Scientific Publications

Journal of Artificial Intelligence and Big Data, 2023, 3, 1368
www.scipublications.org/journal/index.php/jaibd
DOI: 10.31586/jaibd.2023.1368

Review Article

MLOps Frameworks for Reliable Model Deployment in Cloud Data Platforms

Velangani Divya Vardhan Kumar Bandi [1,*]

[1] Sr Data Engineer, USA

*Correspondence: Velangani Divya Vardhan Kumar Bandi (vdivyavardhankb@gmail.com)

Abstract: Machine learning operations (MLOps) comprises the practices, methods, and tooling that facilitate the deployment of reliable ML models in production environments. While many aspects of cloud data platforms are designed to enable reliability, only some managed ML services support the MLOps goals of continuous integration, continuous delivery, data lineage tracking, associated reproducibility, governance, and security. Furthermore, reliability encompasses not only the fulfillment of service-level objectives, but also systematic monitoring, alerting, and incident response automation. Architectural patterns are proposed to enable reliable deployment in cloud data platforms, focusing on the implementation of continuous integration and testing pipelines for ML models and the formulation of continuous delivery and rollout strategies. Continuous integration pipelines reduce the risk of regressions and ensure sufficient model performance at the time of deployment, while continuous delivery pipelines enable rapid updates to production models within acceptable risk profiles. The landscape of publicly available MLOps frameworks, tools, and services is also examined, emphasizing the pros and cons of established and rising solutions in containerization, orchestration, model serving, and inference. Containerization and orchestration contributes to the building of reliable deployment pipelines in cloud data platforms, whether general-purpose tools (e.g. Docker and Kubernetes) or solutions tailored for ML workloads. Containerized serving frameworks designed for high-throughput, low-latency inference can benefit a wide range of business applications, while auto-scaling and model versioning capabilities enhance the ease of use of cloud-native ML services.

Keywords: MLOps, Cloud-Native Machine Learning, Continuous Integration for ML, Continuous Delivery of Models, ML Reliability Engineering, Model Deployment Pipelines, Data Lineage and Reproducibility, ML Governance and Security, Service-Level Objectives (SLOs), Monitoring and Incident Response Automation, CI/CD for Machine Learning, Model Rollout Strategies, Containerization for ML (Docker), Orchestration for ML Workloads (Kubernetes), Model Serving Frameworks, High-Throughput Low-Latency Inference, Auto-Scaling ML Services, Model Versioning, Cloud Data Platform Architecture, Production ML Observability

How to cite this paper:
Kumar Bandi, V. D. V. (2023). MLOps Frameworks for Reliable Model Deployment in Cloud Data Platforms. *Journal of Artificial Intelligence and Big Data, 3*(1), 81-101.
DOI: 10.31586/jaibd.2023.1368

Received: September 21, 2023
Revised: October 31, 2023
Accepted: December 21, 2023
Published: December 27, 2023

1. Introduction

Machine learning operations (MLOps) encompasses the practices and the tools for the collaboration of data scientists, data engineers, operations support, and software engineers to produce, deploy, and maintain machine learning (ML) models in a reliable manner [1]. Data-driven organizations in private and public sectors are increasingly adopting cloud-native solutions powered by public cloud platforms [2]. Such solutions leverage cloud-native patterns for building and deploying machine learning models; accelerate the process of creating a production-ready ML stack by the automated provisioning of data lakes, object storage buckets, databases, orchestration engines, container registries, and virtual clusters; and alleviate the burden of managing infrastructure [3].

The reliability of cloud-native ML deployments benefits from automated model testing and validation, incident detection and response, and model performance monitoring [4]. However, when not properly configured, such cloud services may introduce a single point of failure in the ML production pipeline or not meet the performance requirements [5]. When models are deployed in containers, cloud-native CI/CD patterns can be used for continuous testing and validation. In this context, the reliability of deployment, encompassing continuous integration and continuous delivery processes, comprises multiple components: continuous integration pipelines guarantee the correctness of models and validate the quality of the data, and deployment strategies address the risk inherent to each rollout [6].

1.1. Overview of MLOps and Its Significance in Cloud Environments

MLOps refers to a set of practices that seeks to deploy and maintain ML (Machine Learning) models in production reliably and efficiently. ML has gained popularity in recent years and has many business applications [7]. CI/CD (Continuous Integration/Continuous Delivery) processes have proven very useful in the deployment of software, and similar processes can be established for maintaining ML workloads [8]. Enabling them through the automation of tools in Cloud Data Platforms also addresses ML-specific needs for data lineage, reproducibility, governance, and security. It is when Cloud Data Platforms are linked by automation that models can be delivered reliably, with constant observability of key metrics for development and operational teams [9].

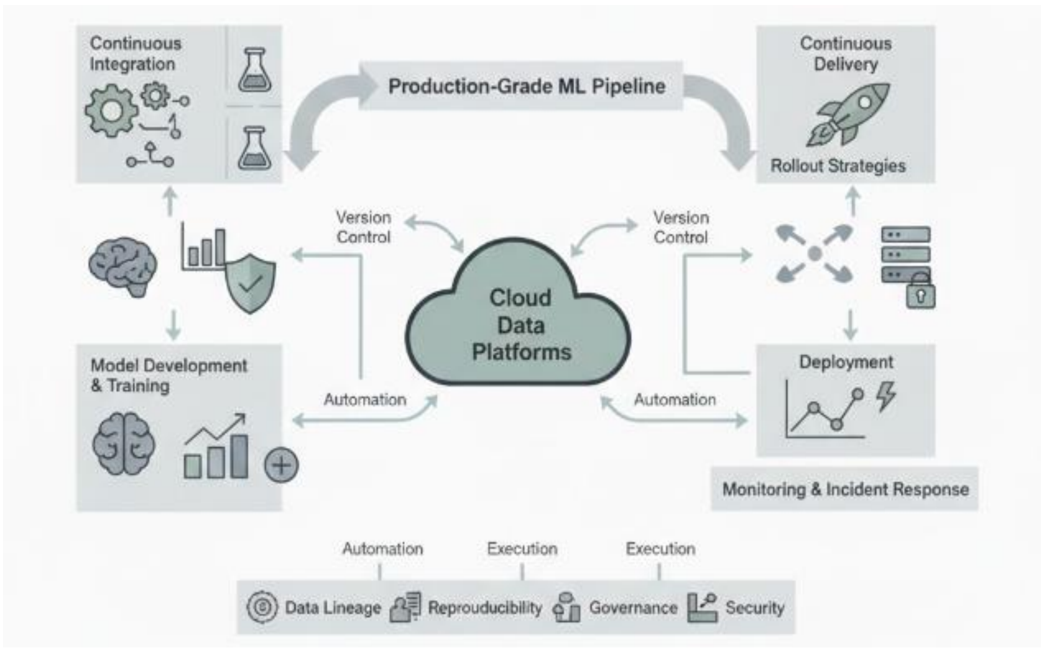

Figure 1. Synchronized Forwards: A Systematic Template for Automated CI/CD and Observability in Cloud-Native MLOps Ecosystems

The goal of this overview of the foundations of MLOps in Cloud Environments is to identify the capabilities needed in Cloud Data Platforms and the tools that fulfill them so that the deployment of ML workloads follows CI/CD processes [10]. Automation is key for reliability and is provided for the various steps in the processes, especially Monitoring and Incident Response [11]. The proposed process template is based on pairs of Forwards, one each for Continuous Integration and Continuous Delivery [12]. Each Forward indicates a Continuous Integration Pipeline for Models that integrates CI aspects, possibly with additional validations, and a Continuous Delivery Pipeline for Models that covers the rest of the aspects of CI/CD, including Rollout Strategies for Models [13].

2. Foundations of MLOps in Cloud Environments

MLOps for cloud data platforms must ensure reliable deployment through automated CI/CD pipelines, active monitoring, and structured incident response. Several aspects contribute to this aim: reliable delivery of ML models, automated rollout strategies to mitigate risk, careful choice of tooling, defining key metrics, and the presence of a mature MLOps ecosystem. With the increasing adoption of public cloud services and deployment of production-level workloads, ensuring reliable execution is critical [14].

Building and deploying ML models is a multi-step process, as outlined in Chapter 1. An MLOps framework establishes CI/CD pipelines to automate these steps, including continuous integration testing of models and constant delivery to production [15]. However, reliability demands more than correct delivery [16]. Monitoring detects incidents, while clear procedures guide response, including reverting to backup models in case of failure. Real-world S3 incidents demonstrated that even foundational cloud services are not always reliable and highlighted the vulnerability of AWS-hosted webpages without a CDN [17].

Table 1. Observed Reliability and Performance Metrics for ML Inference Services

Metric	Value
Availability	0.96875
Latency p50 (ms)	45.60916859994713
Latency p95 (ms)	83.27928987399044
Latency p99 (ms)	112.2915089738782
Avg throughput (req/min)	872.2756944444444

2.1. Key Principles for Implementing MLOps in Cloud-Based Systems

Implementation of MLOps in cloud-based systems can be made reliable by supporting, monitoring, and maintaining these principal tasks: (i) the continuous integration of production models into the codebase, (ii) the validation of training and production data, (iii) the monitoring of production data for drift, (iv) using a canary or blue-green strategy for rollout to production, (v) rollback of the production model in the event of failure, (vi) monitoring the availability and performance of the model serving infrastructure, (vii) monitoring additional quality metrics for the production model, (viii) enabling autoscaling of the model serving infrastructure, and (ix) implementing dashboards for all reliability and quality metrics [18].

The reliability of an ML model in production cannot be guaranteed solely by performing the conventional practices of unit test and continuous deployment; instead, reliability is achieved through a specific reliability-oriented platform. Figure 1 shows the architectural patterns needed for reliable deployment [19]. The validation of both training and production data should be included as part of the pipeline [20]. When a predictive model is deployed in production, it is essential that the distribution of the input data does not change over time, a case known as data drift. These monitoring checks should be in place, with the ability to instantaneously notify the data engineering team should any anomaly occur [21].

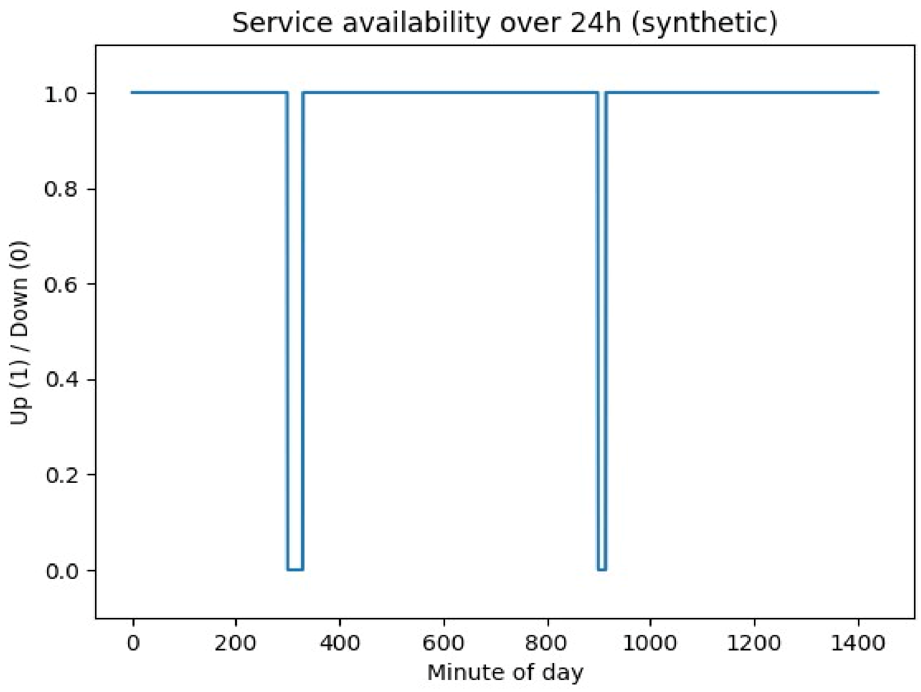

Figure 2. Mathematical Model for Service Availability Measurement in ML Production Systems

Equation A: Availability (uptime ratio)

Step 1 — define total observation window

- Let total observed time be T (e.g., in minutes or seconds).

Step 2 — split time into uptime and downtime

- Let uptime be T_{up}
- Let downtime be T_{down}
- Then:

$$T = T_{\mathrm{up}} + T_{\mathrm{down}}$$

Step 3 — define availability

$$A = \frac{T_{\mathrm{up}}}{T}$$

Step 4 — alternate discrete (ping/check) form

If you check the service N times (e.g., every minute), define an indicator:

$$I_i = \begin{cases} 1 & \text{service is up on check } i \\ 0 & \text{service is down on check } i \end{cases}$$

Then:

$$T_{\mathrm{up}} \propto \sum_{i=1}^{N} I_i \quad \Rightarrow \quad A = \frac{1}{N}\sum_{i=1}^{N} I_i$$

3. Architectural Patterns for Reliable Deployment

Reliable deployment of machine learning (ML) models is crucial for cloud data platforms. Data is constantly changing, so monitoring production systems and detecting problems before they affect users are key requirements. Continuous integration and delivery (CI/CD) practices enable frequent delivery of new features or bug fixes with reduced risk [22]. In MLOps, CI refers to automatically validating modular parts in an ML pipeline when changes occur, while CD means automating the rollout of ML models into production while controlling for data quality and model performance [23].

For new ML models, the CI pipeline checks data correctness and performs unit tests on the ML pipeline, while the QA program conducts model validation. For new data, the CI pipeline executes data validation and ML pipeline tests, ensuring that model performance meets a pre-defined standard [24]. On successful validation, the CD pipeline rolls out the model using canary, blue-green, or feature-flag techniques to control traffic and fallbacks are ready if necessary [25].

Continuous integration and testing pipelines for models cover seven main aspects of the ML process: data quality validation; input feature space monitoring; unit testing of the model training module; integration testing of the full ML pipeline; model performance validation; accuracy monitoring; and data drift detection and alerting. Such procedures furnish assurance that newly deployed models behave as expected in production and prevent failures caused by data issues [26].

3.1. Continuous Integration and Testing for Models

Reliable deployment of models in a production environment requires validation at a single point of time and over time. A robust continuous integration (CI) pipeline for models supports reliable deployment when models are trained, retrained, or replaced [27]. CI for models involves validating and testing the code, models, and data systematically before deploying changes to real-time prediction serving platforms. An MLOps CI pipeline includes the following validation and testing processes: automated validation of data used for retraining; unit tests for code related to data preparation, retraining, and causing model rollout; integration tests for data preparation and model rollout components; testing criteria and guidelines; and automated quality assurance (QA) testing of models before promoting them to production [28].

MLOps practitioners deploy an additional CI test when a new model is introduced. A new model is QA-tested for accuracy on a data window through feature drift detection. Based on the expected model precision, the new model can be integrated with the existing model or become the primary model. Stability can be ensured by validating internal quality metrics that indicate the reliability of making predictions with each of the models that are made serviceable [27].

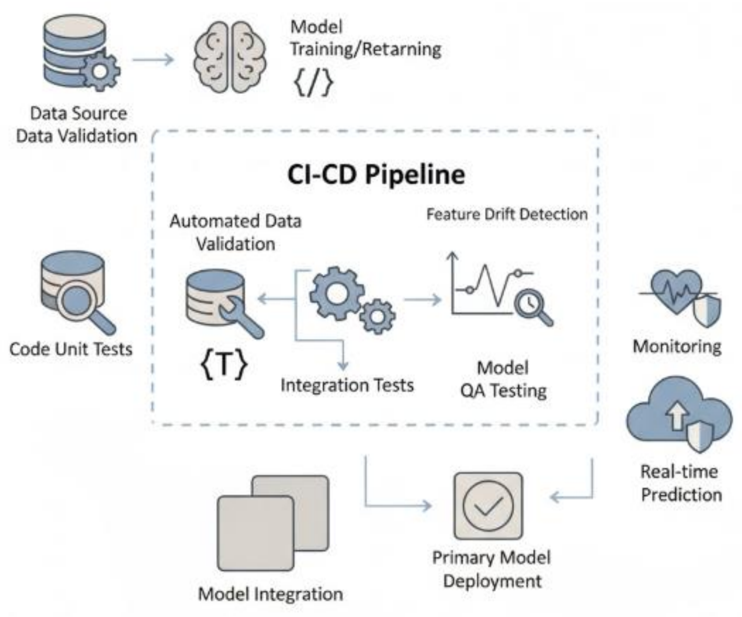

Figure 3. Longitudinal Integrity Frameworks: Continuous Integration and Drift-Resilient Deployment in Clinical MLOps

3.2. Continuous Delivery and Rollout Strategies

MLOps in Cloud Environments — Reliability Requirements

CI/CD pipelines enable teams to push changes to production quickly and safely. Continuous delivery consists of all modifications that enter a production-ready state and become available for release at any time. The actual deployments happen when these packages reach production, typically using automated rollout strategies [28].

Deployment of ML systems is considered reliable when it has the potential to succeed every time. Therefore, ML model deployment requires additional engineering and testing effort before changing production serving paths or the set of models used by another component [29]. One idea for a canary deployment is to route a small proportion of inference traffic through the new model in staging and production or to use the new model for selected user cohorts that do not affect business-critical functionality [30]. Model serving solutions that do not introduce latency hot paths can safely validate abilities, such as prediction quality or input shape, by running in a feature-flagged environment [31]. Model serving solutions that support ingesting incompatible schemas can enable multiple model versions in production while rolling out a new model to users. In the event of failure, trained engineering and operations teams can rapidly deactivate the flow. Additional visibility and health checks provide guidance on when to resume traffic on the new model [32].

Rollout strategies with more drastic fall-back options, such as blue-green or shadowing deployments, can be considered for models whose traffic is small enough to allow doubling resource allocation [33]. Traffic shadowing is the only strategy that is able to detect runtime issues related to user behavior, model execution environment (e.g., operating system and library versions, model artifacts), and interactions with all other downstream services [34].

Integrating ML models into a microservice architecture enables management systems to react to known issues in runtime performance and models to vote on upstream requests using a weighted voting mechanism [35]. Production services can use multiple model versions to bypass performance penalties while backfilling the predictions of demand-based or SLA-driven models in common situations [36]. Use of an identical model serving solution for both prediction and training serving reduces the testing effort needed to ensure that the two facilities provide the same contract and automates the deployment of new training model releases into prediction for shadowing, canary, and blue-green use cases [37].

4. Frameworks and Tooling Landscape

The landscape of frameworks and tools offering MLOps support can be divided into three areas: containerization and orchestration; model serving frameworks and inference engines; and experiment tracking tools [38].

Containerization revolutionized the cloud environment by providing an efficient way to run applications in an isolated environment. Despite being lightweight compared with virtual machines, containers must still be scheduled to run on a set of machines; scheduling becomes complex when applications require persistent storage and a large number of interdependencies in their network [39]. The typical solution is Kubernetes, which supports scheduling of arbitrary containers and provides cluster resource management and orchestration. Although Kubernetes does not natively support advanced scheduling for ML workloads, the ecosystem contains add-ons for specifying jobs and scheduling model training on available resources [40]. Specialized platforms such as Kubeflow, Vertex AI Pipelines, and AWS SageMaker Pipelines provide custom operators to support the full lifecycle of ML workloads using Kubernetes as the underlying orchestration layer [41].

Model serving frameworks facilitate the transition between the training and production stages of model lifecycle management. Although APIs are typically provided for external applications to request inferences for trained models, additional concerns must be addressed: throughput and latency optimizations, versioning of multiple models in the serving layer, automatic scaling, resource allocation during low-load periods, and smart batching of requests. Inference engines integrate seamlessly into the ML stack, understand the expressive model definitions of the chosen framework, and provide considerate wrappers for serving models [42]. Furthermore, several offerings focus specifically on the needs of ML inference: Nvidia Triton, TensorFlow Serving, Seldon, and KFServing. Experiment tracking tools automate and assist in recording information about experiments in order to compare trained models [43]. Considered parameters include metadata for datasets and trained models, metrics and logs, and artifacts such as models and visual representations of results. Prominent tools include MLflow, Weights & Biases, Data Version Control, and Git [44].

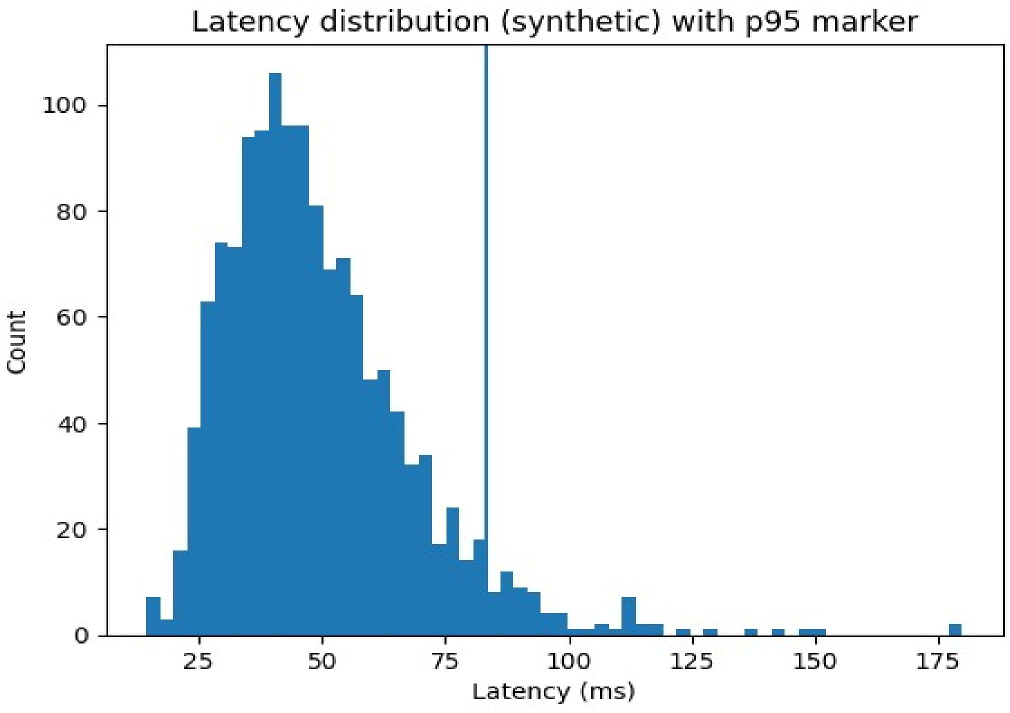

Figure 4. Response-Time Distribution Modeling for Reliability Monitoring in MLOps Platforms

Equation B: Latency (end-to-end response time)

Step 1 — per-request latency

For request j:

$$L_j = t_{\text{out}}^{(j)} - t_{\text{in}}^{(j)}$$

Step 2 — average latency over N requests

$$\bar{L} = \frac{1}{N}\sum_{j=1}^{N} L_j$$

Step 3 — percentile latency (common SLO form)

Sort latencies: $L_{(1)} \leq \cdots \leq L_{(N)}$.

The p-th percentile (e.g., $p = 95$) is approximately:

$$L_p \approx L_{(\lceil pN/100 \rceil)}$$

This is why dashboards often show p50/p95/p99.

Step 4 — alert rule form

If your SLO is “p95 latency ≤ L^*” then an alert condition is:

$$L_{95} > L^*$$

4.1. Containerization and Orchestration for ML workloads

Containerization and orchestration are widely adopted in cloud computing and have significantly influenced architecture patterns for deploying production applications. Resource isolation and independent lifecycle management of services address various concerns such as fault tolerance and technology diversity [45]. Schedulers and orchestration engines simplify service management, such as redundancy setup, load balancing, and failure recovery [46]. Although more common in back-end services, containerization is increasingly utilized for serving machine learning (ML) models because serving requests depends mainly on processing latency and unlike training, predicted features do not have to be saved for later retrieval. The heavy computational nature of large language models is leading to adoption by ML-specific scheduling, for both cost and latency reduction for end-users [47]. Parallel requests can be served simultaneously by different replicas, while model versioning is important for stepwise promotion and rollback. Distributed aspects also optimize resource consumption. Inference is memory bound, especially transformers [48].

Base-container image size is another factor affecting latency for both model serving and retraining. Optimizing workflow also improves container launch overhead and therefore response time in sudden inbound spikes. TorchScript, TensorFlow Saved Model, and ONNX accelerate model inference workflows [49]. TensorRT optimizes tensors for computing engine performance while DeepSpeed compresses model parameters in memory for free up usage for CPU resources. Model serving and container orchestration frameworks support both load and model-level autoscaling [50].

Table 2. Core Reliability Engineering Requirements for Cloud-Based MLOps Platforms

Bin	ref_p	cur_p
6	0.1	0.08783864900556244
7	0.1	0.094841730361359
8	0.1	0.10020408979951179
9	0.1	0.12479490976029453
10	0.1	0.21155308335667694

4.2. Model Serving Frameworks and Inference Engines

Containerization and orchestration of ML workloads have become common practice in production deployments. General-purpose technologies such as Docker and Kubernetes are extensively used for encapsulating run-time environments, instantiating and scaling deployments, and enabling reliability features such as health-checks and auto-restarts [51]. In addition, a range of ML-specific schedulers comes with built-in integrations for monitoring data preparation and inference pipelines [52]. By encapsulating all non-model-related code and resources in external services, complex orchestrators can be avoided. Such an approach allows for massive inference workloads to be processed, with good latency and throughput guarantees, by scaling the serving layer horizontally with dedicated auto-scaling groups [53]. In contrast to standard ML-serving solutions, the drawback of dedicated orchestration is that it requires a lot of preparation and provides limited quality-of-service guarantees [54].

Dedicated model-serving frameworks and inference engines represent the actual workhorses of ML workloads in production, exposed as services by clients, and are invoked by dedicated processes or common schedulers [55]. They aim to provide low-latency inference while maximizing throughput by caching hot models in memory and batching predictions. Take Latency and throughput targets into consideration when selecting these solutions. Orchestration platforms allow using SLO, SLI, and SLA concepts to trigger alerts and trigger actions such as scaling in or out the model-serving region, scaling up or down the machine where the region is deployed, or switching to a lower-refresh-frequency region model [56].

5. Cloud Platform-Specific Considerations

Public cloud providers offer a comprehensive portfolio of services aligned with the growing needs of Machine Learning (ML) projects. Dedicated Managed ML Services streamline the management of ML algorithms but often sacrifice critical aspects such as monitoring quality metrics, measuring drift detection in production data, implementing proper governance procedures, and managing operational Security [57]. Therefore, in practice, part of the infrastructure is usually hosted on self-managed dedicated VMs, Containers, or Kubernetes that allow in-depth control of the reliability and non-functional aspects and integrate with the Managed ML Services to fulfil the full-spectrum ML life cycle [58]. Strategies such as Multi-Cloud, Hybrid environments, and Customer-managed deployment of services delivered by Service Providers, often constrained by Legal aspects such as Data Sovereignty, Data Patterns and Portability concerns and Orchestration of the underlying Infrastructure and Services, emerge to address and mitigate these challenges [59].

Public cloud providers offer a varied portfolio of services aligned with the growing needs of Machine Learning (ML) projects. Dedicated Managed ML Services integrate cloud providers' strengths but introduce cost-related concerns and sacrifice critical aspects such as checking the quality of the produced models, monitoring the predictive/model quality metrics, drift detection of the production data, applying a proper governance process, managing Security on infrastructure Data Patterns and Partitions, non functional aspects such as Latency, Throughput and Availability [60]. It is therefore more common for organizations to self-manage part of the infrastructure using dedicated VMs, Containers, Kubernetes or dedicated for ML Orchestration that allow control and integration of the reliability and non-functional aspects of ML operational Phase while leveraging Managed services for the training, production and Service Users facing Interaction of the ML System [61]. Multi-Cloud, Hybrid and Customer-managed Deployment strategies of Services delivered by Service Providers gradually become standard practices to satisfy Business and Law oriented constraints like Data Sovereignty and Data Patterns [62].

5.1. Public Cloud Offerings and Managed ML Services

Public cloud providers distinguish their service offerings through high-level managed services. The Machine Learning as a Service (MLaaS) paradigm allows customers to efficiently deploy their ML workloads without worrying about the underlying infrastructure, as they are abstracted away by the cloud provider [63]. Many MLaaS services cover the entire ML lifecycle, but none of the major cloud providers offer a solution for all MLOps properties outlined in the previous sections. Some components are well covered in certain services of specific providers, while others are simplified or lacking [64]. The existing public offerings therefore call for additional tooling selection to achieve more reliable deployments and fulfill more MLOps properties [65].

The interaction of MLaaS with the remaining data pipelines must also be addressed, particularly when the data enter the cloud platform from on-premises infrastructure or when it is kept on a different cloud for compliance reasons [66]. For these situations, Data Transfer Services are available at every major provider to help speed up data movements between on-premise equipment or different cloud regions. Despite being tactical services, they also have a large impact on MLOps properties [67].

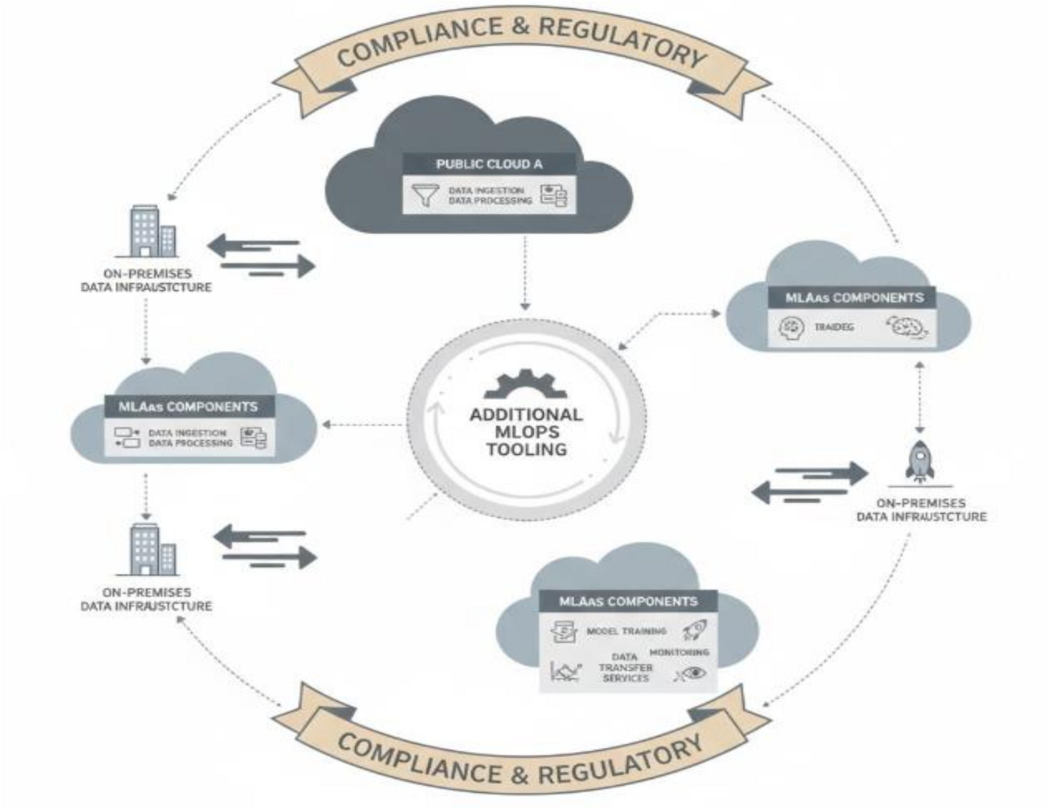

Figure 5. Bridging the MLaaS Gap: Integrated MLOps Orchestration and Tactical Data Transfer in Hybrid-Cloud Environments

5.2. *Multi-Cloud and Hybrid Deployments*

This section explores multi-cloud and hybrid cloud deployments, assessing their benefits and challenges for MLOps in cloud environments. The competitive advantage of public clouds stems from the ability to distribute workloads as-needed. Service offerings tailored for unique needs (e.g., Geolocation Services, Meta's Facebook) are located close to users, and edge resources managed by the cloud providers reduce latency [68]. Nevertheless, data sovereignty laws mandate data processing and storage within national borders. Therefore, controlling the ownership and location of data for data governance is crucial, enabling organizations not only to meet legal requirements but also to control costs and vendor lock-in. Multi-cloud deployments allow organizations to use different public clouds, promoting price competition among providers, while also alleviating the limitations and problems associated with a single cloud vendor [69].

The public cloud services may satisfy all the requirements for the deployment of an ML model. However, in several cases, the required resources address only a specific segment of the workload (e.g., Google’s part-in-place Tensor Processing Units, necessary only for the training phase of significant models) or are idle during periods of little or reduced demand [70]. MLops in the Cloud_My own version identifiers_full mark-up catering to complaints of reduced privacy risk and increased quality of service (QoS) in the scientific research process can take advantage of a hybrid deployment model. In this type of architecture, sensitive data can be kept in private premises, controlling and governing them like an on-premise scenario, while data without additional restrictions are considered in the public cloud [71]. Data portability between the private cloud and the public cloud is crucial to the effectiveness of hybrid deployment. Support for data portability narrows the scope of orchestration and enables the monitoring of sensitive workloads in normal production mode [72].

Table 3. Statistical Drift Detection Metrics for Monitoring Production ML Data Distributions

Drift test	Value
Population Stability Index (PSI)	0.12448176632338913
Kolmogorov–Smirnov statistic (KS)	0.1392
Kullback–Leibler divergence (KL)	0.06668422745742603

6. Reliability and Performance Metrics

Availability targets specify the maximum acceptable downtime for inference. For systems with very low (in the order of magnitude of milliseconds) and predictable latency (few percent are longer), the redundancy and maintenance costs are more significant and can trigger the system downtime [73]. For such systems, maintenance windows should be planned during low-usage times. Downtime windows can be scheduled for periods where the requested service is at its lowest demand. The only tool to measure such an aspect is an external ping or request to the model exposed in the serving layer. If the service respond with the latency defined previously, the request passes; otherwise, it fails [74].

Latency defines the maximum time for a request to be processed (both in one inference cycle and in the whole inference pipeline). It is also defined as the maximum number of time for a request to be processed before it goes dead. Performance dashboards should allow detection of anomalies with respect to these targets, allowing the selection of filters that support the identification of the causes of problems faster and more efficiently [75]. For such alerts, system request or model serving logs can record the inference latency from the moment they enter the model-exposed endpoint until the moment they go out. If the accumulated latency exceeds a certain value, an alert is triggered, indicating that an action should be taken soon (requests are becoming extinct). For performance, dashboards could be implemented to support anomaly detection with respect to the defined KPIs [76].

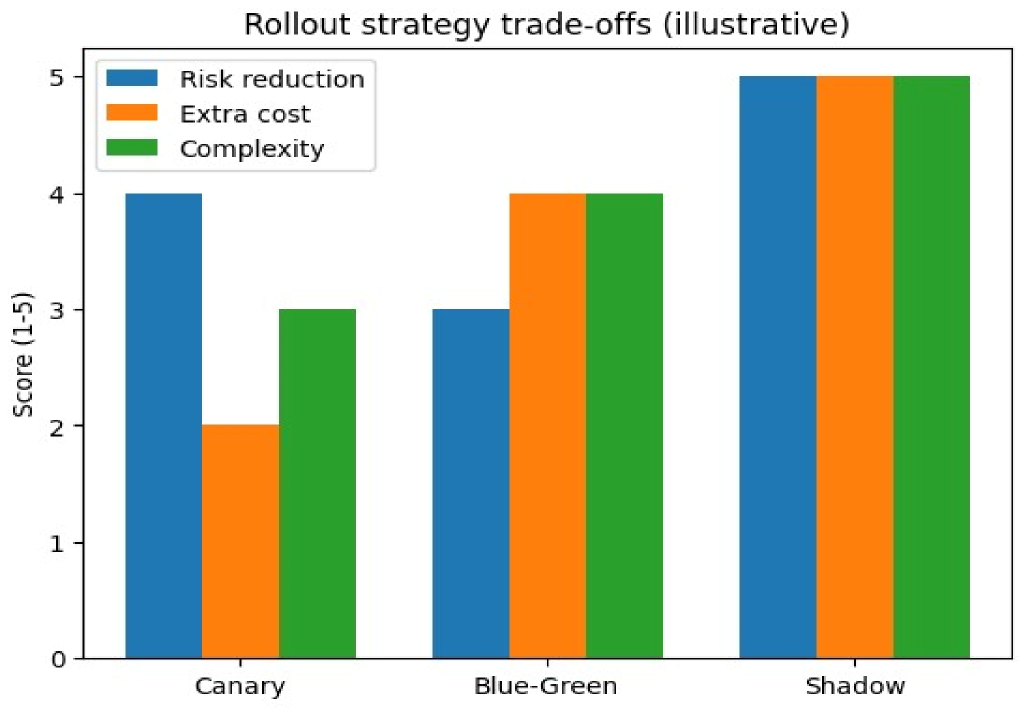

Figure 6. Aggregate Inference Throughput Formulation for Multi-Model Serving Architectures

Equation C: Throughput (inferences per unit time)

Step 1 — count requests

Let N requests be completed in time window Δt.

Step 2 — define throughput

$$X = \frac{N}{\Delta t}$$

Units: requests/second, requests/minute, etc.

Step 3 — aggregated throughput

If multiple models/services $k = 1..K$ run in parallel:

$$X_{\text{total}} = \sum_{k=1}^{K} X_k$$

6.1. Availability, Latency, and Throughput Targets

Availability, latency, and throughput constitute the primary reliability and performance metrics of deployed models. Availability is defined as the proportion of time the service is operational. Latency is the time from request submission to response retrieval and is constrained by user experience and business logic [77]. At the model level, throughput indicates the number of inferences serviced per time unit, and for aggregated deployments, it reflects the load from all models. These metrics guide reliability design choices, formulate SLOs, and allocate monitoring resources [78].

Availability is typically measured by uptime but may also be calculated for specific time windows. MLOps dashboards reflect these measurements, displaying ongoing availability levels, SLO status, and time spent in incidents. Latency SLOs are affected by user experience requirements, and businesses may set strict thresholds or classify latency as an implicit requirement. Throughput SLOs are based on request volume predictions and specified at the aggregate level or per traffic type [79].

6.2. Model Quality Metrics and Drift Detection

Any organization deploying models to production has requirements regarding model availability, prediction latency, and prediction throughput. These targets can typically be defined on a per-model basis, based on business requirements and Service Level Agreements (SLAs). Availability refers to the fraction of time that the route to the model is operational, such as considering underlying components like GPUs for on-demand allocation [80]. Latency is the total time taken for a single request to pass through all components of the inference pipeline, while throughput is the total number of requests processed in a given time window. These metrics can usually be computed, monitored, and displayed on dashboards using standard observability tooling [81].

In addition to these reliability and performance metrics, model quality is a major concern. The prediction quality of models may degrade over time as the statistical properties of the data change [82]. A number of different methods exist for detecting drift or changes in data distributions, such as population stability index (PSI) for feature distributions, Kolmogorov-Smirnov tests for comparing distributions, Kullback-Leibler divergence for comparing distributions, and Chi-square tests for categorical variables. Different methods can be employed for different types of prediction tasks, such as binary classification, multi-class classification, or regression, and threshold values can be defined above which alerts should be generated [83].

7. Conclusion

MLOps is key for applying ML to business problems. However, using ML safely at scale requires specialisation; reliable deployment is often overlooked. A pattern for reliable model deployment in cloud data platforms, organised around the themes of CI/CD pipelines for ML models, automation, monitoring, and incident response, is presented. The automation of CI pipelines and unit tests is crucial [84]. Data validation and model quality control during deployment are equally important; blue-green changes are preferred. Model-serving frameworks and inference engines should be carefully selected to meet reliability and performance targets. Managed services are convenient, but a cloud-agnostic approach is recommended [85].

MLOps is not a goal in itself. MLOps aims to allow the automated application of ML to business problems in production. Although the definition of MLOps is cloud-agnostic, most applications are focused on public clouds. The MLOps research and tooling ecosystem is mainly centred on the continuous integration and delivery of ML models [86]. However, this is hardly sufficient for safely scaling ML applications. This work proposes an architectural pattern specifically dedicated to reliable deployment and operation applied to data platforms in the public cloud [87]. The approach clarifies the roles of availability, latency, and throughput targets for model serving and inference at scale, and establishes the importance of reliable Model Quality Assurance and Incident Management pipelines and processes when deploying ML models into production [88].

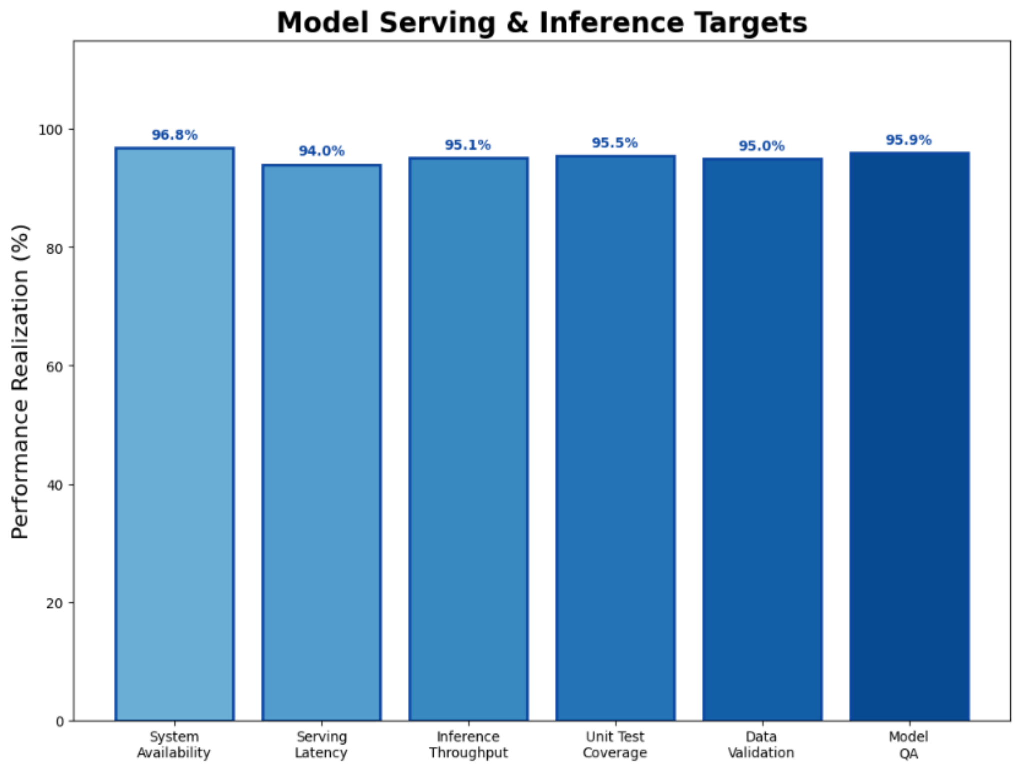

Figure 7. Model Serving & Inference Targets

7.1. Final Thoughts and Future Directions in MLOps

The final remarks reflect on the work, outline implications, discuss current limitations, and suggest directions for future research. A convergence of requirements to ensure the mutual cooperation between knowledge domain experts and DevOps teams is required. Quality assurance processes should be integrated within the CI/CD pipelines that run in cloud environments [90]. The investigation of practical MLOps toolsets and cloud data platform automations that improve the reliability has exposed an immature and scattered tool-market that requires further speeding up [91].

Research on the interaction between people and organizations has recently shown that specialized functional users need assistance for handling specialized tools. In ML systems built on data from the company financial market, knowledge domain experts have been detected unwilling to produce the required high-quality ML models. Data from these sources help strategic decisions taken by company leaders [92]. The organizational support for integrating the work of DevOps teams and the knowledge domain experts has been insufficient. As a consequence, CI/CD strategies for ML models are not used. Deployment is occurring manually by a user that has the knowledge in Python programming and MLOps processes [93].

Continuous integration and continuous delivery processes for ML models are being integrated into cloud environments. Nevertheless, the instability of the models remains an issue. Quality assurance processes that check the quality of models during the testing phase are necessary to further accelerate the maturity of the area [38]. Recent research on ML-based systems has pointed out the delay in the production of practical tools that implement the key concepts of MLOps and the use of public clouds in services offered to support the automation of MLOps tasks [94].

References

[1] Ahmed, M. S., & Cook, A. R. (1979). Analysis of freeway traffic time-series data by using Box-Jenkins techniques. Transportation Research Record, 722, 1–9.

[2] Gartner. (2019). ITIL 4: Create, deliver and support. TSO.

[3] Meda, R. (2023). Intelligent Infrastructure for Real-Time Inventory and Logistics in Retail Supply Chains. Educational Administration: Theory and Practice.

[4] Uday Surendra Yandamuri. (2022). Cloud-Based Data Integration Architectures for Scalable Enterprise Analytics. International Journal of Intelligent Systems and Applications in Engineering, 10(3s), 472–483. Retrieved from https://ijisae.org/index.php/IJISAE/article/view/8005.

[5] Goodfellow, I., Bengio, Y., & Courville, A. (2016). Deep learning. MIT Press.

[6] AI Powered Fraud Detection Systems: Enhancing Risk Assessment in the Insurance Sector. (2023). American Journal of Analytics and Artificial Intelligence (ajaai) With ISSN 3067-283X, 1(1). https://ajaai.com/index.php/ajaai/article/view/14.

[7] Abu-Elkheir, M., Hayajneh, M., & Ali, N. A. (2016). Data management for the Internet of Things: Design primitives and solutions. Sensors, 16(4), 454.

[8] Grieves, M., & Vickers, J. (2017). Digital twin: Mitigating unpredictable, undesirable emergent behavior in complex systems. In F.-J. Kahlen, S. Flumerfelt, & A. Alves (Eds.), Transdisciplinary perspectives on complex systems (pp. 85–113). Springer.

[9] Segireddy, A. R. (2020). Cloud Migration Strategies for High-Volume Financial Messaging Systems.

[10] Kummari, D. N., & Burugulla, J. K. R. (2023). Decision Support Systems for Government Auditing: The Role of AI in Ensuring Transparency and Compliance. International Journal of Finance (IJFIN)-ABDC Journal Quality List, 36(6), 493-532.

[11] Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. Neural Computation, 9(8), 1735–1780.

[12] Al-Turjman, F. (2020). Intelligence-enabled sustainable smart cities: An Internet of Things (IoT) perspective. Elsevier.

[13] Unifying Data Engineering and Machine Learning Pipelines: An Enterprise Roadmap to Automated Model Deployment. (2023). American Online Journal of Science and Engineering (AOJSE) (ISSN: 3067-1140), 1(1). https://aojse.com/index.php/aojse/article/view/19.

[14] International Organization for Standardization. (2018). ISO/IEC 27001:2018 Information security management systems—Requirements. ISO.

[15] Garapati, R. S. (2022). Web-Centric Cloud Framework for Real-Time Monitoring and Risk Prediction in Clinical Trials Using Machine Learning. Current Research in Public Health, 2, 1346.

[16] Avinash Reddy Aitha. (2022). Deep Neural Networks for Property Risk Prediction Leveraging Aerial and Satellite Imaging. International Journal of Communication Networks and Information Security (IJCNIS), 14(3), 1308–1318. Retrieved from https://www.ijcnis.org/index.php/ijcnis/article/view/8609.

[17] Jiang, X., Zhang, L., & Yang, Q. (2021). Edge computing for intelligent transportation systems: A survey. IEEE Internet of Things Journal, 8(19), 14185–14204.

[18] Gottimukkala, V. R. R. (2021). Digital Signal Processing Challenges in Financial Messaging Systems: Case Studies in High-Volume SWIFT Flows.

[19] Kaack, L. H., & Katzev, R. D. (1991). Behavioral intervention strategies for reducing traffic congestion. Transportation Research Part A: General, 25(1), 63–73.

[20] Arel, I., Liu, C., Urbanik, T., & Kohls, A. (2010). Reinforcement learning-based multi-agent system for network traffic signal control. IET Intelligent Transport Systems, 4(2), 128–135.

[21] Amistapuram, K. (2021). Digital Transformation in Insurance: Migrating Enterprise Policy Systems to .NET Core. Universal Journal of Computer Sciences and Communications, 1(1), 1–17. Retrieved from https://www.scipublications.com/journal/index.php/ujcsc/article/view/1348.

[22] Arthurs, P., Gillam, L., Krause, P., Wang, N., Halder, K., & Mouzakitis, A. (2022). A taxonomy and survey of edge cloud computing for intelligent transportation systems and connected vehicles. IEEE Transactions on Intelligent Transportation Systems, 23(7), 6206–6221.

[23] Karpukhin, V., Oguz, B., Min, S., Lewis, P., Wu, L., Edunov, S., Chen, D., & Yih, W.-T. (2020). Dense passage retrieval for open-domain question answering. In Proceedings of EMNLP 2020 (pp. 6769–6781). Association for Computational Linguistics.

[24] Meda, R. (2023). Developing AI-Powered Virtual Color Consultation Tools for Retail and Professional Customers. Journal for ReAttach Therapy and Developmental Diversities. https://doi. org/10.53555/jrtdd. v6i10s (2), 3577.

[25] Khattab, O., & Zaharia, M. (2020). ColBERT: Efficient and effective passage search via contextualized late interaction over BERT. In Proceedings of SIGIR 2020 (pp. 39–48). ACM.

[26] Segireddy, A. R. (2021). Containerization and Microservices in Payment Systems: A Study of Kubernetes and Docker in Financial Applications. Universal Journal of Business and Management, 1(1), 1–17. Retrieved from https://www.scipublications.com/journal/index.php/ujbm/article/view/1352.

[27] Kolla, S. H. (2021). Rule-Based Automation for IT Service Management Workflows. Online Journal of Engineering Sciences, 1(1), 1–14. Retrieved from https://www.scipublications.com/journal/index.php/ojes/article/view/1360

[28] Koonce, P., & Rodegerdts, L. (2008). Traffic signal timing manual. Transportation Research Board.

[29] Bazzan, A. L. C., & Klügl, F. (2014). Introduction to intelligent systems in traffic and transportation. Morgan & Claypool.

[30] Bellaïche, M., & Gruyer, D. (Eds.). (2013). Intelligent vehicles: Enabling technologies and future developments. Wiley.

[31] [58] Varri, D. B. S. (2023). Advanced Threat Intelligence Modeling for Proactive Cyber Defense Systems. Available at SSRN 5774926.

[32] LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. Nature, 521(7553), 436–444.

[33] Inala, R. Revolutionizing Customer Master Data in Insurance Technology Platforms: An AI and MDM Architecture Perspective

[34] Goutham Kumar Sheelam, Hara Krishna Reddy Koppolu. (2022). Data Engineering And Analytics For 5G-Driven Customer Experience In Telecom, Media, And Healthcare. Migration Letters, 19(S2), 1920–1944. Retrieved from https://migrationletters.com/index.php/ml/article/view/11938.

[35] Li, Y., Zheng, Y., Zhang, H., & Chen, L. (2015). Traffic prediction in a bike-sharing system. In Proceedings of SIGSPATIAL 2015 (pp. 1–10). ACM.

[36] Bergstra, J., & Bengio, Y. (2012). Random search for hyper-parameter optimization. Journal of Machine Learning Research, 13, 281–305.

[37] Uday Surendra Yandamuri. (2023). An Intelligent Analytics Framework Combining Big Data and Machine Learning for Business Forecasting. International Journal Of Finance, 36(6), 682-706. https://doi.org/10.5281/zenodo.18095256.

[38] Lopes, L., Bento, C., & Baptista, P. (2021). Environmental impacts of congestion pricing: A review of methods and evidence. Transport Reviews, 41(5), 623–647.

[39] Guntupalli, R. (2023). Optimizing Cloud Infrastructure Performance Using AI: Intelligent Resource Allocation and Predictive Maintenance. Available at SSRN 5329154.

[40] Kalisetty, S. (2023). Harnessing Big Data and Deep Learning for Real-Time Demand Forecasting in Retail: A Scalable AI-Driven Approach. American Online Journal of Science and Engineering (AOJSE)(ISSN: 3067-1140), 1(1).

[41] Lv, Y., Duan, Y., Kang, W., Li, Z., & Wang, F.-Y. (2015). Traffic flow prediction with big data: A deep learning approach. IEEE Transactions on Intelligent Transportation Systems, 16(2), 865–873.

[42] Ma, X., Tao, Z., Wang, Y., Yu, H., & Wang, Y. (2015). Long short-term memory neural network for traffic speed prediction using remote microwave sensor data. Transportation Research Part C: Emerging Technologies, 54, 187–197.

[43] Bonnefon, J.-F., Shariff, A., & Rahwan, I. (2016). The social dilemma of autonomous vehicles. Science, 352(6293), 1573–1576.

[44] Kummari, D. N. (2023). AI-Powered Demand Forecasting for Automotive Components: A Multi-Supplier Data Fusion Approach. European Advanced Journal for Emerging Technologies (EAJET)-p-ISSN 3050-9734 en e-ISSN 3050-9742, 1(1).

[45] Inala, R. AI-Powered Investment Decision Support Systems: Building Smart Data Products with Embedded Governance Controls.

[46] NIST. (2020). Security and privacy controls for information systems and organizations (NIST SP 800-53 Rev. 5). U.S. Department of Commerce.

[47] Brin, S., & Page, L. (1998). The anatomy of a large-scale hypertextual web search engine. Computer Networks and ISDN Systems, 30(1–7), 107–117.

[48] Rongali, S. K. (2021). Cloud-Native API-Led Integration Using MuleSoft and .NET for Scalable Healthcare Interoperability. Available at SSRN 5814563.

[49] Object Management Group. (2016). Business process model and notation (BPMN), version 2.0.2. OMG.

[50] Meda, R. (2023). Data Engineering Architectures for Scalable AI in Paint Manufacturing Operations. European Data Science Journal (EDSJ) p-ISSN 3050-9572 en e-ISSN 3050-9580, 1(1).

[51] Burns, B., Beda, J., & Hightower, K. (2019). Kubernetes: Up & running (2nd ed.). O'Reilly Media.

[52] Pahl, C., Brogi, A., Soldani, J., & Jamshidi, P. (2019). Cloud container technologies: A state-of-the-art review. IEEE Transactions on Cloud Computing, 7(3), 677–692.

[53] Kushvanth Chowdary Nagabhyru. (2023). Accelerating Digital Transformation with AI Driven Data Engineering: Industry Case Studies from Cloud and IoT Domains. Educational Administration: Theory and Practice, 29(4), 5898–5910. https://doi.org/10.53555/kuey.v29i4.10932.

[54] Garapati, R. S. (2022). AI-Augmented Virtual Health Assistant: A Web-Based Solution for Personalized Medication Management and Patient Engagement. Available at SSRN 5639650..

[55] Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., & Chintala, S. (2019). PyTorch: An imperative style, high-performance deep learning library. In Proceedings of NeurIPS 2019 (pp. 8024–8035).

[56] Nagabhyru, K. C. (2023). From Data Silos to Knowledge Graphs: Architecting CrossEnterprise AI Solutions for Scalability and Trust. Available at SSRN 5697663.

[57] Polson, N. G., & Sokolov, V. O. (2017). Deep learning for short-term traffic flow prediction. Transportation Research Part C: Emerging Technologies, 79, 1–17.

[58] Castro-Neto, M., Jeong, Y.-S., Jeong, M.-K., & Han, L. D. (2009). Online-SVR for short-term traffic flow prediction under typical and atypical traffic conditions. Expert Systems with Applications, 36(3), 6164–6173.

[59] Gottimukkala, V. R. R. (2022). Licensing Innovation in the Financial Messaging Ecosystem: Business Models and Global Compliance Impact. International Journal of Scientific Research and Modern Technology, 1(12), 177-186.

[60] Rahman, M., & Liu, H. X. (2021). A review of connected and automated vehicle impacts on traffic flow and control. Transportation Research Part C: Emerging Technologies, 124, 102939.

[61] Aitha, A. R. (2023). CloudBased Microservices Architecture for Seamless Insurance Policy Administration. International Journal of Finance (IJFIN)-ABDC Journal Quality List, 36(6), 607-632.

[62] Garapati, R. S. (2023). Optimizing Energy Consumption in Smart Build-ings Through Web-Integrated AI and Cloud-Driven Control Systems.

[63] Chen, X., Zahiri, M., & Zhang, S. (2017). Understanding ridesplitting behavior of on-demand ride services: An ensemble learning approach. Transportation Research Part C: Emerging Technologies, 76, 51–70.

[64] Cheng, W.-H., Kuo, C.-Y., Lin, W.-C., & Hsu, C.-T. (2018). Real-time traffic incident detection based on video analytics: A survey. ACM Computing Surveys, 50(3), 1–35.

[65] Shaheen, S., & Cohen, A. (2020). Shared mobility policy playbook. Institute of Transportation Studies, UC Berkeley.

[66] Gottimukkala, V. R. R. (2023). Privacy-Preserving Machine Learning Models for Transaction Monitoring in Global Banking Networks. International Journal of Finance (IJFIN)-ABDC Journal Quality List, 36(6), 633-652.

[67] Varri, D. B. S. (2022). AI-Driven Risk Assessment And Compliance Automation In Multi-Cloud Environments. Available at SSRN 5774924.

[68] Codd, E. F. (1970). A relational model of data for large shared data banks. Communications of the ACM, 13(6), 377–387.

[69] Sutton, R. S., & Barto, A. G. (2018). Reinforcement learning: An introduction (2nd ed.). MIT Press.

[70] Avinash Reddy Segireddy. (2022). Terraform and Ansible in Building Resilient Cloud-Native Payment Architectures. Keerthi Amistapuram. (2023). Privacy-Preserving Machine Learning Models for Sensitive Customer Data in Insurance Systems. Educational Administration: Theory and Practice, 29(4), 5950–5958. https://doi.org/10.53555/kuey.v29i4.10965.

[71] Daganzo, C. F. (1994). The cell transmission model: A dynamic representation of highway traffic consistent with the hydrodynamic theory. Transportation Research Part B: Methodological, 28(4), 269–287.

[72] Nandan, B. P., & Chitta, S. S. (2023). Machine Learning Driven Metrology and Defect Detection in Extreme Ultraviolet (EUV) Lithography: A Paradigm Shift in Semiconductor Manufacturing. Educational Administration: Theory and Practice, 29 (4), 4555–4568.

[73] Rongali, S. K. (2022). AI-Driven Automation in Healthcare Claims and EHR Processing Using MuleSoft and Machine Learning Pipelines. Available at SSRN 5763022.

[74] Ta, N., Liu, H., & Chen, M. (2020). Advanced travel demand management in smart cities: A survey. IEEE Access, 8, 152126–152145.

[75] Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2019). BERT: Pre-training of deep bidirectional transformers for language understanding. In Proceedings of NAACL-HLT 2019 (pp. 4171–4186). Association for Computational Linguistics.

[76] Rongali, S. K. (2023). Explainable Artificial Intelligence (XAI) Framework for Transparent Clinical Decision Support Systems. International Journal of Medical Toxicology and Legal Medicine, 26(3), 22-31.

[77] Guntupalli, R. (2023). AI-Driven Threat Detection and Mitigation in Cloud Infrastructure: Enhancing Security through Machine Learning and Anomaly Detection. Available at SSRN 5329158.

[78] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., & Polosukhin, I. (2017). Attention is all you need. In Proceedings of NeurIPS 2017 (pp. 5998–6008).

[79] Challa, K. Dynamic Neural Network Architectures for Real-Time Fraud Detection in Digital Payment Systems Using Machine Learning and Generative AI.

[80] Ding, W., & Rossiter, J. A. (2016). Model predictive control: A review and future directions. In 2016 IEEE 55th Conference on Decision and Control (CDC) (pp. 4149–4154). IEEE.

[81] Yu, B., Yin, H., & Zhu, Z. (2018). Spatio-temporal graph convolutional networks: A deep learning framework for traffic forecasting. In Proceedings of IJCAI 2018 (pp. 3634–3640).

[82] Yandamuri, U. S. (2022). Big Data Pipelines for Cross-Domain Decision Support: A Cloud-Centric Approach. International Journal of Scientific Research and Modern Technology, 1(12), 227–237. https://doi.org/10.38124/ijsrmt.v1i12.1111.

[83] Kiran Reddy Burugulla, J. (2023). Transforming Payment Systems Through AI And ML: A Cloud-Native Approach. Educational Administration: Theory and Practice. https://doi.org/10.53555/kuey.v29i4.10144.

[84] Dwork, C., & Roth, A. (2014). The algorithmic foundations of differential privacy. Foundations and Trends in Theoretical Computer Science, 9(3–4), 211–407.

[85] Zheng, Y., Capra, L., Wolfson, O., & Yang, H. (2014). Urban computing: Concepts, methodologies, and applications. ACM Transactions on Intelligent Systems and Technology, 5(3), 1–55.

[86] Varri, D. B. S. (2022). A Framework for Cloud-Integrated Database Hardening in Hybrid AWS-Azure Environments: Security Posture Automation Through Wiz-Driven Insights. International Journal of Scientific Research and Modern Technology, 1(12), 216-226.

[87] European Commission. (2019). The ethics guidelines for trustworthy AI. Publications Office of the European Union.

[88] Amistapuram, K. (2022). Fraud Detection and Risk Modeling in Insurance: Early Adoption of Machine Learning in Claims Processing. Available at SSRN 5741982

[89] Kummari, D. N. (2023). Energy Consumption Optimization in Smart Factories Using AI-Based Analytics: Evidence from Automotive Plants. Journal for Reattach Therapy and Development Diversities. https://doi. org/10.53555/jrtdd. v6i10s (2), 3572.

[90] Caruana, R., et al. (2015). Intelligible models for healthcare. Proceedings of ACM SIGKDD, 1721–1730.

[91] Farah, H., Bekhor, S., & Polus, A. (2014). Risk evaluation by modeling of passing behavior on two-lane rural highways. Accident Analysis & Prevention, 70, 109–118.

[92] Siva Hemanth Kolla. (2023). Deep Learning–Driven Retrieval-Augmented Generation for Enterprise ITSM Automation: A Governance-Aligned Large Language Model Architecture. Journal of Computational Analysis and Applications (JoCAAA), 31(4), 2489–2502. Retrieved from https://www.eudoxuspress.com/index.php/pub/article/view/4774

[93] Ramesh Inala. (2023). Big Data Architectures for Modernizing Customer Master Systems in Group Insurance and Retirement Planning. Educational Administration: Theory and Practice, 29(4), 5493–5505. https://doi.org/10.53555/kuey.v29i4.10424.

[94] Bholat, D., Gharbawi, M., & Thew, O. (2023). Machine learning, big data, and system observability. Financial Stability Review, 27, 33–49.