

Integration of Machine Learning and Big Data for Intelligent Cloud-Based Decision Support

George Emmanuel

Independent Researcher

Midwestern, United States

Email ID: George_emmohio3102@yahoo.com

Abstract

Organizations have invested heavily in big data platforms and machine learning (ML), yet many ML models never reach production, and many that do fail to improve decisions. The problem is rarely the algorithm alone. It lies in how data, models, and decisions are connected over time: data pipelines that silently change, models that drift, predictions that stop short of recommending action, and outputs that never enter the processes where decisions are made. This article proposes a closed-loop framework for integrating ML and big data into intelligent cloud-based decision support. The framework covers a hybrid cloud data foundation for sensitive data, validated data and feature pipelines, managed model development and serving, a decision layer that moves from prediction to prescription, a retrieval-augmented natural-language decision assistant, workflow-driven decision execution, and monitoring and autonomous operations that feed outcomes back into learning. The article also defines four levels of decision support, from descriptive to prescriptive, and a production-readiness checklist for decision models. Using a design-oriented approach grounded in ten studies published between 2017 and 2025, it illustrates the framework with hospital capacity planning. It argues that intelligent decision support depends on treating data, models, and decisions as one continuously operated loop rather than as separate projects.

Keywords: machine learning, big data, decision support systems, cloud computing, MLOps, prescriptive analytics, concept drift, hybrid cloud, retrieval-augmented generation, autonomous operations, enterprise workflow

1. Introduction

Decision support systems aim to help people make better decisions by combining data, models, and human judgment. Big data and ML have greatly expanded what such systems can do. Organizations can now learn from years of detailed operational records and predict demand, risk, and failure with accuracy that was previously out of reach. Cloud computing provides the storage and computing capacity to do this at scale.

Many organizations nonetheless find that ML projects deliver less than expected. Paleyes et al. (2022) surveyed published case studies of ML deployment and found challenges at every stage: data management, model learning, verification, deployment, and ongoing operation. Many of these challenges had little to do with algorithms and much to do with integration, monitoring, and organizational processes. Kreuzberger et al. (2023) describe machine learning operations (MLOps) as the set of practices, roles, and architecture needed to move ML from experiments to reliable production systems. They note that the field still lacks a shared understanding of how to do this.

Decision support adds a further requirement. A prediction does not tell a decision-maker what to do. Bertsimas and Kallus (2020) show how to move from predictive to prescriptive analytics by using ML to inform optimization directly, producing recommended decisions rather than forecasts alone. Even a good recommendation must reach the process where the decision is made, which requires integration with enterprise workflows (Basireddy, 2022). Decision-makers also increasingly expect to ask questions in plain language, which generative AI and retrieval-augmented generation (RAG) now make possible (Leveraging Generative AI and RAG, 2023).

This article asks: How can ML and big data be integrated into cloud-based decision support so that it remains accurate, actionable, and reliable over time? It contributes a closed-loop integration framework, a four-level model of decision support, a production-readiness checklist for decision models, and an illustrative application in hospital capacity planning.

2. Related Work

2.1 From prediction to prescription

Bertsimas and Kallus (2020) propose methods that use ML models such as nearest neighbors, decision trees, and random forests to weight historical observations when solving decision

problems under uncertainty. They show that this approach produces better decisions than first predicting a quantity and then optimizing as if the prediction were certain. For decision support, the lesson is that the goal is a good decision, not only an accurate forecast, and the analytics should be designed around that goal.

2.2 Data in production ML

Polyzotis et al. (2018) survey the data lifecycle challenges of production ML. These include understanding and validating data, cleaning it, and preparing it for training and serving. They highlight that data errors, schema changes, and differences between training and serving data are major causes of model failure. Such failures are often silent: the model keeps producing outputs, but the outputs are wrong. Big data pipelines, with many sources and transformations, are especially exposed to these problems.

2.3 Hybrid cloud storage for sensitive data

Decision support in sectors such as healthcare and finance depends on sensitive data. The study on A Hybrid Cloud Storage Architecture for Large-Scale Medical Data (2017) proposes combining private infrastructure for sensitive records with public cloud resources for scalable storage and processing. This balances security and compliance against capacity and cost. For ML-driven decision support, the architecture lets identifiable data remain under strict control while de-identified or aggregated data is used for large-scale model training in the public cloud.

2.4 MLOps and production readiness

Kreuzberger et al. (2023) define MLOps through its principles, including automation, versioning, continuous training, monitoring, and reproducibility. They also define its components, such as pipelines, feature stores, model registries, and serving, and the roles involved. Breck et al. (2017) proposed the ML Test Score, a rubric of tests and monitoring practices across data, model development, infrastructure, and production monitoring. It measures whether an ML system is ready for production and helps reduce technical debt. Together, these works provide a vocabulary and a checklist for operating decision models responsibly.

2.5 Concept drift

Decision environments change. Lu et al. (2019) review learning under concept drift, meaning changes over time in the statistical relationship between inputs and outputs. They cover drift detection, understanding when, how, and where drift occurs, and adaptation strategies. A decision model trained on past conditions may give poor recommendations once conditions change, so drift detection and response are essential parts of decision support.

2.6 Generative AI and retrieval

The study on Leveraging Generative Artificial Intelligence and Retrieval-Augmented Generation to Improve Software Engineering, Automated Code Generation, and Intelligent Debugging Systems (2023) shows how generative models grounded in retrieved project knowledge can generate and debug code. Two uses follow for decision support. First, a RAG-based assistant can answer decision-makers' questions by retrieving model outputs, data summaries, and policies, and explaining them in plain language. Second, the same techniques can help data teams write and debug the data pipelines and analysis code on which decision models depend.

2.7 Autonomous operations

Sannidhanam (2025) examines autonomous AI agents for cloud infrastructure operations. These agents monitor systems, diagnose issues, and carry out corrective actions within defined limits under human oversight. For ML decision support, agents can watch pipeline health, data quality, and model performance, restart failed jobs, and trigger retraining within approved boundaries.

2.8 Workflow-driven decision execution

Basireddy (2022) proposes configurable enterprise workflow architecture for digitizing risk management, in which assessment, approval, escalation, and remediation are defined as configurable workflows with audit trails. In decision support, such workflows carry recommendations to the people who must approve them, record decisions and overrides, and provide the outcome data that closes the learning loop.

2.9 Research gap

These strands have developed separately. What is missing is an integrated view of how they

form a continuously operating decision support loop in the cloud.

3. Research Methodology

The study follows a design-oriented research approach. The problem, that ML and big data investments often fail to improve decisions, was identified from research on ML deployment. Requirements were derived from the reviewed studies: protected and scalable data, validated pipelines, operated models, decision-oriented analytics, accessible explanation, integration with decision processes, and adaptation to change. A closed-loop framework, a decision support level model, and a readiness checklist were designed from ten studies and demonstrated through an illustrative hospital case. The demonstration shows feasibility and internal consistency. Empirical evaluation is left to future work.

4. Closed-Loop Integration Framework

The framework connects eight components in a loop. Decisions and their outcomes flow back to improve data and models (Table 1).

Table 1. Closed-loop framework for ML and big data decision support in the cloud

Component	Role	Key practices	Supporting studies
1. Hybrid data foundation	Store sensitive and large-scale data appropriately	Private storage for identifiable data, public cloud for de-identified analytics	<i>Hybrid Cloud Storage Architecture</i> (2017)
2. Data and feature pipelines	Prepare consistent, validated inputs	Schema and distribution checks, shared features for training and serving	Polyzotis et al. (2018)
3. Model development and serving	Train, version, and deploy models	Pipelines, model registry, continuous training	Kreuzberger et al. (2023)
4. Decision layer	Turn predictions into recommended actions	Prescriptive methods, constraints, scenario comparison	Bertsimas & Kallus (2020)
5. Decision assistant	Explain and answer questions in plain language	RAG over model outputs, data summaries, and policies	<i>Leveraging Generative AI and RAG</i> (2023)
6. Decision execution	Route recommendations for approval and action	Configurable workflows, override capture, audit	Basireddy (2022)
7. Monitoring and operations	Detect failures and drift, keep the loop running	Data, model, and drift monitoring, autonomous agents	Lu et al. (2019); Sannidhanam (2025)
8. Feedback	Learn from decisions and	Outcome capture, retraining	Kreuzberger et al. (2023);

Hybrid data foundation. Sensitive, identifiable records stay in private infrastructure. De-identified and aggregated data needed for large-scale training and analysis moves to public cloud storage (Hybrid Cloud Storage Architecture, 2017).

Data and feature pipelines. Every pipeline validates incoming data against expected schemas and distributions and blocks or flags anomalies before they reach models (Polyzotis et al., 2018). Features are computed once and shared between training and serving, which prevents differences between the data a model learned from and the data it sees in use.

Model development and serving. Models are trained through automated, versioned pipelines, registered with their data and parameters, and deployed through managed serving (Kreuzberger et al., 2023). Continuous training pipelines allow retraining when triggered.

Decision layer. Rather than passing forecasts to decision-makers to interpret, the decision layer uses them to recommend actions subject to constraints such as budgets, capacities, and rules (Bertsimas & Kallus, 2020). It presents alternatives and their expected consequences so that decision-makers can compare scenarios.

Decision assistant. A RAG-based assistant lets decision-makers ask questions such as "Why is the recommendation higher than last week?" or "What happens if we add one more team?" It retrieves the relevant model outputs, data summaries, and policies and answers with citations (Leveraging Generative AI and RAG, 2023). The same retrieval-grounded generation helps data teams write and debug pipeline code.

Decision execution. Recommendations enter configurable workflows that route them to the people with authority to approve, modify, or reject them. The workflows record each decision and the reasons for overrides (Basireddy, 2022).

Monitoring and operations. Monitoring covers data quality, model accuracy, and drift (Lu et al., 2019). Autonomous agents restart failed pipelines, scale services, and trigger retraining when drift exceeds agreed thresholds. They escalate issues beyond their authority to people (Sannidhanam, 2025).

Feedback. Actual outcomes, together with the decisions taken and overrides recorded in workflows, flow back into the data foundation. Overrides are particularly valuable, because they show where human judgment disagreed with the model and why.

5. Levels of Decision Support

The framework supports four levels of decision support, each answering a different question (Table 2).

Table 2. Levels of ML and big data decision support

Level	Question	ML and big data techniques	Output	Human role
Descriptive	What happened?	Large-scale aggregation, dashboards	Summaries and trends	Interpret and investigate
Diagnostic	Why did it happen?	Drill-down, attribution, anomaly detection	Explanations of causes	Confirm causes
Predictive	What will happen?	Supervised learning, time series forecasting	Forecasts with uncertainty	Judge implications
Prescriptive	What should we do?	ML-informed optimization under constraints	Recommended actions and alternatives	Approve, adjust, or reject

Organizations usually progress through these levels. Moving to the prescriptive level brings the most value and also the most responsibility. The framework therefore places prescriptive outputs within approval workflows, so that people keep decision authority (Basireddy, 2022; Bertsimas & Kallus, 2020).

6. Production Readiness for Decision Models

Building on the ML Test Score (Breck et al., 2017) and MLOps principles (Kreuzberger et al., 2023), Table 3 sets out a readiness checklist for decision models.

Table 3. Production-readiness checklist for ML decision models

Area	Readiness checks	Supporting studies
Data	Schemas validated, input distributions monitored, training and serving features consistent	Polyzotis et al. (2018); Breck et al. (2017)
Model	Performance tested against a simple baseline and across important subgroups, reproducible from versioned data and code	Breck et al. (2017); Kreuzberger et al. (2023)
Decision	Recommendations respect constraints, alternatives shown, decision quality measured and not only prediction	Bertsimas & Kallus (2020)



	accuracy	
Infrastructure	Pipelines automated, rollback possible, sensitive data kept in the appropriate environment	Kreuzberger et al. (2023); <i>Hybrid Cloud Storage Architecture</i> (2017)
Monitoring	Drift detected with defined response, agent actions limited and logged	Lu et al. (2019); Sannidhanam (2025)
Process	Recommendations routed through workflows, overrides recorded, outcomes fed back	Basireddy (2022)

A decision model that fails any area should not be relied on for consequential decisions. The checklist links the technical readiness emphasized in MLOps with the decision-level readiness needed for decision support.

7. Illustrative Case: Hospital Capacity Planning

Consider a hospital group that must decide each day how many beds to open and how many nurses to schedule across several wards.

Data foundation. Patient-level records, including admissions, diagnoses, and lengths of stay, remain in the hospital's private cloud. De-identified daily counts, together with external data such as seasonal illness indicators and local events, move to the public cloud for model training (Hybrid Cloud Storage Architecture, 2017).

Pipelines and models. Daily pipelines validate the incoming data (Polyzotis et al., 2018). One morning the pipeline flags that emergency admissions from one site are missing because a feed failed, and it holds that site's forecast rather than treating the gap as zero admissions. Forecasting models, trained and versioned through the MLOps pipeline (Kreuzberger et al., 2023), predict admissions and discharges for each ward over the next seven days, with uncertainty ranges.

Prescriptive recommendations. The decision layer uses the forecasts and their uncertainty to recommend bed openings and nurse staffing for each ward (Bertsimas & Kallus, 2020). It respects constraints such as minimum nurse-to-patient ratios, available staff, and budget. It shows two alternatives: a lower-cost plan with a higher risk of overcrowding, and a more cautious plan.

Decision assistant. The operations manager asks the assistant why the recommended staffing for the respiratory ward is higher than last week. The assistant retrieves the forecast, the respiratory illness indicator, and the staffing policy. It explains that rising regional respiratory

cases and a higher forecast for admissions from the emergency department drive the increase, and it cites each source (Leveraging Generative AI and RAG, 2023).

Decision execution. The recommendation enters the staffing approval workflow (Basireddy, 2022). The manager approves the cautious plan for the respiratory ward and the lower-cost plan elsewhere, and records the reason. The approved plan is sent to the scheduling system.

Monitoring and adaptation. Over the following weeks, a new outpatient program reduces admissions for one condition. Drift monitoring detects that forecasts for the affected ward have become consistently too high (Lu et al., 2019). The operations agent triggers retraining within its approved limits (Sannidhanam, 2025), and the retrained model is validated before deployment.

Feedback. Actual admissions, the decisions taken, and the manager's overrides flow back into the data foundation. They improve both forecasts and understanding of when managers prefer caution.

The case shows the loop in operation. Protected data feeds validated pipelines and models, predictions become constrained recommendations, and recommendations are explained and executed through governed workflows. The system adapts as conditions change.

8. Discussion

8.1 Implications

For research, the framework connects work on prescriptive analytics, production ML, drift, generative assistants, autonomous operations, and workflows around the single goal of better decisions over time. It shows that decision support quality depends on the whole loop, not only model accuracy. The readiness checklist extends production-ML practice with decision-level criteria.

For practitioners, the priorities are to protect sensitive data through hybrid architecture, validate data before it reaches models, aim for recommendations rather than forecasts alone, and operate the system as a continuous service rather than a one-time project.

8.2 Limitations and future work

The framework was demonstrated through an illustrative case and draws on a focused set of ten studies. Future work should implement and evaluate it, measuring decision quality, time

from data to decision, frequency and causes of overrides, and recovery from drift. Further research is needed on measuring decision quality rather than prediction accuracy, on learning from human overrides, on setting safe limits for autonomous retraining, and on evaluating the accuracy of generative decision assistants.

9. Conclusion

ML and big data can transform decision support only when they are integrated into a loop that runs continuously. This article proposed a closed-loop framework linking a hybrid cloud data foundation, validated pipelines, operated models, prescriptive decision logic, a retrieval-grounded decision assistant, workflow-driven execution, and monitoring with autonomous operations and feedback. A four-level model and a readiness checklist show how organizations can advance decision support responsibly. The hospital case showed the loop turning sensitive data into explained, approved, and adaptive decisions. The central conclusion is that intelligent decision support comes from continuously connecting data, models, decisions, and outcomes, not from any single model.

References

1. Basireddy, S. R. (2022). Digitizing risk management through configurable enterprise workflow architecture. *SAMRIDDHI: A Journal of Physical Sciences, Engineering and Technology*, 14(4), 231–239.
2. Bertsimas, D., & Kallus, N. (2020). From predictive to prescriptive analytics. *Management Science*, 66(3), 1025–1044.
3. Breck, E., Cai, S., Nielsen, E., Salib, M., & Sculley, D. (2017). The ML test score: A rubric for ML production readiness and technical debt reduction. In *Proceedings of the 2017 IEEE International Conference on Big Data* (pp. 1123–1132).
4. Sannidhanam, A.H., Alladi, S. (2017). A Hybrid Cloud Storage Architecture for Large-Scale Medical Data (Anjani Haritha Sannidhanam & Shivani Alladi, Trans.). *International Journal of Food Safety and Public Health*, 4(1), 01-09.
<https://ijpp.org/journal/index.php/ajb/article/view/585>
5. Kreuzberger, D., Kühl, N., & Hirschl, S. (2023). Machine learning operations (MLOps): Overview, definition, and architecture. *IEEE Access*, 11, 31866–31879.



6. Cherukuri, M. (2023). Leveraging Generative Artificial Intelligence and Retrieval-Augmented Generation to Improve Software Engineering, Automated Code Generation, and Intelligent Debugging Systems. *International Journal of Humanities and Information Technology*, 5(04), 140-163. <https://doi.org/10.21590/ijhit.05.04.11>
7. Lu, J., Liu, A., Dong, F., Gu, F., Gama, J., & Zhang, G. (2019). Learning under concept drift: A review. *IEEE Transactions on Knowledge and Data Engineering*, 31(12), 2346–2363.
8. Paleyes, A., Urma, R.-G., & Lawrence, N. D. (2022). Challenges in deploying machine learning: A survey of case studies. *ACM Computing Surveys*, 55(6), Article 114.
9. Polyzotis, N., Roy, S., Whang, S. E., & Zinkevich, M. (2018). Data lifecycle challenges in production machine learning: A survey. *ACM SIGMOD Record*, 47(2), 17–28.
10. Sannidhanam, A. H. (2025). Autonomous AI agents for cloud infrastructure operations. *Journal of Contemporary Science and Technology Management*, 1(01), 83–100. <https://jcstm.info/index.php/jcstm/article/view/20>