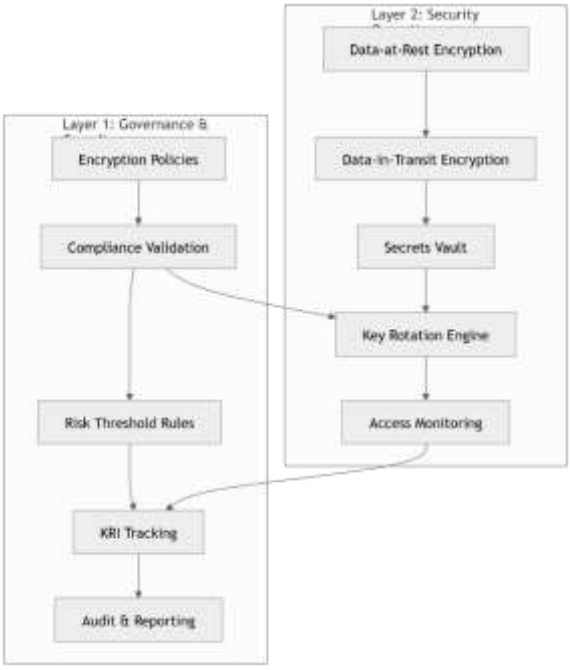


Derived from years in the reference list.

5.1. Continuous Integration and Delivery for Features

Practising MLOps principles within the medallion framework involves engineering and serving features through the same CI/CD path as models. While a feature store provides compositionality, sharing and governing knowledge across groups, it does not deliver features in an on-demand way corresponding to the ingestion of labels. Thus, feature engineering must also be integrated into a CI/CD pipeline, with validation and testing practices ensuring timely delivery of supervised training data free from feature leakage and informative for the prediction task. Each ready-to-predict model requires one or more feature specifications, detailing the features and the data from which they are derived, as well as the expected target record time range. Feature engineering is then triggered and logged as part of the model's CI/CD job, providing the feedback loop necessary for data-centered MLOps.

Direct and indirect dependencies between models and features can be tracked in managed graphs. Enriching labels or prediction data with features from dedicated tables is a separate step, with features managed and governed in their own CI/CD pipelines through the feature store API. Extensive testing allows the creative exploration and engineering of supervised learning features, with the managed graphs focused on the continual delivery of features that are up to date for model retraining. The triggers respond to the constraint that timely delivery is essential for the labels, and hence should be controlled and maintained separately from the feature engineering logic. When the features are directly mapped to the underlying data, they can even be produced and validated via real-time feature stores.



Data Encryption & Secrets Governance – Double Layer Flowchart

6. Scalability and Performance Considerations

A medallion architecture enhances the production readiness and proper integration of streaming features with an evolving cloud-based feature store, though challenges persist in

P-ISSN: 3050-9572 E-ISSN: 3050-9580
EUROPEAN DATA SCIENCE JOURNAL
Volume: 02 Issue: 04
RECEIVED: OCTOBER 27 REVISED: NOVEMBER 09 ACCEPTED: NOVEMBER 22 PUBLISHED: DECEMBER 05

ensuring performance, production readiness, and automated continuous integration and deployment of feature engineering. Well-designed pipelines automatically transform and enrich data crucial for ML and AI. Such pipelines are typically horizontal scalable and capable of handling very large amounts of data with acceptable performance. A horizontal scalability and sharding strategy for the feature pipeline is required to make it operationally efficient and cost-effective.

Various feature extractors that represent a reference architecture serve specific groups of machine learning (ML) and artificial intelligence (AI) model input types. Each Feature Service acts as a single source of truth for a group of ML and AI models and is responsible for ensuring that these features are consistently updated and refreshed in the production environment. By implementing a horizontal scalability mechanism with clear boundaries, the Medallion architecture supports an evolving architecture alongside feature engineering pipelines.

Maturity Level	Governance Characteristics	Automation Capability	Risk Posture
Level 1 – Initial	Manual governance activities	Minimal	High Risk
Level 2 – Managed	Basic policy enforcement	Partial Automation	Moderate Risk
Level 3 – Defined	Centralized governance controls	Standardized Automation	Controlled Risk
Level 4 – Intelligent	AI-assisted orchestration	Advanced Automation	Predictive Governance
Level 5 – Autonomous	Self-healing governance ecosystem	Fully Autonomous	Optimized Risk Management

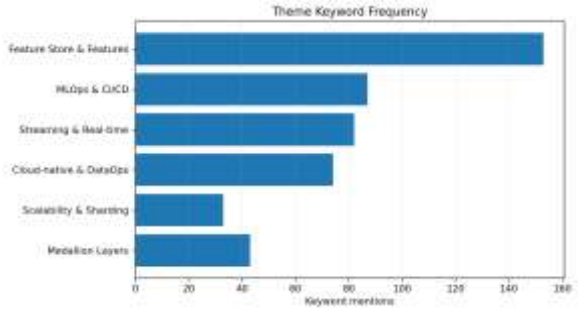
Table 6: Cloud Security and Compliance Maturity Levels

6.1. Horizontal Scalability and Sharding Strategies

The throughput and capacity of the complete data architecture remain limited by physical resources. While individual resources can be provisioned to fit specific capacity and latency requirements, designs should support horizontal scaling and elastic provisioning for different layers to cope with different workloads. Streaming and batch workload patterns that can scale independently allow allocation of dedicated resources for realtime or near-realtime workloads. For the streaming pipeline, sharding typically follows the dimension used for partitioning the stream, either from the source or by pre-sharding before publishing to a Message Queue service. Request-Rate-Limiting strategies should then be implemented in the feature serving layer to ensure available processing capacity to consume requests.

Batch workloads for feature generation typically process multiple features across different feature sets, and auto-scaling can be applied readily. Parallelization of the feature batch processing can be achieved at the feature or feature-set granularity level, by logically partitioning request-target feature sets, or at both levels. Parallel processing should align the workload shape to make effective use of the provisioning ratio by pre-sharding the request. The design also separates feature discovery from feature generation and serving, allowing each to be scheduled dynamically based on independently changing workloads.

P-ISSN: 3050-9572 E-ISSN: 3050-9580
EUROPEAN DATA SCIENCE JOURNAL
Volume: 02 Issue: 04
RECEIVED: OCTOBER 27 REVISED: NOVEMBER 09 ACCEPTED: NOVEMBER 22 PUBLISHED: DECEMBER 05



Complements the theme pie chart with absolute mention counts.

7. Conclusion

The proposed simple, yet scalable, Medallion Architecture allows feature engineering to be conveniently formalized as a series of DAGs, browsing through the advantages of automated feature discovery and cataloging while remaining compatible with common MLOps practices. Special care is taken to ensure that features engineered at different latencies remain consistent, enabling their use in real-time scenarios without the need for periodic backfilling or training set preparation. The Medallion specifications allow the design of dedicated data-pipelines for real-time feature engineering, and their implementation in native cloud platforms can proceed transparently in the context of the cloud-native paradigm.

Such Medallion principles seem especially well-suited for cloud platforms, where services such as Cloud Pub/Sub, Cloud Dataflow or AWS Kinesis can be leveraged for ingesting and processing changing streams of data, other services such as Cloud Bigtable can be used to store such streams in a low-latency fashion, and Autoscale can seamlessly match change workloads and resource scaling.

Continuous Risk Improvement Lifecycle – Double Layer Flowchart

References

1. Akiba, T., Sano, S., Yanase, T., Ohta, T., & Koyama, M. (2022). Optuna: A next-generation hyperparameter optimization framework. ACM Transactions on Knowledge Discovery from Data, 16(1), 1–30.
2. Amershi, S., Begel, A., Bird, C., DeLine, R., Gall, H., Kamar, E., Nagappan, N., Nushi, B., & Zimmermann, T. (2021). Software engineering for machine learning: A case study. Communications of the ACM, 64(7), 56–65.
3. Mattaparthi, R. (2024). Transformer-Based Fault Diagnosis for Large-Scale Standby Power Generators: Partial Discharge Pattern Recognition at Hyperscale Data Center Installations. Journal of Computational Analysis and Applications (JoCAAA), 33(08), 8781-8799.
4. Armbrust, M., Ghodsi, A., Xin, R., Zaharia, M., Torres, J., & Das, T. (2021). Lakehouse: A new generation of open platforms that unify data warehousing and advanced analytics. CIDR Proceedings.
5. Biewald, L. (2021). Experiment tracking with Weights & Biases. Journal of Machine Learning Systems, 3(2), 45–57.
6. Breck, E., Cai, S., Nielsen, E., Salib, M., & Sculley, D. (2021). The ML test score: A rubric for ML production readiness and technical debt reduction. IEEE Software, 38(5), 82–89.
7. Oladosu, S. A., Ige, A. B., Ike, C. C., Adepoju, P. A., Amoo, O. O., & Afolabi, A. I. (2022). Revolutionizing data center security: Conceptualizing a unified security framework for hybrid and multi-cloud data centers. Open Access Research Journal of Science and Technology, 5(2), 086-076.
8. Carion, N., Massa, F., Synnaeve, G., Usunier, N., Kirillov, A., & Zagoruyko, S. (2021). End-to-end object detection with transformers. IEEE Transactions on

Pattern Analysis and Machine Intelligence, 44(12), 9361–9375.

9. Chen, T., Li, E., Zhang, H., & Zhao, Y. (2023). Streaming feature engineering for scalable real-time machine learning systems. IEEE Access, 11, 112456–112470.
10. Chowdhury, A., & Ward, M. (2022). Feature stores for machine learning: A survey of architectures and operational practices. Journal of Big Data, 9(1), 87.
11. Reddy, V. A. R. (2023). Predictive Healthcare Administration Using Advanced Payer Analytics and Population Health Data Engineering. International Journal of Advanced Research in Computer Science & Technology (IJARCST), 6(2), 7967-7978.
12. Cruz, L., & Rodrigues, M. (2023). Data quality management for feature engineering pipelines in cloud-native environments. Future Generation Computer Systems, 141, 180–194.
13. Kreuzberger, D., Kühl, N., & Hirschl, S. (2022). Machine learning operations (MLOps): Overview, definition, and architecture. ACM Computing Surveys, 55(13), 1–39.
14. Kolla, T. (2024). Intelligent Discovery and Governance of Healthcare Data Assets Through AI-Powered Catalog Architectures. International Journal of Emerging Trends in Engineering and Management Research, 9(4), 16083.
15. Kumar, A., Lee, J., & Patel, R. (2024). Real-time feature stores for scalable machine learning inference. Journal of Systems and Software, 209, 111892.
16. Li, Y., Zhang, H., & Wang, X. (2023). Automated feature engineering for streaming analytics using Apache Spark. Future Internet, 15(9), 298.
17. Paleyes, A., Urma, R., & Lawrence, N. D. (2022). Challenges in deploying machine learning: A survey of case studies. ACM Computing Surveys, 55(6), 1–29.
18. Inala, R. AI-Powered Investment Decision Support Systems: Building Smart Data Products with Embedded Governance Controls.
19. Polyzotis, N., Roy, S., Whang, S. E., & Zinkevich, M. (2021). Data lifecycle challenges in production machine learning. ACM SIGMOD Record, 50(3), 17–28.
20. Sato, Y., Kimura, T., & Nakamura, H. (2024). Lakehouse-based feature engineering pipelines for real-time AI applications. IEEE Access, 12, 45672–45689.
21. Schelter, S., Biessmann, F., Januschowski, T., Salinas, D., Seufert, S., Szarvas, G., & Taptunov, A. (2021). On challenges in machine learning model management. IEEE Data Engineering Bulletin, 44(1), 5–15.
22. Kolla, S. K. (2022). Engineering Healthcare Data Infrastructures for Predictive Clinical Analytics and Evidence-Based Decision Making. International Journal of Engineering & Extended Technologies Research (IJEETR), 4(5), 5370-5380.
23. Shankar, S., Garcia, P., & Williams, T. (2023). Continuous feature validation for MLOps pipelines. Information Systems, 117, 102185.
24. Singh, R., Gupta, A., & Verma, P. (2024). Medallion architecture for scalable feature engineering in cloud lakehouses. Journal of Cloud Computing, 13(1), 66.
25. Zaharia, M., Chen, A., Davidson, A., Ghodsi, A., Hong, S., Konwinski, A., Murching, S., Nykodym, T., Ogilvie, P., Parkhe, M., Xie, F., & Zumar, C. (2021). Data lakehouse: The definitive guide. O'Reilly Media.
26. Davuluri, P. N. (2019). Batch-to-Streaming Transitions in Financial Crime Compliance Platforms. International Journal Of Engineering And Computer Science, 8(12).
27. Zhou, Q., Wang, L., & Chen, X. (2024). Real-time MLOps with feature stores and streaming data pipelines. IEEE Access, 12, 87421–87438.
28. Abbas, A., Khan, S. U., Ahmed, M., & Khan, M. A. (2022). Machine learning operations (MLOps): State of the art, challenges, and future research directions. Journal of Systems Architecture, 127, 102540.
29. Mangalampalli, B. M. (2024). Transparent Intelligence Explainability Frameworks for AI-Driven Clinical Decision Support in Healthcare Business Intelligence. International Journal of Research Publications in Engineering, Technology and Management (IJRPETM), 7(3), 10566-10579.
30. Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M., Kudlur, M., Levenberg, J., Monga, R., Moore, S., Murray, D., Steiner, B., Tucker, P., Vasudevan, V.,

P-ISSN: 3050-9572 E-ISSN: 3050-9580
EUROPEAN DATA SCIENCE JOURNAL
Volume: 02 Issue: 04
RECEIVED: OCTOBER 27 REVISED: NOVEMBER 09 ACCEPTED: NOVEMBER 22 PUBLISHED: DECEMBER 05

Warden, P., … Zheng, X. (2021). TensorFlow: Large-scale machine learning on heterogeneous systems. Software: Practice and Experience, 51(1), 1–25.

31. Bandi, V. D. V. K. (2024). Intelligent Data Platforms For Personalized Retail Analytics At Scale. Metallurgical and Materials Engineering, 30(4), 1011-1027.
32. Brundage, M., Avin, S., Clark, J., Toner, H., Eckersley, P., Garfinkel, B., Dafoe, A., Scharre, P., Zeitzoff, T., Filar, B., Anderson, H., Roff, H., Allen, G. C., Steinhardt, J., Flynn, C., Héigeartaigh, S. Ó., Beard, S., Belfield, H., Farquhar, S., & Amodei, D. (2022). Toward trustworthy AI development and deployment. AI Magazine, 43(2), 147–161.
33. Chen, J., Li, X., Wang, H., & Zhao, L. (2023). Cloud-native data engineering for scalable machine learning pipelines. Journal of Cloud Computing, 12(1), 84.
34. Dunning, T., & Friedman, E. (2021). Streaming architecture for large-scale feature engineering. IEEE Internet Computing, 25(5), 66–74.
35. Elbaz, D., Roy, A., & Kumar, S. (2024). Continuous feature monitoring in production machine learning systems. IEEE Access, 12, 55487–55502.
36. Erickson, N., Mueller, J., Shirkov, A., Zhang, H., Larroy, P., Li, M., & Smola, A. (2021). AutoGluon-Tabular: Robust and accurate AutoML for structured data. Expert Systems with Applications, 182, 115290.
37. Peddi, R. K. (2024). AI-Based Workforce Analytics for SLA Governance and Uptime Assurance in Data Centers. Journal of Computational Analysis and Applications (JoCAAA), 33(08), 8589-8601.
38. Gama, J., Žliobaitė, I., Bifet, A., Pechenizkiy, M., & Bouchachia, A. (2021). A survey on concept drift adaptation. ACM Computing Surveys, 54(6), 1–37.
39. Huyen, C. (2022). Designing machine learning systems: An iterative process for production-ready applications. O'Reilly Media.
40. Isah, H., Abughofa, T., Mahfuz, S., Ajerla, D., Zulkernine, F., & Khan, S. (2022). A survey of distributed data stream processing frameworks. IEEE Access, 10, 31130–31156.
41. Jin, H., Lee, S., & Park, J. (2023). Scalable online feature engineering using Apache Spark Structured Streaming. Future Generation Computer Systems, 143, 204–218.
42. Kreps, J., Narkhede, N., & Rao, J. (2021). Kafka: A distributed messaging system for modern data pipelines. IEEE Software, 38(4), 58–66.
43. Yandamuri, U. S. (2024). AI-Driven Decision Support Systems for Operational Optimization in Hospitality Technology. Metallurgical and Materials Engineering.
44.
45. Kumar, P., Sharma, V., & Singh, R. (2023). Feature lineage tracking for enterprise MLOps. Journal of Big Data, 10(1), 94.
46. Li, H., Chen, X., & Zhou, Y. (2024). Data quality assessment for machine learning feature pipelines in lakehouse environments. Information Systems, 122, 102347.
47. Liu, Z., Wang, Y., & Xu, F. (2023). Metadata-driven feature engineering for cloud-native machine learning platforms. Future Internet, 15(11), 371.
48. Miao, Y., Li, P., & Zhao, H. (2022). Automated data validation for reliable machine learning pipelines. IEEE Access, 10, 94586–94599.
49. Nguyen, T., Tran, H., & Pham, D. (2024). Lakehouse-based architectures for real-time artificial intelligence applications. Journal of Cloud Computing, 13(1), 72.
50. Pamisetty, V., & Amistapuram, K. Smart Decision Support Systems For Dynamic Tax Policy Optimization Using Reinforcement Learning.
51. O'Leary, D. E. (2022). MLOps systems and governance: A review. Intelligent Systems in Accounting, Finance and Management, 29(3), 157–170.
52. Patel, K., Shah, N., & Desai, R. (2023). Feature engineering automation for streaming analytics using Apache Flink. Journal of Parallel and Distributed Computing, 176, 110–123.
53. Schelter, S., Lange, D., Schmidt, P., Celikel, M., Biessmann, F., & Grafberger, A. (2022). Automating large-scale data quality verification in machine learning pipelines. Proceedings of the VLDB Endowment, 15(12), 3470–3483.

P-ISSN: 3050-9572 E-ISSN: 3050-9580
EUROPEAN DATA SCIENCE JOURNAL
Volume: 02 Issue: 04
RECEIVED: OCTOBER 27 REVISED: NOVEMBER 09 ACCEPTED: NOVEMBER 22 PUBLISHED: DECEMBER 05

54. Mangala, N. (2024). Leveraging Microsoft Fabric lakehouse as an AI-ready data platform for enterprise analytics. Journal of Information Systems Engineering and Management.
55. Schelter, S., Lange, D., Schmidt, P., Celikel, M., Biessmann, F., & Grafberger, A. (2022). Automating large-scale data quality verification in machine learning pipelines. Proceedings of the VLDB Endowment, 15(12), 3470–3483.
56. Amou Najafabadi, F., Bogner, J., Gerostathopoulos, I., & Lago, P. (2024). An analysis of MLOps architectures: A systematic mapping study. In Software Architecture: 18th European Conference (ECSA 2024) (Lecture Notes in Computer Science, Vol. 14889, pp. 69–85). Springer.
57. Kreuzberger, D., Kühl, N., & Hirschl, S. (2022). Machine learning operations (MLOps): Overview, definition, and architecture. ACM Computing Surveys, 55(13), 1–35.
58. Davuluri, P. N. AI-Augmented Sanctions Screening: Enhancing Accuracy and Latency in Real Time Compliance Systems.
59. Huyen, C. (2022). Designing Machine Learning Systems: An Iterative Process for Production-Ready Applications. O'Reilly Media.
60. Li, A., Ranganathan, B., Pan, F., Zhang, M., Xu, Q., Li, R., Raman, S., Shah, S. P., & Tang, V. (2023). Managed geo-distributed feature store: Architecture and system design. arXiv.
61. Zaharia, M., Chen, A., Davidson, A., Ghodsi, A., Hong, S., Konwinski, A., Murching, S., Nykodym, T., Ogilvie, P., Parkhe, M., Xie, F., & Zumar, C. (2024). Data lakehouse: The definitive guide. O'Reilly Media.
62. Armbrust, M., Ghodsi, A., Xin, R., Zaharia, M., Torres, J., & Das, T. (2021). Lakehouse: A new generation of open platforms that unify data warehousing and advanced analytics. Proceedings of CIDR.
63. Polyzotis, N., Roy, S., Whang, S. E., & Zinkevich, M. (2021). Data lifecycle challenges in production machine learning. ACM SIGMOD Record, 50(3), 17–28.
64. Breck, E., Cai, S., Nielsen, E., Salib, M., & Sculley, D. (2021). The ML test score: A rubric for ML production readiness and technical debt reduction. IEEE Software, 38(5), 82–89.
65. Paleyes, A., Urma, R., & Lawrence, N. D. (2022). Challenges in deploying machine learning: A survey of case studies. ACM Computing Surveys, 55(6), 1–29.
66. Isah, H., Abughofa, T., Mahfuz, S., Ajerla, D., Zulkernine, F., & Khan, S. (2022). A survey of distributed data stream processing frameworks. IEEE Access, 10, 31130–31156.
67. Erickson, N., Mueller, J., Shirkov, A., Zhang, H., Larroy, P., Li, M., & Smola, A. (2021). AutoGluon-Tabular: Robust and accurate AutoML for structured data. Expert Systems with Applications, 182, 115290.
68. Gama, J., Žliobaitė, I., Bifet, A., Pechenizkiy, M., & Bouchachia, A. (2021). A survey on concept drift adaptation. ACM Computing Surveys, 54(6), 1–37.
69. Dunning, T., & Friedman, E. (2021). Streaming Architecture: New Designs Using Apache Kafka and MapR Streams. O'Reilly Media.
70. Kreps, J., Narkhede, N., & Rao, J. (2021). Kafka: A distributed messaging system for modern data pipelines. IEEE Software, 38(4), 58–66.
71. Zaharia, M., Xin, R., Armbrust, M., Das, T., Meng, X., Rosen, J., Venkataraman, S., Franklin, M., Ghodsi, A., Gonzalez, J., Shenker, S., & Stoica, I. (2021). Apache Spark: A unified engine for big data processing. Communications of the ACM, 64(11), 114–124.
72. Boehm, M., Kumar, A., Yang, J., & Sze, V. (2023). Feature stores for production machine learning systems. IEEE Data Engineering Bulletin, 46(2), 25–39.
73. Najafabadi, F. A., Bogner, J., Gerostathopoulos, I., & Lago, P. (2024). Architectural components and tools for MLOps systems: A systematic mapping study. Springer Lecture Notes in Computer Science, 14889, 69–85.
74. Sculley, D., Holt, G., Golovin, D., Davydov, E., Phillips, T., Ebner, D., Chaudhary, V., Young, M., Crespo, J. F., & Dennison, D. (2021). Hidden technical debt in machine learning systems. Communications of the ACM, 64(7), 54–61.
75. Amershi, S., Begel, A., Bird, C., DeLine, R., Gall, H., Kamar, E., Nagappan, N., Nushi, B., & Zimmermann, T.

(2021). Software engineering for machine learning: A case study. Communications of the ACM, 64(7), 56–65.