

CORNUCOPIA: Tool Support for Selecting Machine Learning Lifecycle Artifact Management Systems

Marius Schlegel^a and Kai-Uwe Sattler^b
TU Ilmenau, Germany

Keywords: Machine Learning Lifecycle, Artifact Management, Web Tool, Decision Support.

Abstract: The explorative and iterative nature of developing and operating machine learning (ML) applications leads to a variety of ML artifacts, such as datasets, models, hyperparameters, metrics, software, and configurations. To enable comparability, traceability, and reproducibility of ML artifacts across the ML lifecycle steps and iterations, platforms, frameworks, and tools have been developed to support their collection, storage, and management. Selecting the best-suited ML artifact management systems (AMSs) for a particular use case is often challenging and time-consuming due to the plethora of AMSs, their different focus, and imprecise specifications of features and properties. Based on assessment criteria and their application to a representative selection of more than 60 AMSs, this paper introduces an interactive web tool that enables the convenient and time-efficient exploration and comparison of ML AMSs.

1 INTRODUCTION

Machine learning (ML) approaches are well established in a wide range of application domains. In contrast to engineering traditional software, the development of ML systems is different: requirements engineering, data and feature preparation, model development, and model operation tasks are integrated into a unified lifecycle (see Fig. 1) which is often iterated several times (Chaoji et al., 2016; Amershi et al., 2019; Google, 2022). As a consequence, achieving comparability, traceability, reproducibility, and explainability of model and data artifacts across the lifecycle steps and iterations is challenging.


An essential foundation to meet these requirements is the collection, storage, and management of artifacts created and used within the ML lifecycle, such as models, datasets, metadata, and software (Sculley et al., 2015; Kumar et al., 2017; Polyzotis et al., 2018; Smith, 2018; Schelter et al., 2018). To support and simplify ML artifact management activities, a variety of platforms, systems, frameworks, and tools have been created, focusing on different aspects, such as pipeline management (Baylor et al., 2017; Schad et al., 2021; Luo et al., 2021), experiment management (Weights & Biases, 2022; Tsay et al., 2018; Greff et al., 2017), data and feature management (Feast Authors, 2022; Log-


icalClocks, 2022), model management (Vartak et al., 2016; Miao et al., 2017; Gharibi et al., 2019), as well as holistic lifecycle management (Zaharia et al., 2018; Allegro AI, 2022; Aguilar et al., 2021). We refer to these collectively as *ML artifact management systems* (ML AMSs).

The plethora of different ML AMSs makes it difficult and time-consuming to identify optimal candidates for a specific use case. Moreover, functional and non-functional properties are often not precisely specified and not easily comparable due to different terminology used in papers and documentations. Schlegel and Sattler developed a catalog of criteria for the assessment of ML AMSs and applied it to more than 60 ML AMSs from academia and industry (Schlegel and Sattler, 2022). However, there is no tool support that enables researchers and practitioners to conveniently explore and compare ML AMSs at criteria and subcriteria level.

This paper aims to close this gap and introduces tool support for this purpose. Specifically, we make the following contribution: Based on our assessment criteria and its application to a comprehensive selection of AMSs (§ 2), we present CORNUCOPIA – an interactive web tool for exploring and comparing ML AMSs (§ 3).^{1,2}

¹The tool's name is derived from the Latin name for the horn of plenty because our tool provides a container for the plethora of different ML AMSs and their individual

^a  <https://orcid.org/0000-0001-6596-2823>

^b  <https://orcid.org/0000-0003-1608-7721>

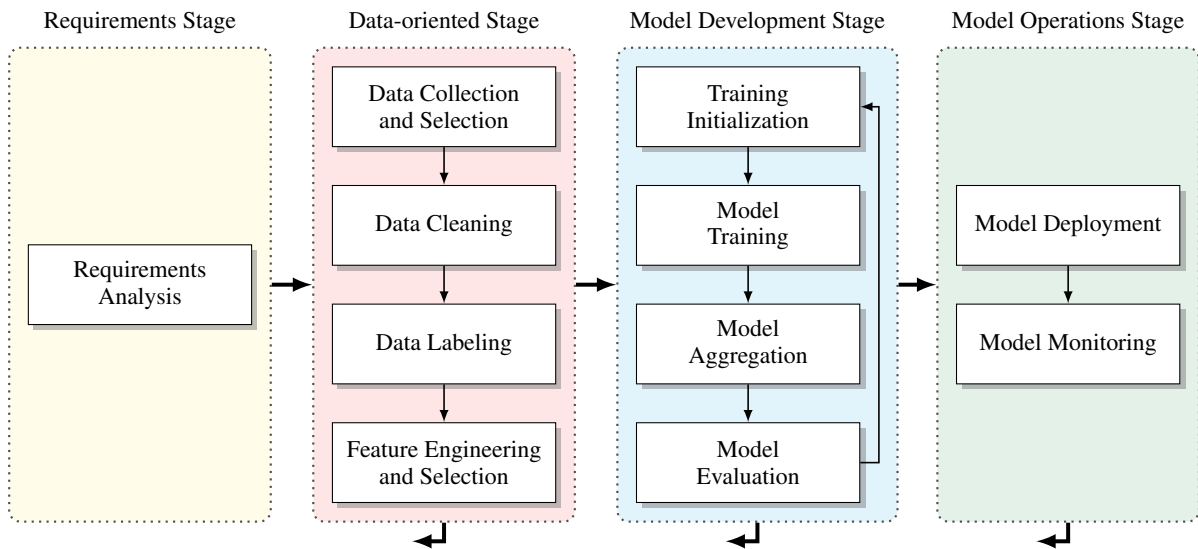


Figure 1: ML lifecycle based on (Chaoji et al., 2016; Amershi et al., 2019; Google, 2022).

2 ASSESSMENT FRAMEWORK

In this section, we outline the ML AMS assessment framework that is the foundation for CORNUCOPIA. Based on a systematic literature review investigating state-of-the-art and state-of-the-practice in ML artifact collection, storage, and management, we extracted a comprehensive catalog of categories, criteria (italicized), and subcriteria (in square brackets) (see Tab. 1). Categories group the assessment criteria, while subcriteria are used to subdivide criteria. In the following, we focus on the level of categories and criteria; more details can be taken from (Schlegel and Sattler, 2022).

Lifecycle Integration. This category describes for which parts of the ML lifecycle an ML AMS provides artifact collection and management capabilities. The stages form the criteria, with the steps assigned to each stage forming the subcriteria (see also Fig. 1).

Artifact Support. Orthogonal to the previous category, this category indicates which types of artifacts are supported and managed by an ML AMS. Based on the different kinds of ML artifacts, we distinguish between the criteria *Data-related*, *Model*, *Metadata*, and *Software Artifacts*. The first two criteria represent core resources that are either input, output, or both for a lifecycle step. *Data-related Artifacts* are datasets (used for training, validation, and testing), annotations

and labels, and features. *Model Artifacts* represent trained models. The latter two criteria represent the corresponding artifact types, that enable the comparability and reproducibility of individual ML lifecycle steps and their results. The criterion *Metadata Artifacts* covers different metadata types such as identifiers, data-related metadata, model-related metadata (such as inspectable model parameters, hyperparameters, and metrics), experiment and project abstractions, pipeline abstractions, and execution-related logs and statistics. The criterion *Software Artifacts* comprises source code and notebooks, e. g. for data processing, experimentation and model training, and serving, as well as configurations and execution-related environment dependencies and containers, e. g. Conda environments, Docker containers, or virtual machines.

Operations. This category indicates operations provided for handling and managing ML artifacts. The criterion *Logging & Versioning* represents any operations for logging or capturing single artifacts, creating checkpoints of a project comprising several artifacts, and reverting or rolling back to an earlier committed or snapshot version. *Exploration* includes any query, comparison, lineage, provenance, and visualization operations that help to gain concrete insights into ML artifacts. The criterion *Management* characterizes operations for handling and using stored artifacts, such as modification, deletion, execution (e. g. for the execution and orchestration of data processing or model training experiments and pipelines), and deployment (e. g. for offline (batch) and online (on-demand) model

properties and supports their structured investigation.

²<https://cornucopia-app.github.io>

Table 1: Assessment categories, criteria, and subcriteria.

Category	Criteria and Subcriteria
Lifecycle Integration	<i>Requirements Stage</i> [Model Requirements Analysis] <i>Data-oriented Stage</i> [Data Collection, Data Preparation & Cleaning, Data Labeling, Feature Engineering & Selection] <i>Model-oriented Stage</i> [Model Design, Model Training, Model Evaluation, Model Optimization] <i>Operations Stage</i> [Model Deployment, Model Monitoring]
Artifact Support	<i>Data-related Artifacts</i> [Dataset, Annotations & Labels, Features] <i>Model Artifacts</i> [Model] <i>Metadata Artifacts</i> [Identification, Model Parameters, Model Hyperparameters, Model Metrics, Experiments & Projects, Pipelines, Execution Logs & Statistics] <i>Software Artifacts</i> [Source Code, Notebooks, Configurations, Environment]
Operations	<i>Logging & Versioning</i> [Log/Capture, Commit, Revert/Rollback] <i>Exploration</i> [Query, Compare, Lineage, Provenance, Visualize] <i>Management</i> [Modify, Delete, Execute & Run, Deploy] <i>Collaboration</i> [Export & Import, Share]
Collection & Storage	<i>Collection Automation</i> [Intrusive, Non-intrusive] <i>Storage Type</i> [Filesystem, Database, Object/BLOB Storage, Repository] <i>Versioning</i> [Repository, Snapshot]
Interface & Integration	<i>Interface</i> [API/SDK, CLI, Web UI] <i>Language Support & Integration</i> [Languages, Frameworks, Notebook]
Operation & Licensing	<i>Operation</i> [Local, On-premise, Cloud] <i>License</i> [Free, Non-free]

serving). *Collaboration* indicates the presence of export, import, and sharing functionality.

Collection & Storage. This category describes the artifact collection and storage model. The criterion *Collection Automation* represents the degree of manual effort required to collect and capture ML artifacts. The collection of artifacts is intrusive, which requires engineers to explicitly add instructions or API calls within the source code, non-intrusive, which means that no explicit manual actions are required and the collection is performed automatically, or both. *Storage Type* describes the type of storage used and supported by an AMS such as filesystems, databases, object stores, and version-controlled repositories. The criterion *Versioning* characterizes how versioning of artifacts is accomplished, which can be either via a version control system (managed by means of a repository as indicated by the corresponding storage type) or via manually triggered snapshots of individually selected ML artifacts.

Interface & Integration. This category characterizes the provided user interfaces and integration capabilities. The criterion *Interface* states the type of provided interfaces that may be based on an API (e. g. a REST API) or higher-level SDK, command line interface (CLI), and/or web application. *Language Support & Integration* distinguishes between the integration

into programming languages (e. g. into Python via provided libraries), frameworks providing functional integration for the steps of the ML lifecycle (e. g. data processing with Apache Spark (Apache Software Foundation, 2022), model training with TensorFlow (Google, 2022b), or model orchestration with Metaflow (Netflix, Inc., 2022)), and/or notebook support (e. g. Jupyter (Project Jupyter, 2022), Apache Zeppelin (Apache, 2022), or TensorBoard (Google, 2022a)).

Operation & Licensing. The last category covers two usage-related criteria. The criterion *Operation* defines whether the operation of an ML AMS is local (e. g. Python libraries), on-premise (e. g. server-based systems) or by a dedicated cloud provider (e. g. hosted services). *License* describes the type of license, which is either classified as free (e. g. public domain, permissive, or copyleft licenses) or non-free (e. g. non-commercial or proprietary licenses).

3 INTERACTIVE WEB TOOL

This section presents CORNUCOPIA – an interactive web tool for exploring and comparing ML AMSs. Based on an intuitive workflow (see Fig. 2), we explain the functionality and user interface (§ 3.1). Subsequently, we describe the architecture design (§ 3.2) and implementation (§ 3.3).

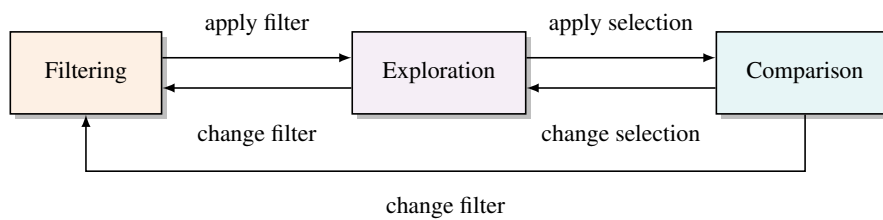


Figure 2: CORNUCOPIA's 3-step workflow.

3.1 Functionality and User Interface

CORNUCOPIA first enables users to filter the set of all ML AMSs according to their requirements and, subsequently, to explore as well as to compare the set of resulting candidates. CORNUCOPIA guides users through a predefined workflow consisting of three steps (see Fig. 2): Filtering, Exploration, and Comparison. These steps are consistently reflected in the web interface.

Filtering. The Filtering step aims at restricting the total set of 64 ML AMSs to only those to be considered further and in more detail for a concrete use case. The applicable filters correspond to the assessment criteria and subcriteria (see also Tab. 1). In addition, a restriction to specific AMSs can also be made here already. Each filter is represented as a drop-down box with a text field and the already selected values (see Fig. 3.(a)). Additional options allow to customize the exploration visualization (e. g. sort selected criteria/systems or show/hide tooltips).

Exploration. In the Exploration step, the ML AMSs and their assessments according to the filter criteria are visualized below in the form of a heatmap (see Fig. 3.(b)). For each ML AMS (x-axis) and each criterion (y-axis), the proportion of fulfilled or present subcriteria is normalized to the value range $[0, 1]$ and represented by a cell hue ranging from dark red (i. e. not fulfilled/present) to dark green (i. e. completely fulfilled/present).³ If the fulfillment/presence of all subcriteria of a criterion is unclear or not (exactly) known, the corresponding cell's hue is white.

That enables a simple yet effective exploration, initial comparison, and further selection of candidates. Clicking on axis tick labels on the y-axis opens a sidebar with explanations of the criteria and their corresponding subcriteria. By clicking on axis tick labels on the x-axis, descriptions and literature references are

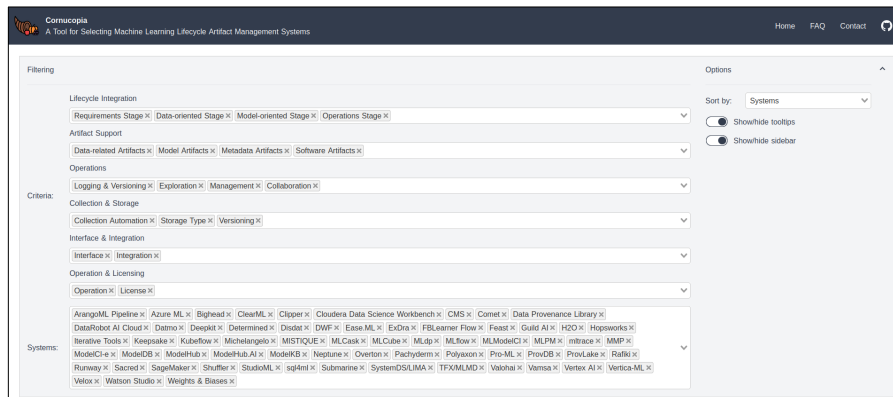
displayed in the sidebar (see also Fig. 3.(b)). Furthermore, clicking on individual cells also opens the sidebar, which then displays detailed information about the respective assessment. Thus, it is possible to obtain a more detailed overview and to trace back the basis on which certain assessments have been made (sidebar). To include either more or less ML AMSs, the filter criteria can be either relaxed or tightened (which results in a transition to the Filtering step).

Comparison. Based on the further refined and restricted candidates set, the Comparison step facilitates a tabular comparison (see Fig. 3.(c)) that lays the foundation for the final selection. The table contains all assessments on subcriteria level and, thus, provides the most fine-grained view. If none of these ML AMSs matches, the selection of candidates can be changed (transition to the Exploration step) or restarted (transition to the Filtering step).

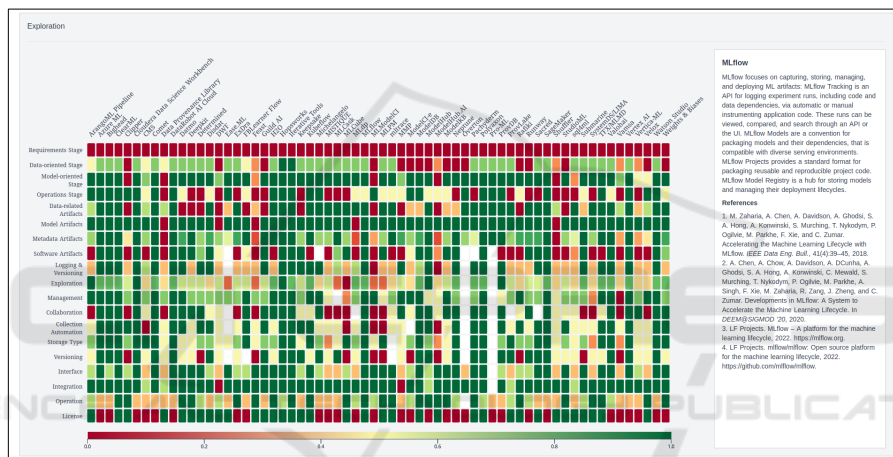
3.2 Architecture Design

CORNUCOPIA's architecture is based on the Model-View-ViewModel (MVVM) pattern (Gossman, 2005). Basically, MVVM separates the representation from the logic of the graphical user interface (GUI). The Model represents the data access layer for the content that is displayed to and manipulated by the user. For this purpose, it notifies about data changes and performs validation of the data passed by the user. The View represents all GUI elements. It binds to the properties of the ViewModel to display and manipulate content and pass user input. The ViewModel contains the UI logic and serves as a link between the View and the Model. It exchanges information with the Model via method or service calls. Moreover, it makes public properties and functions available to the View. These are bound by the view to controls in order to output content or forward UI events. Compared to the classical Model-View-Controller pattern, MVVM improves testability and reduces implementation effort, since no separate controller instances are required.

³The last criterion "License" is an exception due to its binary character: The corresponding cell hue is either dark red ("non-free") or dark green ("free").



(a) Filtering step: Filters can be set via drop-down boxes which also show all already selected values. Additionally, options enable customization of the visualization.



(b) Exploration step: The heatmap visualizes the proportion of fulfilled or present subcriteria for each criterion (y-axis) and ML AMS (x-axis) (normalized to [0, 1]). The sidebar displays additional descriptions of criteria and AMSs, and assessment details.

Comparison			CLEARML	HOPSWORKS	MLFLOW	POLYAXON
Lifecycle Integration	Requirements Stage	Model Requirements Analysis	X	X	X	X
		Data Collection	X	X	X	X
	Data-oriented Stage	Data Preparation & Cleaning	✓	✓	✓	✓
		Data Labeling	✓	✓	✓	✓
		Feature Engineering & Selection	✓	✓	✓	✓
		Model Design	✓	✓	✓	✓
	Model-oriented Stage	Model Training	✓	✓	✓	✓
		Model Evaluation	✓	✓	✓	✓
		Model Optimization	✓	✓	✓	✓
		Model Deployment	✓	✓	✓	✓
Operations Stage	Model Monitoring	✓	✓	X	✓	
Artifact Support	Data-related Artifacts	Dataset	✓	✓	✓	✓
		Annotations & Labels	✓	✓	✓	✓
	Model Artifacts	Features	✓	✓	✓	✓
		Model	✓	✓	✓	✓
	Metadata Artifacts	Identification	✓	✓	✓	✓
		Model Parameters	✓	✓	✓	✓

(c) Comparison step: The fine-grained tabular view of subcriteria level assessments enables the final selection of one or more complementary ML AMSs.

Figure 3: Screenshots of CORNUCOPIA’s user interface components.

3.3 Implementation

CORNUCOPIA is realized as a single-page application (SPA) (Mesbah and van Deursen, 2007). An SPA is a JavaScript-driven web application that consists of a single HTML document and whose content is dynamically reloaded in response to user actions. This enables a higher reactivity compared to multi-page web applications (Fink and Flatow, 2014). The MVVM-based software architecture is implemented by using the Nuxt.js framework (NuxtLabs, 2022a; NuxtLabs, 2022b), which is built upon Vue.js (You, Evan, 2022; The Vue.js Authors, 2022). In contrast to vanilla Vue, Nuxt provides a predefined environment and project structure, automatic routing, different rendering modes (such as server-side rendering, client-side rendering, and static site generation), and integration with many other frameworks (for UI, testing, etc.). The D3.js library (Bostock, Mike, 2022; The D3 Authors, 2022) is used for visualizing data using modern web standards and a data-driven approach to DOM manipulation.

4 CONCLUSION

This paper addresses the problem of the overwhelming and opaque landscape of tools, frameworks, and platforms for managing ML lifecycle artifacts. Based on the assessment of 64 ML AMSs, we present CORNUCOPIA – a web tool that enables researchers and practitioners to conveniently explore and compare ML AMSs.⁴ CORNUCOPIA guides users through a three-step workflow comprising the filtering of the assessed ML AMSs, the exploration of a filtered set of ML AMSs, and the comparison of a selection of candidate ML AMSs.

CORNUCOPIA is under ongoing development.⁵ Next, on-demand calculated statistics and drag-and-drop features (such as for reordering of rows and/or columns) are added. Moreover, many of the assessed AMSs are also under continuous development. Thus, the data basis of CORNUCOPIA reflects the current state of the system landscape. To ensure that our tool will remain valuable for the community in the future, we will publish the source code on GitHub and invite the submission of pull requests.

⁴<https://cornucopia-app.github.io>

⁵<https://github.com/cornucopia-app>

ACKNOWLEDGEMENTS

This work was partially funded by the Thuringian Ministry of Economic Affairs, Science and Digital Society (grant 5575/10-3).

REFERENCES

- Aguilar, L., Dao, D., Gan, S., Gürel, N. M., Hollenstein, N., Jiang, J., Karlas, B., Lemmin, T., Li, T., Li, Y., Rao, X., Rausch, J., Renggli, C., Rimanic, L., Weber, M., Zhang, S., Zhao, Z., Schawinski, K., Wu, W., and Zhang, C. (2021). Ease.ML: A Lifecycle Management System for Machine Learning. In *Proceedings of the 11th Annual Conference on Innovative Data Systems Research, CIDR '21*. www.cidrdb.org.
- Allegro AI (2022). ClearML – MLOps for Data Science Teams. <https://clear.ml>.
- Amershi, S., Begel, A., Bird, C., DeLine, R., Gall, H., Kamar, E., Nagappan, N., Nushi, B., and Zimmermann, T. (2019). Software Engineering for Machine Learning: A Case Study. In *Proceedings of the 2019 IEEE/ACM 41st International Conference on Software Engineering: Software Engineering in Practice, SEIP@ICSE '19*, pages 291–300. IEEE/ACM.
- Apache (2022). Zeppelin. <https://zeppelin.apache.org>.
- Apache Software Foundation (2022). Apache Spark – Unified engine for large-scale data analytics. <https://spark.apache.org>.
- Baylor, D., Breck, E., Cheng, H.-T., Fiedel, N., Foo, C. Y., Haque, Z., Haykal, S., Ispir, M., Jain, V., Koc, L., Koo, C. Y., Lew, L., Mewald, C., Modi, A. N., Polyzotis, N., Ramesh, S., Roy, S., Whang, S. E., Wicke, M., Wilkiewicz, J., Zhang, X., and Zinkevich, M. (2017). TFX: A TensorFlow-Based Production-Scale Machine Learning Platform. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '17*, pages 1387–1395. ACM.
- Bostock, Mike (2022). D3.js – Data-Driven Documents. <https://d3js.org>.
- Chaoji, V., Rastogi, R., and Roy, G. (2016). Machine Learning in the Real World. *Proceedings of the VLDB Endowment*, 9(13):1597–1600.
- Feast Authors (2022). Feast: Feature Store for Machine Learning. <https://feast.dev>.
- Fink, G. and Flatow, I. (2014). *Pro Single Page Application Development*. Apress.
- Gharibi, G., Walunj, V., Rella, S., and Lee, Y. (2019). ModelKB: Towards Automated Management of the Modeling Lifecycle in Deep Learning. In *Proceedings of the 7th International Workshop on Realizing Artificial Intelligence Synergies in Software Engineering, RAISE@ICSE '19*, pages 28–34. IEEE/ACM.
- Google (2022). Machine Learning Workflow. <https://cloud.google.com/ai-platform/docs/ml-solutions-overview>.
- Google (2022a). TensorBoard. <https://www.tensorflow.org/tensorboard>.

- Google (2022b). TensorFlow. <https://www.tensorflow.org>.
- Gossman, J. (2005). Introduction to Model/View/ViewModel pattern for building WPF apps. <https://docs.microsoft.com/en-us/archive/blogs/johngossman/introduction-to-modelviewviewmodel-pattern-for-building-wpf-apps>.
- Greff, K., Klein, A., Chovanec, M., Hutter, F., and Schmidhuber, J. (2017). The Sacred Infrastructure for Computational Research. In *Proceedings of the 16th Python in Science Conference*, SciPy '17, pages 49–56.
- Kumar, A., Boehm, M., and Yang, J. (2017). Data Management in Machine Learning: Challenges, Techniques, and Systems. In *Proceedings of the 2017 ACM International Conference on Management of Data*, SIGMOD '17, pages 1717–1722. ACM.
- LogicalClocks (2022). Hopsworks: Enterprise Feature Store & End-to-End ML Pipeline. <https://www.hopsworks.ai>.
- Luo, Z., Yeung, S. H., Zhang, M., Zheng, K., Zhu, L., Chen, G., Fan, F., Lin, Q., Ngiam, K. Y., and Chin Ooi, B. (2021). MLCask: Efficient Management of Component Evolution in Collaborative Data Analytics Pipelines. In *Proceedings of the 2021 IEEE 37th International Conference on Data Engineering*, ICDE '21, pages 1655–1666. IEEE.
- Mesbah, A. and van Deursen, A. (2007). Migrating Multi-page Web Applications to Single-page AJAX Interfaces. In *Proceedings of the 11th European Conference on Software Maintenance and Reengineering, Software Evolution in Complex Software Intensive Systems*, CSMR '07, pages 181–190. IEEE.
- Miao, H., Li, A., Davis, L. S., and Deshpande, A. (2017). Towards Unified Data and Lifecycle Management for Deep Learning. In *Proceedings of the 2017 IEEE 33rd International Conference on Data Engineering*, ICDE '17, pages 571–582. IEEE.
- Netflix, Inc. (2022). Metaflow. <https://metaflow.org>.
- NuxtLabs (2022a). Nuxt – The Intuitive Vue Framework. <https://nuxtjs.org>.
- NuxtLabs (2022b). nuxt/nuxt.js: The Intuitive Vue(2) Framework. <https://github.com/nuxt/nuxt.js/>.
- Polyzotis, N., Roy, S., Whang, S. E., and Zinkevich, M. (2018). Data Lifecycle Challenges in Production Machine Learning: A Survey. *ACM SIGMOD Record*, 47(2):17–28.
- Project Jupyter (2022). Project Jupyter. <https://jupyter.org>.
- Schad, J., Sambasivan, R., and Woodward, C. (2021). Arangopipe, a tool for machine learning meta-data management. *Data Science*, 4(2):85–99.
- Schelter, S., Bießmann, F., Januschowski, T., Salinas, D., Seufert, S., and Szarvas, G. (2018). On Challenges in Machine Learning Model Management. *IEEE Data Engineering Bulletin*, 41(4):5–15.
- Schlegel, M. and Sattler, K.-U. (2022). Management of Machine Learning Lifecycle Artifacts: A Survey. *ACM SIGMOD Record*. To appear.
- Sculley, D., Holt, G., Golovin, D., Davydov, E., Phillips, T., Ebner, D., Chaudhary, V., Young, M., Crespo, J.-F., and Dennison, D. (2015). Hidden Technical Debt in Machine Learning Systems. In *Advances in Neural Information Processing Systems*, NIPS '15, pages 2503–2511. Curran Associates, Inc.
- Smith, J. (2018). *Machine Learning Systems*. Manning Publications. ISBN 9781617293337.
- The D3 Authors (2022). d3/d3: Bring data to life with SVG, Canvas and HTML. <https://github.com/d3/d3>.
- The Vue.js Authors (2022). vuejs/vue: Vue.js is a progressive, incrementally-adoptable JavaScript framework for building UI on the web. <https://github.com/vuejs/vue>.
- Tsay, J., Mummert, T., Bobroff, N., Braz, A., Westerink, P., and Hirzel, M. (2018). Runway: machine learning model experiment management tool. In *Proceedings of the 1st Conference on Systems and Machine Learning*, SysML '18.
- Vartak, M., Subramanyam, H., Lee, W.-E., Viswanathan, S., Husnoo, S., Madden, S., and Zaharia, M. (2016). ModelDB: A System for Machine Learning Model Management. In *Proceedings of the Workshop on Human-In-the-Loop Data Analytics*, HILDA@SIGMOD '16. ACM.
- Weights & Biases (2022). Weights & Biases – Developer Tools for ML. <https://wandb.ai>.
- You, Evan (2022). Vue.js – The Progressive JavaScript Framework. <https://vuejs.org>.
- Zaharia, M., Chen, A., Davidson, A., Ghodsi, A., Hong, S. A., Konwinski, A., Murching, S., Nykodym, T., Ogilvie, P., Parkhe, M., Xie, F., and Zumar, C. (2018). Accelerating the Machine Learning Lifecycle with MLflow. *IEEE Data Engineering Bulletin*, 41(4):39–45.