

Machine Learning Lineage for Trustworthy Machine Learning Systems

Information Framework for MLOps Pipelines

Mikko Raatikainen^{ID}, University of Helsinki

Charalampos Souris, Silo AI

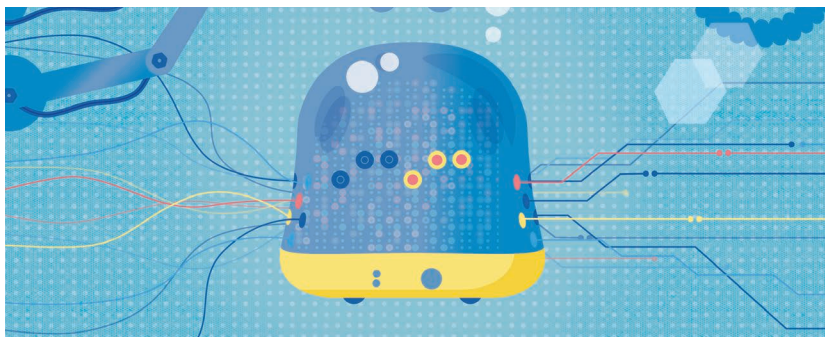
Jukka Remes^{ID}, Haaga-Helia University of Applied Sciences and Silo AI

Vlad Stirbu^{ID}, CompliancePal

// We describe machine learning (ML) lineage as a framework to holistically capture and connect the required information about ML model development and operations. ML lineage distinguishes between model and prediction levels, conceptually encompassing separate yet interconnected core domains for the project, experiment, model, and prediction. //

THE RESURGENCE OF AI, especially machine learning (ML), has reached technological maturity and is applied broadly. ML-based systems are fundamentally software-driven¹ and inherently involve software components, such as those in the user interface and back end. However, the nondeterministic and highly opaque or black-box nature of the ML part—based on learning behavior from training data rather than being algorithmically prescribed—can benefit from distinct engineering practices. Such practices for holistic lifecycle management of ML, encompassing development, training, deployment, and serving, are maturing under the overarching framework of MLOps.²

ML can provide business value through invaluable functionalities and features. ML also brings forward unique risks and performance quality hurdles that require attention.³ Concurrently, requirements for AI/ML



Digital Object Identifier 10.1109/MS.2024.3414317

Date of publication 19 June 2024; date of current version 11 December 2024.

This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see <https://creativecommons.org/licenses/by/4.0/>

have grown increasingly stringent and complex, requiring transparency and pieces of evidence for trustworthiness. There are societal discussions about the implications of AI technologies. To address challenges regarding AI, public authorities, researchers, and industry experts have established regulations and practices. For example, there is the AI Act proposal in the European Union and a comparable Biden administration Executive Order on AI in the United States. Prominent information technology giants are also actively investing in incorporating principles of responsibility into AI systems, such as Microsoft's "Responsible AI Standard" playbook and Google's "Responsible AI Practices." Likewise, scholars have delved into AI governance⁴ and ethics.⁵

Challenges in ML Information

Information related to ML-based systems within production environments throughout the lifecycle journey is needed to ensure and monitor business value and trustworthiness. However, the emphasis on AI regulation, governance, and ethics revolves around high-level concepts, requirements, and individual practices. In contrast, MLOps focuses on pipelines to produce and maintain ML model artifacts. Organizations face a shortage of straightforward and unified means to comprehensively capture, store, connect, trace, and retrieve relevant end-to-end information about their AI-based systems.^{6,7,8} Therefore, organizations end up employing diverse approaches driven by specific use case requirements, tools, and team preferences. At best, tools like MLFlow paired with techniques like model cards and sheets^{9,10,11} capture static pieces of information pertaining to specific

phases but lack a dynamic holistic view and integration with each other. According to our experiences, ML and data engineering teams practicing MLOps also seem to prioritize technical superiority and integrations, potentially leading to a deprioritization of other efforts required for trustworthy ML-based systems.

Data management and software engineering have more mature and holistic solutions for information and artifacts. One culminating point is formulated as *data lineage*, or *provenance*, which addresses inquiries about the origin of data, the transformations data undergo, and data usage locations.¹² Data lineage differentiates the schema data lineage at the dataset or schema level from the instance data lineage about individual data entries, both addressing the aspects of the origins of data ("where") and the manipulations of data ("how"). Data lineage has demonstrated its relevance to practitioners, resulting in support from major data tooling vendors. Software engineering has established traceable ways of working that consider an artifact as a specification for other artifacts or pass an artifact forward in a pipeline. For instance, downstream workflows start from requirements management or issue tracking, or various concerns are addressed through abstraction-refinement or design decisions in structural views. Moreover, software engineering ways of working include checkpoints or quality gates, such as triaging bugs, pull requests, and reviews, before proceeding further.

The challenges in the information management of ML pertain to the involvement of data and software that can be succinctly framed as follows: How can comprehensive information on ML-based systems and their development and operations be

captured, integrated, and effectively managed throughout their lifecycle? While data and software engineering practices are insufficient and may not be directly transferable, their underlying principles can be adopted.

The ML Lineage Framework

Organizations must establish a comprehensive framework to implement an integrated information model and its associated best practices. This framework is essential for ensuring and demonstrating value and trustworthiness. The information model should span the entire lifecycle, covering data acquisition and experimentation, operational deployment, and serving. The model should also extend to retraining, maintenance, and subsequent incremental development. It is imperative that this comprehensive information model actively engages all key stakeholders and their concerns. We refer to the framework that addresses this information model as *ML lineage*.

The ML lineage fundamentally differentiates two interconnected levels (Figure 1):

1. *Model-level ML lineage* pertains to the information involved in developing, training, integrating, and deploying an ML-based system. Model-level ML lineage gathers input from relevant stakeholders and tools and comprehensively addresses the workflows and lifecycle from initial objectives downstream to operational readiness with possible updates.
2. *Prediction-level ML lineage* captures information about the operations of a specific model and its predictions, where transparency may be crucial. Focusing solely on predictions may

be insufficient, but considering input holistically, such as if the input is augmented with some contextual information, can be necessary. Respectively, outcomes extend beyond predictions to include users' behavior, such as their actions in response to the prediction and whether users consent to and understand it.

In practice, the prediction-level ML lineage extends the model-level ML lineage from design to operations time. The prediction-level ML lineage often requires the connection of predictions upstream to the information stored within the model-level ML lineage, such as model or training data characteristics. Thus, prediction-level ML lineage is not distinguished as an ontological instance, but it represents the usage in the operation time realm of the development time-oriented model-level lineage. Therefore, model-level lineage must be established before it is possible to proceed establishing the prediction-level lineage.

The ML lineage comprehensively addresses information, distinguishing among the “where,” “how,” and “why” aspects. The “where” aspect encompasses sources, such as training data origins and the model’s journey through the MLOps pipeline, including its quality gates. ML lineage must facilitate traceability back and forth within the MLOps pipeline and beyond, such as connecting quality assurance reports to compliance policies. The “how” aspect provides specific details on transformations, such as how the model was constructed from training data, thereby complementing the “where” aspect with details. Finally, the “why” aspect specifies the rationales and justifications for any artifact’s design decisions, such as the selection of training data or algorithms.

Information Diversity in ML Lineage

The purpose and key advantage of ML lineage lie in its capability to comprehensively capture information and engage stakeholders with specific domain knowledge. Below, we explore the diversity of key concerns or aspects.

Business Subject Matter

Business is pivotal in eliciting the problem within which an ML system operates, being accountable for the requirements, and determining the appropriate business metrics and key performance indicators (KPIs) to evaluate success.¹³ The metrics and KPIs should be realized and integrated into ML lineage and the monitoring pipelines and dashboards to the greatest extent possible.

Risk Management and Compliance

This concern encompasses the responsibility of identifying risks following

established industry practices, as well as recognizing the regulatory and ethical boundaries that an ML solution must adhere to. Stakeholders, including risk management and mitigation process owners, are accountable for updating the ML lineage model with relevant risk assessment reports, as well as ethics measures, such as bias and fairness, based on monitoring.⁵ External auditing bodies or regulators may also be involved in assessments.

Data Science and ML Expertise

Data science and ML expertise covers utilizing the correct training data and selecting suitable algorithms and parameters to solve business problems. While the work includes intrinsically recording technical design and experimentation in ML lineage, the responsibility extends to documenting the approaches and ensuring reproducibility in the results and rationales.¹⁴ The responsibility also extends to debugging

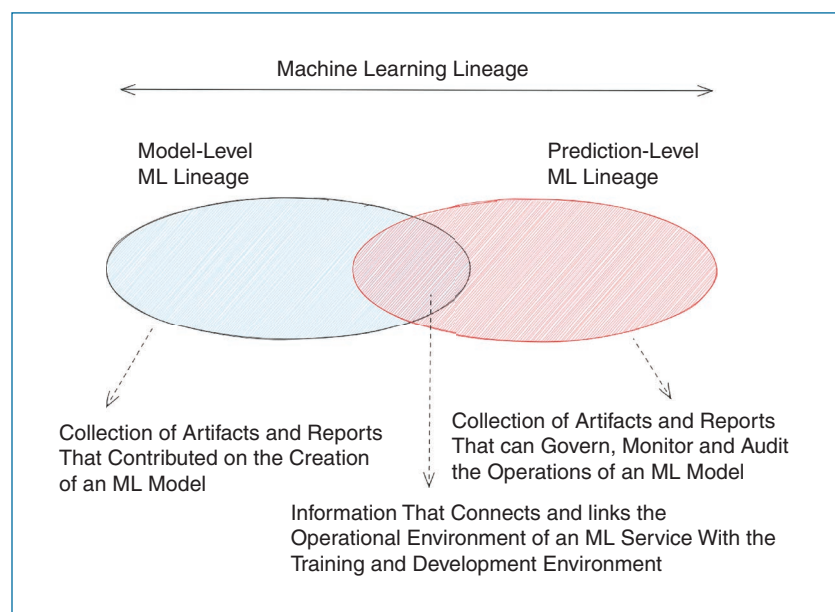


FIGURE 1. The representation of ML lineage involves a combination of model and prediction levels.

and preparing explanations of model operations.

Data and ML Engineering

Data and ML engineering, or *DataOps* or *MLOps*, covers scalable and reliable infrastructure around the ML model and system, such as DataOps and MLOps pipelines, production servers, and monitoring components and dashboards. Data and ML engineering actively interfaces with various software engineering activities throughout the DevOps phases. ML lineage must capture various infrastructure configurations, versions, and logs.

Data Ownership

Data owners and stewards ensure data quality, governance, and accessibility throughout development, training, testing, and serving. The data provided to the project must follow relevant regulatory requirements, such as the European Union's General Data Protection Regulation. Data owners, together with data scientists and business owners, define and follow the data quality KPIs and document them in the ML lineage information model.

Quality Assurance

Quality assurance (QA) measures and ensures that the developed ML systems meet the set requirements and constraints. In the context of ML lineage, the outcomes of QA work must be stored and made transparent and traceable for the sake of trustworthiness.

Prediction Comprehension

Augmenting predictions for comprehension can be important and nontrivial beyond the given output value, as a prediction often operates in a somewhat black-box manner. Users and customers may seek

insights into how specific predictions were reached, such as investment choices or loan approvals. They must be provided with explainability and QA reports on the models applied to these decisions. Here, the ML lineage model plays a crucial role in facilitating relevant information storage, correlation, and retrieval.

System Monitoring

This pertains to monitoring operations and serving time, evaluating whether an ML model and system function as intended and meet users' expectations by assessing input, inference, and outcome. Infrastructure monitoring involves tracking computing load, latency, and transaction logging. In service-level monitoring, the statistical reliability of inference implies that a single prediction may be inconclusive, yet it can still serve as an indicator. Recording and connecting predictions with the corresponding model-level lineage information enables, e.g., a broader analysis of model accuracy across larger datasets and facilitates the identification of data drift by comparing production and training data, both of which are peculiarities of ML.

ML Lineage Concepts and Schema

Information about the aforementioned concerns and stakeholders spans distinct domains characterized by their respective lifespans, scopes, and areas of interest. This information can be structured into identifiable entities, creating a schema. The schema embodies the ML lineage within and among these domains (Figure 2). Within the ML lineage, the core entities include project, model, experiment, and prediction entities, each accompanied by associated entities. We characterize the ML lineage

schema with regard to these four core entities, followed by an overview of their relationships, example content, and accountable stakeholders.

Project entity combines specific information within the ML lineage that holds crosscutting, holistic project-level significance and pertains to any model and its development within the project. Project entity encompasses objectives in "business entity," as well as "QA entity" and "risk and compliance entity" as internal and external constraints, respectively. The entities are also differentiated due to possibly different accountable stakeholders. These entities define measures and standards that must be measured and realized in other entities, including reporting or logging the outcomes. For example, risk and compliance can define bias and fairness metrics, such as nondiscriminatory measures for training data that each experiment's training data must satisfy and report. Project entity may also refer to the development infrastructure if deemed relevant to ML lineage. The project and its aggregating entities persist and remain operational throughout the project's lifecycle, undergoing potential evolution or change.

Experiment entity pertains to all ML model experiments within the context of a project, preserving information about the ML model's characteristics and quality, facilitating reproducibility. The source code version of the model is separated into a dedicated "source code entity." As training data are a significant aspect of ML but can have its data governance and lineage, we emphasize "data entity" as a closely related but separate entity. Experiment entity records the numerous experiments typically conducted with the aim of achieving a model that aligns with

the objectives outlined in the project entity. While not all experiments result in immediate success or appear to have more than short-lived effects, they may nonetheless incorporate valuable information. Different experiments also enclose variations in ML model characteristics, ranging from minor data or parameter adjustments to significant algorithm selection and design alterations.

Model entity refers to models that have been considered successful among the experiments and have been taken into further consideration downstream. That is, while multiple experiments can be conducted, only a few are chosen to undergo various quality gates and staging phases before possibly reaching the operational stage. Moreover, in the realm of an ML-based system, data drifts require retraining, multiple models may be concurrently operational, and multiple versions of each model can exist. Hence, documenting the lifecycle of these distinct models and their performance and behavior becomes imperative. This encompasses information concerning QA reports,

lineage to the originating experiment, and further training data, code, and configuration.

Prediction entity records specific prediction-related information for the above-described prediction-level lineage at operations time for a served model. Collections of predictions are utilized to monitor both a single model and the ML system. We distinguish “input data entity” from “user action entity” because software external to the model is often applied to extract and manage such information. Because the model does not operate in isolation, “infrastructure entity” captures the operational production software and infrastructure integrated and serving with the model while the model’s source code is accessible by lineage through experiment.

Although these core entities represent a key set of domains, their compositional structure may vary based on the use case, allowing mutability. For example, a project could manifest as a product or service, or the core entity aggregating other entities may be regarded as business entity.

Entities can sometimes extend beyond the project to encompass other initiatives within the organization, thereby defining intraorganizational standards and beyond. Risk and compliance entity, in particular, often encloses general regulations and standards.

These core entities are associated with each other and establish different granularities, as illustrated by the associations in Figure 2. For simplicity, we use associations between core entities to depict that the arrow points to a one-to-many relation, and aggregation to underscore that the core entities are often composed of other entities. For example, in a project, several experiments are conducted, but only some advance downstream until the deployed and serving models, each originating from a single experiment, serve multiple predictions. These associations are quintessential for the ML lineage by facilitating connections among different domains, concerns, and stakeholders. The core entities also facilitate various quality gates within and between them, necessitating contributions to the ML lineage before proceeding.

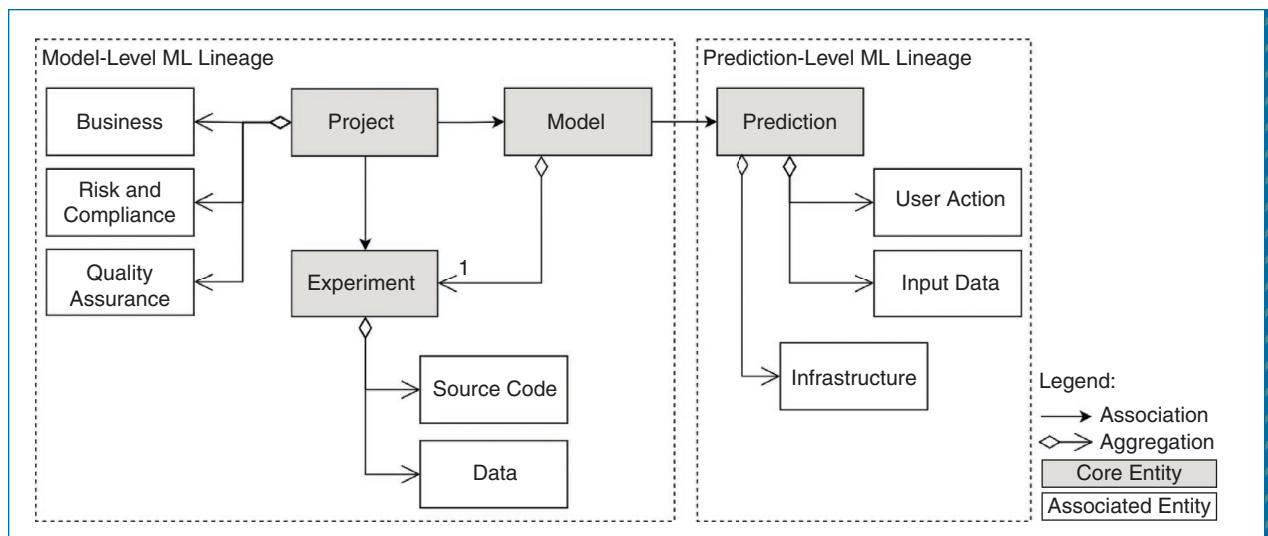


FIGURE 2. The ML lineage information model (in UML notation). “1” denotes the required cardinality.

The distinct domains of entities highlight diverse concerns and stakeholders, including that someone must be accountable for producing and maintaining information accessible to other stakeholders. An example illustrating various entities, content, and accountable stakeholders is in Table 1. The relevant information and accountable stakeholders depend on the organization. Context-specific responsibilities can also affect the compositional structure of ML lineage. For instance, one person may be assigned both risk and compliance accountability and QA, making it reasonable to combine respective entities. In any case, the purpose of the entities is to clarify accountability within the existing collaborative development environment, although the product or project manager ultimately bears the final responsibility.

ML Lineage in MLOps Pipelines

Integration of ML lineage within an MLOps pipeline is essential. Open

source pipelines have been developed, serving as archetypal references and often relying on tools like Kubeflow pipelines for orchestration, MLFlow for experiment tracking, KServe for serving, and Prometheus for monitoring and analytics. Cloud vendors' pipelines frequently make use of these same open source tools as well.

An MLOps pipeline based on tools like those mentioned above can relatively effortlessly generate and capture at least much technical information necessary for the ML lineage, thereby reducing the development effort required. For example, MLFlow, currently one of the most popular ML tracking tools, can be used in MLOps pipelines to capture extensive information about experiments.

However, the existing MLOps tools and pipelines have limitations. First, these tools face challenges in collecting and tracking all relevant information, particularly for business KPIs and quality assurance. For instance, MLFlow currently lacks the capability to include bias, fairness, and risk metrics in a manner

that applies to all models generated within a project. In particular, while collecting all details for prediction-level lineage may not require specialized tools, it is often neglected. Second, information is not integrated across tools. Despite the integrated workflows for model artifacts, each tool operates with its information in isolation. The information is not hidden or restricted. Rather, integrations do not exist, so integrated information and dashboards are not conveniently available.

As an example, an experimentation-driven open source pipeline¹⁵ based on the tools mentioned above readily generates technical details for ML lineage. However, owing to the inherent lack of correlation information among open source tools, the pipeline incorporates dedicated self-implemented components for collecting information from different tools, particularly for A/B testing purposes. In fact, the majority of the needed self-implemented code is focused on gathering information rather than managing integrations

Table 1. ML lineage entities, example content, and accountable stakeholder.

Entity	Example of contents	Accountable
Project	Project resources	Project manager
Business	Business objectives and KPIs	Business manager
Risk and compliance	Compliance restrictions, bias, and fairness metrics and thresholds	Auditors and risk experts
Quality assurance	Test and quality assurance material, such as benchmarks to test	QA experts
Model	Model version and state, QA reports for the data, and model code reference	Model owners (data scientists/ML experts)
Experiment	Experiment results, algorithms, parameters, and data used	Data scientists/ML engineers
Data	Reference to training datasets and their data lineage, data quality KPIs, and reports	Data owners/data engineers
Source code	Version of the source code	Data scientists
Prediction	Input and output data, model level lineage reference, and customer's problem and context	ML engineer

for ML model artifacts in the MLOps pipeline. The effort and size of these self-implemented components for information collection and correlation were modest, making it straightforward to realize a simple ML lineage. However, the ML lineage is limited because the pipeline does not gather complex QA, risk, and compliance information, as its tools also lack the ability to do so.

Application Advantages and Challenges

The advantage of ML lineage is that it enhances end-to-end accountability, transparency, and evidence for ML-based systems, thereby increasing business value and trustworthiness through thorough lifecycle documentation. ML lineage engages stakeholders at various organizational levels and roles, clarifying accountability, enabling clearer assignment of responsibilities, and facilitating interaction touchpoints. For example, developers become better informed about the business impact, while the QA team gains better oversight of technical details. ML lineage contributes advantageously to process rigor by documenting and tracing processes according to good engineering practices and facilitating quality gates. ML lineage easily integrates with existing workflows and tools, often requiring minimal additional effort, such as generating model cards or integrating with existing pipelines, as exemplified above. At the organizational level, ML lineage provides users or authorities with comprehensive end-to-end details for operations and predictions.

The challenges persist in the overhead any documentation creates. Documentation must serve a purpose, such as evidencing operations

ABOUT THE AUTHORS



MIKKO RAATIKAINEN is a software engineer at the University of Helsinki, 00100 Helsinki, Finland. His research interests include empirical research in software engineering and AI-based systems. Raatikainen received his D.Sc. (Tech.) in software engineering from Aalto University, Finland. Contact him at mikko.raatikainen@helsinki.fi.



CHARALAMPOS SOURIS works as a chief architect at Silo AI, 111 21 Stockholm, Sweden. His research interests include data and machine learning, with a focus on MLOps. Souris received his master's degree in telecommunication systems from KTH Royal Institute of Technology at Stockholm Sweden. Contact him at harry.souris@siloi.ai.



JUKKA REMES is a senior lecturer of information and communications technology and AI at Haaga-Helia University of Applied Sciences, 00531 Helsinki, Finland, and a lead AI solutions architect at Silo AI, 00180 Helsinki, Finland, where he has led technical and competence development for machine learning development and operations. His research interests include connecting machine learning research with product development and machine learning with software development. Remes received his D.Sc. (Tech.) in computer science and engineering from the University of Oulu, Finland. Contact him at jukka.remes@haaga-helia.fi.



VLAD STIRBU is the founder of CompliancePal, 33820 Tampere, Finland. His research interests include software development in regulation-intensive industries and quantum computing. Stirbu received his D.Sc. (Tech.) in software engineering from Tampere University of Technology, Finland. He is a Member of IEEE. Contact him at vlad.stirbu@compliancepal.eu.

or explicating design rationales. However, extensively integrating both model-level and prediction-level lineage and storing all predictions along with their relevant metadata can be prohibitively expensive, resource-intensive, and potentially unnecessary. This excessive information can become a notable barrier, and

stakeholders may object to the resources and effort needed, especially when their primary goal is to develop and operate only a few ML systems of low criticality. For example, comprehensive ML lineage can be excessive for noncritical business-to-consumer domains with high volumes, such as online marketing through web

page ads. Another practical application challenge is that in the current MLOps tools, fragmented information integration and the limitations of operations time and business-oriented metrics hinder a holistic view that requires development effort. Consequently, a challenge in applying ML lineage relates to determining the reasonable amount of information that should—or must—be collected and stored, depending on the use case, such as uncertainty and context, including safety criticality or risks.

ML lineage transcends mere data capture: It serves as a comprehensive end-to-end framework for capturing information about an ML system, enabling the monitoring of technology, business value, and quality objectives. ML lineage bridges the ML development-operations gap through the integration of the model and prediction-level information. ML lineage is best realized within an MLOps pipeline, where the focus is on upfront planning, adaptation, and tailored infrastructure rather than a technical overhaul. Establishing and implementing a comprehensive end-to-end ML lineage information model from the outset is essential, rather than retrofitting it in the later stages of development. ML lineage is particularly critical for systems requiring regulatory compliance, strict governance, trustworthiness, or high transparency. We also recommend considering the application of ML lineage in other use cases, as it can enhance overall trustworthiness and transparency, thereby increasing value and quality. Technically, the undertaking is reasonable, but organizations need to decide what and how business goals and quality targets will be measured and integrated into the

ML lineage. Finally, we are not prescribing the depth of the collected information; this should be contingent upon the use case, as ML lineage allows mutability. ☞

References

1. T. Mikkonen, J. Nurminen, M. Raatikainen, I. Fronza, N. Mäkitalo, and T. Männistö, “Is machine learning software just software: A maintainability view,” in *Proc. Int. Conf. Softw. Qual.*, New York, NY, USA: Springer-Verlag, 2021, pp. 94–105.
2. D. Kreuzberger, N. Kühl, and S. Hirschl, “Machine learning operations (MLOps): Overview, definition, and architecture,” *IEEE Access*, vol. 11, pp. 31,866–31,879, 2023, doi: [10.1109/ACCESS.2023.3262138](https://doi.org/10.1109/ACCESS.2023.3262138).
3. D. Sculley et al., “Hidden technical debt in machine learning systems,” in *Proc. Int. Conf. Neural Inf. Process. Syst.*, vol. 2, Cambridge, MA, USA: MIT Press, 2015, pp. 2503–2511.
4. M. Mäntymäki, M. Minkinen, T. Birkstedt, and M. Viljanen, “Defining organizational AI governance,” *AI Ethics*, vol. 2, no. 4, pp. 603–609, 2022, doi: [10.1007/s43681-022-00143-x](https://doi.org/10.1007/s43681-022-00143-x).
5. N. Mehrabi, F. Morstatter, N. Saxena, K. Lerman, and A. Galstyan, “A survey on bias and fairness in machine learning,” *ACM Comput. Surv. (CSUR)*, vol. 54, no. 6, pp. 1–35, 2021, doi: [10.1145/3457607](https://doi.org/10.1145/3457607).
6. L. Baier, F. Jöhren, and S. Seebacher, “Challenges in the deployment and operation of machine learning in practice,” in *Proc. Eur. Conf. Inform. Syst. (ECIS)*, vol. 1, 2019, pp. 1–19.
7. F. Königstorfer and S. Thalmann, “AI documentation: A path to accountability,” *J. Responsible Technol.*, vol. 11, 2022, Art. no. 100043, doi: [10.1016/j.jrt.2022.100043](https://doi.org/10.1016/j.jrt.2022.100043).
8. M. Steidl, V. Golendukhina, M. Felderer, and R. Ramler, “Automation and development effort in continuous AI development: A practitioners’ survey,” in *Proc. Euromicro Conf. Softw. Eng. Adv. Appl. (SEAA)*, 2023, pp. 120–127.
9. M. Mitchell et al., “Model cards for model reporting,” in *Proc. Conf. Fairness, Accountability, Transparency*, 2019, pp. 220–229, doi: [10.1145/3287560.3287596](https://doi.org/10.1145/3287560.3287596).
10. M. Arnold et al., “FactSheets: Increasing trust in AI services through supplier’s declarations of conformity,” *IBM J. Res. Develop.*, vol. 63, no. 4/5, pp. 6:1–6:13, Jul./Sep. 2019, doi: [10.1147/JRD.2019.2942288](https://doi.org/10.1147/JRD.2019.2942288).
11. T. K. Gilbert, N. Lambert, S. Dean, T. Zick, A. Snoswell, and S. Mehta, “Reward reports for reinforcement learning,” in *Proc. AAAI/ACM Conf. AI, Ethics, Soc.*, 2023, pp. 84–130, doi: [10.1145/3600211.3604698](https://doi.org/10.1145/3600211.3604698).
12. R. Ikeda and J. Widom, *Data Lineage: A Survey*. Redwood City, CA, USA: Stanford Univ. Publications, 2009, vol. 8090, no. 918, p. 1. [Online]. Available: <http://ilpubs.stanford.edu>
13. A. I. Canhoto and F. Clear, “Artificial intelligence and machine learning as business tools: A framework for diagnosing value destruction potential,” *Bus. Horiz.*, vol. 63, no. 2, pp. 183–193, 2020, doi: [10.1016/j.bushor.2019.11.003](https://doi.org/10.1016/j.bushor.2019.11.003).
14. R. Tatman, J. VanderPlas, and S. Dane, “A practical taxonomy of reproducibility for machine learning research,” presented at the 2nd Reproducibility Mach. Learn. Workshop Int. Conf. Mach. Learn., Stockholm, Sweden, 2018.
15. Y. Luo, M. Raatikainen, and J. Nurminen, “Autonomously adaptive machine learning systems: Experimentation-driven open-source pipeline,” in *Proc. 49th Euromicro Conf. Softw. Eng. Adv. Appl. (SEAA)*, Piscataway, NJ, USA: IEEE Press, 2023, pp. 44–52.