JOURNAL OF INTERNATIONAL CRISIS AND RISK COMMUNICATION RESEARCH
ISSN: 2576-0017
2026, VOL 9, NO 1

From One Model To Many: Managing Dozens Of ML Models In The Enterprise

Swati Kumari

Nucleus Teq, USA

Abstract
Today, businesses depend on numerous models that are built and deployed using machine learning algorithms in their operations across various departments; however, managing these models in a scalable manner introduces a host of issues related to consistency, governance, and efficiency in operations. By failing to implement a unified strategy in these environments, a business incurs diminished benefits from their machine learning applications due to a lack of clear models regarding responsibility or overlooked model decay in their operations. This review highlights integrated model operations solutions that help in managing a large number of models in a systematic manner using strategies that include integrated models in a unified framework that promotes a system or structure that favors scalability in their architecture model.

Keywords: MLOPs Frameworks, Model Governance, Scalable Architecture, Continuous Monitoring, Dependency Management.

Introduction
With machine learning being integrated throughout departments and products, dozens or even hundreds of models are currently in production at most organizations. What started as one experimental model can easily expand to be a rich ecosystem with multiple business purposes such as risk assessment and anomaly detection up to recommendation engines and demand forecasting. Both models serve particular requirements, but they form a complex interdependence, data flow, and operations requirement. The growth of machine learning systems in the business world has created a lot of complexity which goes way beyond the model algorithms, and into data dependencies, configuration management, system requirements and monitoring systems which altogether create what researchers refer to as the hidden technical debt of machine learning [1]. Such technical debt compounds silently as organizations grow their machine learning activities to produce maintenance overheads that eventually become more expensive than the original model development. This is compounded when the models communicate with each other, exchange data pipelines, and rely on external systems to create a complicated dependency graph that is growing harder to uphold without systematic methods of governance and lifecycle management.

This is because the absence of model management results in redundancy, confusion and operational risk particularly when models get old, drift, or when they need updating. Teams end up being duplicates of efforts in creating similar capabilities, models decay without notice, and key questions on ownership and compliance are left unanswered. To figure out the reasons behind data science project failures, it is necessary to look at the disconnect between experimental creation and operational production where most difficulties are not due to the constrained nature of algorithms but instead due to organizational and operational aspects [2]. Failure of projects is common because of a lack of proper infrastructure to deploy models, lack of proper cooperation between data science and engineering institutions, absence of distinct business goals that relate to model output, and the lack of monitoring mechanisms that can identify when

models are no longer effective. The literature review has indicated that the key failure points include low quality of data that compromises the reliability of models, failure to match model objectives with the real business requirements, failure to scale experimental models to production demands, and failure to be interpretable to make stakeholders trust and embrace the model recommendations. Furthermore, most organizations do not adequately plan to sustain models once deployed and assume machine learning is a short development project but not an operating liability that needs specific resources and focus.

This paper describes the value of MLOps frameworks to manage large inventories of models, addressing lifecycle management, governance, and scalability, and strategies that can be useful when enterprise teams face such complexity. Organizations can help to make their haphazard collection of models into a logical, manageable system that generates sustained business value and has low operational overhead and technical debt by applying structured model registry, automated monitoring, standardized model deployment pipelines and extensive documentation practices. The MLOps paradigm builds upon the ideas of DevOps to machine learning environments, with an emphasis on automation, reproducibility, collaboration, and constant improvement in the entire model lifecycle, starting with initial experimentation and extending through production deployment and eventual retirement.

Establishing Centralized Model Governance

Multi-model management is mostly based on good governance structures which are capable of giving one a visibility of the whole model portfolio. The model registry is an authoritative source of truth about all models, listing their uses, owners, dependences, and state. In so much more than version control, this registry records abundant metadata of training data-sets, performance profiles of models, approval, and business reasons. The application of the MLOps paradigm fills the gaping hole between the experimental model creation and the deployment of the model in production with systematic approaches to managing model lifecycle [3]. In a company that uses a centralized-model registry, full catalogs are generated, which record the design choices undertaken by each model, the set up of hyperparameters, the origin of training data, validation outcomes, and deployment history, establishing a transparent environment in previously isolated teams. In the future, this centralization will become critical as any enterprise starts to scale out its machine learning functions because in its absence, duplicated work, or inconsistent practice, and the loss of institutional knowledge have frequently happened whenever team members left. Model registries are not just a technical repository of artifacts; they can be seen as a channel of communication between data scientists, machine engineering, and software developers, on one hand, and the business stakeholders, on the other hand, about what models they have, their performance, and the kind of business problem that they address, downstream effects being less friction across cross-functional cooperation and time to value acceleration of machine learning projects.

Governance architectures define standard procedures on model development, validation and deployment. These include making sure that all the models undergo the right reviews prior to being sent into production, a check of technical viability, issues of fairness, and adherence to business goals. Such documentation requirement will bring in uniformity such that new team members will find it easier to comprehend the available models and audit decisions made by stakeholders. The MLOps taxonomy recognizes a number of key elements which must be incorporated in governance structures, such as continuous integration and continuous deployment pipelines, modified to machine learning workflows, automated testing processes, which can check code quality and model performance and monitoring systems, which identify degradations in production environments [3]. The institutionalization of best practices in standardized processes makes sure that best practices are institutionalized throughout the organization so that models pass through a rigorous validation process regardless of who develops them and who business unit will sponsor them. These structures establish straightforward standards of model acceptance that establish the standard of acceptable performance, metrics of fairness among different demographic categories, explainability requirements to assist the stakeholders to comprehend the model rationale, and documentation conventions that include design choices and recognized constraints. A governing framework needs to be sufficiently detailed and at the same time practical without being

bureaucratic and slowing innovation but still with enough rigor to stop the flawed models emerging into the production and resulting in business or reputational risks.

The approval workflows and the access controls prevent unnecessary changes that would permit agile development. The permission is efficiently granted to different stakeholders-data scientists, engineers, compliance officers and business owners-as dictated by their roles. This framework brings the appropriate balance of speed of innovation, and control that offers models a mechanism of progressing through the stages of development without going through the significant checkpoints. The knowledge that machine learning implementation issues are exclusively technical issues demonstrates that it is part of a greater scope that encompasses organizational elements, information quality issues, and integration issues [4]. This study found that successful machine learning implementations need to be coordinated with various organizational functions, and the functions and decision-making authority must be well defined throughout the life cycle of models. These duties are formalized in the form of role-based access controls that provide data scientists the authority to do experiments with model architectures and training processes but limit the authority to do production deployment to engineers identified as having the responsibility of guaranteeing operational readiness. To facilitate the work of auditors and guarantee the compliance with the regulation, compliance officers get read access throughout the model portfolio without hampering development processes. This governance acknowledges that failures in machine learning projects are not due to the flaws associated with the algorithms, but rather to the lack of alignment between technical teams and business stakeholders, lack of proper infrastructure to deploy the production and lack of proper processes to maintain and monitor after initial deployments.

Table 1: Organizational Challenges in Machine Learning Deployment vs. Governance Solutions [3, 4]

Challenge Category	Specific Issue	Frequency/Impact	Governance Solution	Expected Outcome
Team Coordination	Duplicated efforts across teams	High - leads to wasted resources	Centralized model registry with visibility	Elimination of redundant development
Knowledge Management	Loss of institutional knowledge during transitions	High impedance - impedes maintenance	Comprehensive documentation standards	Preserved expertise, easier handoffs
Deployment Readiness	Technical obstacles and integration complexity	Critical - blocks production	Standardized deployment processes	Streamlined path to production
Stakeholder Alignment	Misalignment between technical and business teams	High - causes project failure	Clear responsibility delineation, shared registry	Improved collaboration, aligned objectives
Compliance & Auditing	Unclear ownership and regulatory adherence	Critical - creates legal risk	Role-based access, audit trails	Regulatory compliance, clear accountability
Model Quality Assurance	Flawed models reaching production	Critical - damages business outcomes	Approval checkpoints, fairness metrics	Prevented production issues, maintained trust

Architecting for Scalability and Consistency

The multi-model management architecture in MLOps must have the form that enhances reusability and reduces the burden to maintenance. The common infrastructure implies stores, data pipelines and inference services-induce redundant development and ensured consistency. Teams do not need to reinvent

the art of data handling, model serving, and monitoring but inherit known patterns as they progressively work on top of common underlying components. By integrating MLOps practices into the continuous integration and continuous deployment process, organizations entirely transform how they build and maintain large-scale machine learning systems 5. Studies demonstrate that accuracy of models and their operational reliability is greatly enhanced by automating the model training, validation and deployment stages since it guarantees uniformity in the operations throughout the entire life cycle of development. The shared infrastructure between these parts eliminates the problem of fragmentation by different teams developing individual solutions hence forming technical silos. This avoids knowledge transfer and maintenance when the priorities of the organizations change. The feature stores specifically solve a very pressing problem in machine learning processes: they offer a single repository in which feature definitions can be standardized, versioned and reused by many models. Thus, they make the training and serving environments consistent and prevent the training-serving skew that often can ruin model performance when in production. The architectural approach emphasizes that machine learning infrastructure is a platform, rather than a collection of discrete mechanisms; therefore, platform teams need not invest in complex platforms with regards to monitoring, security, and performance optimization that are valuable to all models, but instead, every project must independently satisfy those core needs.

Model packaging and containerization allows the deployment of any model. The model predicting customer churn or manufacturing defects uses the same deployment conventions and log format as well as the same API. This simplifies operations significantly: platform teams develop tools that cut across the entire model portfolio, instead of specializing to each use case. Continuous Integration, Continuous Deployment model of MLOps develops an automated work process to support quality of models by fully testing models prior to deployment. It will minimize the possibility of taking to production a poorly performing or unsound model 5. Since the deployment is containerized, the environment of development, staging, and production systems is the same, and the very familiar failure mode of models performing well in an experimental environment but failing unexpectedly in a production workload and data distribution situation is avoided. Models can be easily inter-operated with downstream business applications using standard APIs and interfaces; consuming systems can use the same protocols to interact with models regardless of the underlying algorithms, frameworks, or other implementation characteristics, making them less coupled and allowing models to be updated without any supporting systems having to be changed.

Shared utility libraries and template projects accelerate development through the ability to enforce best practice. The approved templates on which Data scientists can create new models contain the pre-existing monitoring hooks, logging conventions as well as documentation structures. Delivered libraries of common utility implementations (tested and debugged) are used to save development time and errors, as well as to find and use feature engineering, model evaluation, and bias detection. The hidden costs of the machine learning system in technical debt are linked to situations when the organizations fail to address the issues related to data dependencies, the complexity of the model, and fairness consideration in a systematic way. The research also addresses the technical debt one can encounter in machine learning, such as a problem in code quality, not to mention a problem in data quality, lack of documentation, lack of testing, and lack of continuous monitoring of model fairness between dissimilar demographic groups. The mitigation of these risks is provided through template-based development which hardens fairness tests, bias detection mechanisms and demands extensive documentation into standard project structure up front. This ensures that such important considerations are managed in a consistent and dependable manner, involving various staffs and initiatives. These templates encode organizational experience and regulatory specifications into usable models that lessen the cognitive load on individual information scientists with consistency in the manner models are created, confirmed and implemented throughout the enterprise.

Table 2: Shared Infrastructure Components and Their Benefits in MLOps Architecture [5, 6]

Infrastructure Component	Primary Purpose	Key Capability	Problem Solved	Organizational Benefit
Feature Stores	Centralized feature repository	Standardized, versioned feature definitions	Training-serving skew	Synchronized environments, reusable features across models
Data Pipelines	Automated data processing workflows	Consistent data transformation and validation	Data fragmentation, quality issues	Reduced redundant development, improved data reliability
Inference Services	Unified model serving platform	Standardized deployment and API exposure	Environmental inconsistency	Simplified operations, uniform model access
Continuous Integration Pipelines	Automated model validation	Comprehensive testing before deployment	Underperforming models reaching production	Enhanced model accuracy, reduced deployment risk
Template Projects	Standardized development starting points	Pre-configured monitoring, logging, and documentation	Inconsistent practices, technical debt	Accelerated development, enforced best practices
Shared Utility Libraries	Reusable code implementations	Tested feature engineering, evaluation, and bias detection	Code duplication, implementation errors	Reduced development time, consistent quality
Container-Based Deployment	Environment standardization	Consistent runtime across development and production	Environment-specific failures	Eliminated deployment inconsistencies

Implementing Continuous Monitoring and Maintenance

Production models need constant attention to identify degradation and trigger necessary interventions. Automated monitoring systems track performance metrics, data quality indications, and operational health across all the deployed models. These systems identify concerning trends-like rising prediction error, shifting input distributions, or increasing latency-well before they significantly affect business outcomes. In a large-scale machine learning system, an implementation of cognitive computing architecture requires sophisticated monitoring frameworks, capable of handling computational complexity and volumes of data from production deployments of systems [7]. Research has shown that a scalable architecture for machine learning needs to embed monitoring at different levels, ranging from model performance metrics to patterns of system-wide resource utilization, ensuring that the degradation of any component is surfaced well before it cascades to dependent systems. These monitoring systems must have very low performance overhead while collecting comprehensive telemetry data across distributed infrastructure and capturing model predictions, ground truth labels, feature distributions, inference latencies, resource consumption patterns, and system health indicators. The cognitive computing approach emphasizes intelligent monitoring systems for identifying anomalies, correlating symptoms across multiple models, and prioritizing alerts based on business impact rather than mere notification for every metric deviation. Monitoring infrastructure needs to scale with machine learning systems that scale to handle very large datasets and complex models, using distributed data collection, efficient storage mechanisms, and real-time analytics, enabling platform teams to maintain visibility across extensive model portfolios and not be overwhelmed by sheer data volume.

Drift detection mechanisms raise flags when model assumptions cease to hold as a result of shifting data patterns or evolving business conditions. Various statistical tests pit recent data against training set distributions and trigger alerts when the divergence crosses tolerable limits. Performance tracking, which involves pitting actual outcomes against predictions, lets teams know when models lose their predictive powers, even in cases when data distributions seem to be stable. Understanding the full life cycle of machine learning models shows us that monitoring and maintenance are important stages that have a significant impact on the long-term success of the models and the delivery of business value [8]. The model life cycle involves problem definition, data collection and preparation, model development and training, deployment into production, continuous monitoring, maintenance, and model retirement in cases where the underlying capabilities become obsolete or are overtaken by superior approaches. Research pinpoints that efficient drift detection relies on the determination of baseline performance metrics within the validation phase and their constant comparison with production performance using statistical methods corresponding to the type of model and application domain in question. The challenge significantly increases in real-world deployments, where ground truth labels may arrive with considerable delay, requiring proxy metrics and leading indicators that could indicate potential issues before definitive performance measures become available. In this respect, an organization has to carefully balance sensitivity and specificity during drift detection, tuning alert thresholds to minimize false positives, which create alert fatigue, but make sure actual deterioration gets attention well before the impact becomes severe.

Structured retraining workflows respond to detected problems with appropriate interventions. Some models need to be refreshed frequently with new data; others remain stable for long periods. Automated pipelines handle the routine retraining for high-cadence models, while alert systems notify teams when manual intervention is called for. Version control and rollback make it easy for teams to revert to problematic updates when retraining introduces unexpected issues. The lifecycle view makes it clear that model maintenance continues through their operational life, with different models having different stability characteristics that dictate appropriate maintenance strategies 8. High-frequency models that operate in fast-evolving domains rely on automated retraining pipelines that constantly refresh their models with fresh data, validate retrained versions against hold-out datasets, and deploy updates with minimal human intervention. By contrast, models deployed in stable domains can perform effectively for an extended period with only periodic retraining, initiated either by detected drift or scheduled reviews. Any retraining must be preceded by comprehensive validation processes to confirm improvements over previous versions before their deployment, preventing a situation where automated systems inadvertently degrade model quality. Version control systems keep complete audit trails of every model iteration, which let teams understand evolutionary trajectories and easily return to previous versions when updates introduce unexpected regressions.

Table 3: Monitoring Framework Components and Detection Capabilities [7, 8]

Monitoring Component	Key Metrics	Detection Focus	Alert Trigger
Model Performance	Prediction accuracy, error rates	Model degradation	Performance threshold violations
Feature Distribution	Input statistics, distribution shifts	Data drift	Divergence from training baseline
Inference Latency	Response time	Performance bottlenecks	Latency threshold exceedance
System Health	Service availability, error rates	Component failures	Cross-system anomalies

Concept Drift	Feature-outcome relationships	Evolving patterns	Baseline deviation

Managing Dependencies and Model Interactions

Models seldom exist in isolation; they consume outputs from upstream models, share data sources, and drive business processes together. Mapping these interdependencies helps teams anticipate the downstream consequences of changes before they deploy updates. Dependency graphs visualize which models are consuming shared features, relying on a common data pipeline, or each other's predictions. Empirical studies of technical debt in machine learning software have identified 23 distinct categories of admitted technical debt; the most significant sources of accumulated debt that hinder long-term maintainability involve dependency management and architectural complexity [9]. The study showed that machine learning systems have forms of technical debt not present in traditional software, such as entangled dependencies, in which changes to one component unexpectedly affect seemingly unrelated models; pipeline jungles, in which complex data processing workflows become hard to modify or debug; and dead experimental code paths that remain in production systems because teams fear removing them might break dependent components. Analysis of open-source machine learning codebases illustrates how developers acknowledge-either through code comments or documentation-that the systems they create contain fragile interdependencies, but they lack either time or resources to refactor the system toward more maintainable architectures. The research puts forth the idea that model dependencies need to be managed by considering machine learning systems as complex distributed applications and not loose collections of independent models that require end-to-end dependency tracking, impact analysis tools, and architectural patterns that minimize coupling between components while still enabling necessary collaboration and data sharing across models.

Coordinated deployment strategies account for these interactions. When updating several related models, there are a number of decisions to be made: whether to deploy all models simultaneously, stage the releases carefully, or even maintain backwards compatibility during transitions. Testing frameworks validate not only individual model performance but also the system-level behavior of interacting models. The study of technical debt in machine learning systems shows convincingly that poor testing practices and an inability to consider system-level interactions contribute greatly to production failures and maintenance problems [9]. Researchers found that about 40% of machine learning technical debt involves testing challenges, including an inability to recreate production environments in test settings, inadequate coverage for edge cases and failure modes, and poor validation of how models will actually be deployed in conjunction with other components in a complex production system. Coordinated deployment strategies must address these testing challenges through: comprehensive integration test suites that exercise complete prediction pipelines; canary deployment patterns that expose small populations of users to changes before broader rollouts; and the ability to roll back quickly when a coordinated update introduces unexpected system-level issues. Testing machine learning systems is fundamentally different from testing traditional software because correctness cannot be validated definitively with unit tests. Models may pass individual validation checks with flying colors yet exhibit undesirable behavior when integrated with production data pipelines and dependent systems.

Data lineage tracking follows information flow from source systems through transformations to final predictions. This is crucial for debugging unexpected behavior, ensuring the conformance of data usage policies, and interpreting how far downstream models are affected by upstream data quality issues. When problems occur, teams can identify root causes rather than investigate every individual model. Literature that investigated automatic data quality assessment argued that comprehensive data lineage tracking and quality monitoring were foundational requirements for any trustworthy machine learning system [10]. It was proved in this research work that the dimensions in which the quality of data manifests include completeness, consistency, accuracy, timeliness, and validity, each with different mechanisms for detection and remediation. It is also argued that poor data quality is among the leading factors associated with the failure of machine learning projects. Whatever algorithmic sophistication exists, once there is

biased, incomplete, or corrupted data on which a model is trained, the resulting predictions are always unreliable. An automatic data quality system should permanently monitor streams of incoming data by comparing current observations with expected distributions and historical patterns to detect anomalies that could reveal failures of upstream systems, data corruption, or a structural change in the processes that generate the data, thus invalidating model assumptions.

Table 4: Technical Debt Categories in ML Systems and Their Impacts [9, 10]

Technical Debt Category	Manifestation	Impact on System	Percentage of Total Debt	Mitigation Strategy
Dependency Management	Entangled dependencies between models	Unexpected effects from component changes	Major contributor	Comprehensive dependency tracking, impact analysis tools
Testing Challenges	Inadequate integration testing	Production failures, maintenance difficulties	40%	Integration test suites, canary deployments, and rollback capabilities
Pipeline Jungles	Complex data processing workflows	Difficult modification and debugging	Significant source	Architectural patterns minimizing coupling
Dead Experimental Code	Unused code paths in production	Fear of removal breaking dependencies	Contributing factor	Regular code audits, clear ownership
System-Level Interactions	Insufficient validation of model interactions	Problematic behavior in production	Part of testing debt	Complete pipeline testing, staged rollouts

Conclusion

A multi-model management paradigm within the enterprise necessitates systematic practices that correspond to far more than the optimization of models, incorporating lifecycle management, governance, and architecture standardization. The scaling of the organization with machine learning necessitates the understanding that technical excellence, by itself and as the culmination of technological efforts, does not suffice without the subsequent focus on investments in the development of centralized registry-based visibility for the portfolio, shared infrastructural development to avoid ‘redundant development efforts,’ continuous monitoring to proactively address decreases, and understanding and preventing the failure cascade among interlinked systems. MLOps corresponds to the challenges by leveraging the effective and successful principles of DevOps, as applied to machine learning, to emphasize the identification and focus on automation, reproducibility, and collaboration throughout the lifecycle of models, and that too starting from the exploratory phase to productionizing and eventually to model retirement. Outlining effective governance with the proper definition and implementation of ‘role-based access control and approval workflows' necessitates ‘innovation velocity' while also maintaining the proper and necessary ‘balance between necessary oversight, regulatory compliance, and trust and buy-in with various stakeholders.’ Common architecture elements correspond to the ‘feature store, containerized platforms, and development templates' that shall propel and establish ‘speed to value and the establishment and promotion of all proper practices related to fairness evaluation, bias detection, and overarching documentation.’ This, correspondingly, is where the increase in the size and complexity of model portfolios corresponds to the challenges that necessitate the effective management practices to lay the essential grounds for the organization to establish the achievement and optimization of the utmost ‘business value realization from machine learning investments, while also maintaining the optimization

and necessary focus on the management and control of operating cost and minimizing the buildup and creation of technical debt, besides remaining effective and nimble as per the paradigms and changing and evolving as per the changing and evolving unexplored and new and emerging ‘business conditions and changing and evolving new opportunities.

References

[1] Dev Kumar Chaudhary et al., "A Review on Hidden Debts in Machine Learning Systems," ResearchGate, August 2018. [Online]. Available: https://www.researchgate.net/publication/334238242_A_Review_on_Hidden_Debts_in_Machine_Learning_Systems
[2] Balaram Panda., "Why Data Science Projects Fail," ResearchGate, August 2023. [Online]. Available: https://www.researchgate.net/publication/373017005_Why_Data_Science_Projects_Fail
[3] Matteo Testi et al., "MLOps: A Taxonomy and a Methodology," ResearchGate, June 2022. [Online]. Available: https://www.researchgate.net/publication/361669590_MLOps_A_Taxonomy_and_a_Methodology
[4] Enrico Barbierato & Alice Gatti., "The Challenges of Machine Learning: A Critical Review," ResearchGate, January 2024. [Online]. Available: https://www.researchgate.net/publication/377541001_The_Challenges_of_Machine_Learning_A_Critical_Review
[5] Suresh Kumar Gaware et al., "MLOps for Enhancing the Accuracy of Machine Learning Models using DevOps Continuous Integration and Continuous Deployment," ResearchGate, June 2023. [Online]. Available: https://www.researchgate.net/publication/371218868_MLOps_for_Enhancing_the_Accuracy_of_Machine_Learning_Models_using_DevOps_Continuous_Integration_and_Continuous_Deployment
[6] Chong Huang et al., "Hidden Technical Debts for Fair Machine Learning in Financial Services," ResearchGate, March 2021. [Online]. Available: https://www.researchgate.net/publication/350254067_Hidden_Technical_Debts_for_Fair_Machine_Learning_in_Financial_Services
[7] Samir Mittal, "Cognitive Computing Architectures for Machine Deep Learning at Scale," ResearchGate, June 2017. [Online]. Available: https://www.researchgate.net/publication/318496497_Cognitive_Computing_Architectures_for_Machine_Deep_Learning_at_Scale
[8] Emmanuel OK et al., "Lifecycle of Machine Learning Models," ResearchGate, April 2022. [Online]. Available: https://www.researchgate.net/publication/389713001_Lifecycle_of_Machine_Learning_Models
[9] David Obrien et al., "23 shades of admitted technical debt: an empirical study on machine learning software," ResearchGate, November 2022. [Online]. Available: https://www.researchgate.net/publication/365269195_23_shades_of_self-admitted_technical_debt_an_empirical_study_on_machine_learning_software
[10] Deepak Chandran & Vikram Gupta., "A Short Review of the Literature on Automatic Data Quality," ResearchGate, January 2022. [Online]. Available: https://www.researchgate.net/publication/360942355_A_Short_Review_of_the_Literature_on_Automatic_Data_Quality