Model Share AI: An Integrated Toolkit for Collaborative Machine Learning Model Development, Provenance Tracking, and Deployment in Python

Heinrich Peters[1] 🆔 ✉, and **Michael Parrott**[1] ✉

[1]Columbia University

SciPy 2024

July 8 - July 14, 2024

Abstract

Machine learning (ML) is revolutionizing a wide range of research areas and industries, but many ML projects never progress past the proof-of-concept stage. To address this problem, we introduce Model Share AI (AIMS), a platform designed to streamline collaborative model development, model provenance tracking, and model deployment, as well as a host of other functions aiming to maximize the real-world impact of ML research. AIMS features collaborative project spaces and a standardized model evaluation process that ranks model submissions based on their performance on holdout evaluation data, enabling users to run experiments and competitions. In addition, various model metadata are automatically captured to facilitate provenance tracking and allow users to learn from and build on previous submissions. Furthermore, AIMS allows users to deploy ML models built in Scikit-Learn, TensorFlow Keras, or PyTorch into live REST APIs and automatically generated web apps with minimal code. The ability to collaboratively develop and rapidly deploy models, making them accessible to non-technical end-users through automatically generated web apps, ensures that ML projects can transition smoothly from concept to real-world application.

1. INTRODUCTION

Machine learning (ML) is revolutionizing a wide range of research areas and industries, providing data-driven solutions to important societal problems. The success of many ML projects depends on effective collaboration, rigorous evaluation, and the ability to deploy models. Traditionally, researchers and practitioners have been using version-control systems like GitHub in combination with custom model evaluation and benchmarking experiments to ensure reproducibility and to compare models. However, these systems tend to lack easy-to-use, structured pathways specifically designed to collaboratively develop and rapidly deploy ML models. Furthermore, the creation of custom resources for model evaluation, benchmarking, and deployment, can require substantial upfront effort. As a result, most models do not progress past the proof-of-concept stage and are never deployed [1], [2], preventing a wider audience from participating in the promise of applied ML research.

While the recent rise of platforms like Hugging Face Hub [3], TensorFlow Hub [4], and MLflow [5], [6], [7], illustrates the demand for open-source model repositories and MLOps solutions, barriers of entry are still high for researchers, educators, and practitioners from non-technical disciplines. Model Share AI (AIMS) addresses this problem by providing a lightweight, easy-to-use alternative. In a few lines of code, users can create Model Play-

grounds - standardized ML project spaces that offer an all-in-one toolkit for collaborative model improvement, experiment tracking, model metadata analytics, and instant model deployment, allowing researchers to rapidly iterate on ML models in one streamlined workflow. The present paper provides an introduction to AIMS by situating it in relation to other current solutions and outlining its key functions, technical background, and workflow.

2. RELATED WORK

The primary objective of AIMS is to offer an easy pathway to organize ML projects by reducing common and complex tasks down to minimal code. Its approach builds on the work of various projects focused on providing value for common ML tasks (e.g., model improvement, version tracking, deployment, etc.). Currently, there are several open-source tools and platforms, such as Hugging Face Hub [3], TensorFlow Hub [4], MLflow [5], [6], [7], and OpenML [8], [9], [10], providing ML model repositories where researchers can find and download model objects or deploy models. Hugging Face Hub [3] is a platform allowing users to share pre-trained models, datasets, and demos of ML projects. It has GitHub-inspired features for code-sharing and collaboration, such as discussions and pull requests, and it includes a pathway for model deployment into API endpoints. Similarly, TensorFlow Hub [4] is a repository and library for reusable ML in TensorFlow, enabling users to fine-tune and deploy deep learning models. Deployment is facilitated by TensorFlow Serving [11], which allows users to make their models accessible on a server through API endpoints. MLflow [5], [6], [7] is an open-source platform that manages the end-to-end ML lifecycle. It provides experiment tracking, code packaging, a model registry, model serving, and integration with all popular ML frameworks. While Hugging Face Hub, TensorFlow Hub, and MLflow are well suited for large-scale deep learning tasks, OpenML [8], [9], [10], focuses more on classic ML and model reproducibility. Here, researchers can share, explore, and experiment with ML models, datasets, and workflows, but there is currently no option for model deployment. The OpenML API provides access through various programming languages, but users can also employ an OpenML web interface to browse and visualize data, models, and experiments.

Hugging Face Hub and TensorFlow Hub are primarily *model-focused* platforms, providing repositories of pre-trained models that users can fine-tune for their specific purposes. OpenML and MLflow, on the other hand, are more *task-focused*, emphasizing model evaluation and benchmarking on standardized tasks. While all of these platforms are extensively used by ML researchers and practitioners, including for industry-scale projects, their wide range of features and customizations can be overwhelming for researchers from non-technical disciplines, students, and educators. In comparison, AIMS stands out by prioritizing ease of use and a hyper-collaborative approach. Unlike Hugging Face Hub and TensorFlow Hub, AIMS emphasizes model metadata analytics and task-focused collaboration. Compared to MLflow, it offers more community-based features like competitions that promote collective problem-solving and crowd-sourcing. Moreover, while OpenML excels in model evaluation and benchmarking, AIMS provides additional capabilities for model deployment, allowing users to share models directly from their local environment into live REST APIs and auto-generated web applications. Taken together, the key distinctions of AIMS are its beginner-friendly design and its strong focus on collaborative model development, as well as its goal of providing value not just for ML researchers but also for researchers, practitioners, and educators from non-technical disciplines.

3. MODEL SHARE AI

AIMS provides standardized ML project spaces (Model Playgrounds; see Figure 1) with accessible MLOps features designed to support collaborative model development, model
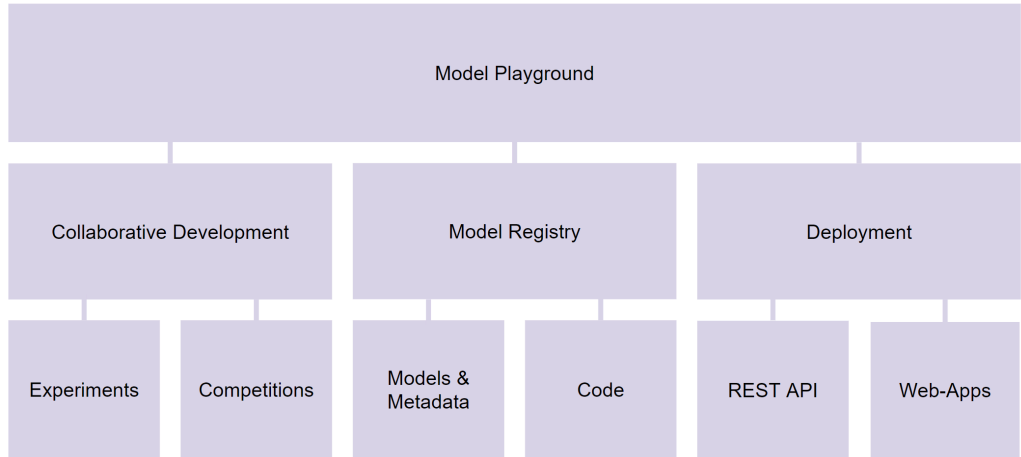
Figure 1. *A Model Playground is a standardized ML project space, representing a ML task that is associated with a specific dataset and a task type (classification, regression). A Model Playground includes Experiments and Competitions for collaborative model development, a model registry with model metadata and user-generated code, as well as deployment functionalities including a live REST API and pre-built web-apps.*

metadata analytics, and model deployment, as well as a host of other functions aiming to maximize the real-world impact of ML research. As such, it simplifies, combines, and extends the capabilities of current solutions in 4 important ways:

1. Collaborative model development: AIMS facilitates collaborative model development and crowd-sourcing through standardized model evaluation procedures, experiment tracking, and competitions.
2. Model registry: AIMS hosts model objects, as well as model metadata, automatically extracted from each model submission, enabling users to analyze which modeling decisions lead to high performance on specific tasks.
3. Model deployment: Model deployment is simplified considerably, allowing users to share their models into live REST APIs and pre-built web apps directly from their Python training environments with just a few lines of code.
4. AIMS provides a wide range of supporting functionalities, including workflows for reproducibility, data sharing, code sharing, creating ML project portfolios, and more.

3.1. Key Functions

3.1.1. *Collaborative Model Development:*

A key feature of AIMS is its focus on collaborative model development and crowd-sourced model improvement, enabling teams to iterate quickly by allowing collaborators to build on each other's progress, even across libraries. For supervised learning tasks, users can collaboratively submit models into Experiments or Competitions associated with a Model Playground project in order to track model performance and rank submissions in standardized leaderboards according to their evaluation metric of choice. Experiments and Competitions are set up by providing holdout evaluation data against which the predictions of submitted models are evaluated. Standardized model evaluations allow collaborators to track the performance of their models along with a wide range of model metadata that are automatically extracted from submitted models and added to the model registry (see section below). Out of the box, AIMS calculates accuracy, f1-score, precision, and recall for classification tasks, and mean squared error, root mean squared error, mean absolute error, and R^2-scores for regression tasks. The main difference between Experiments and Competitions is that a proportion of the evaluation data is kept secret for Competitions,

preventing participants from deliberately overfitting on evaluation data. Being able to submit models into shared Experiments enables ML teams to standardize tasks, rigorously track their progress, and build on each other's success, while Competitions facilitate crowd-sourced solutions. Both Experiments and Competitions can be either public (any AIMS user can submit) or private (only designated team members can submit). Users can deploy any model from an Experiment or Competition into the REST API associated with their Model Playground with a single line of code.

3.1.2. *Model Registry:*

Model versions are made available for each Model Playground and comprehensive model metadata are automatically extracted for each submitted model. In addition to evaluation metrics, this includes hyperparameter settings for Scikit-Learn models and model architecture data (such as layer types and dimensions, number of parameters, optimizers, loss function, memory size) for Keras and Pytorch models. Users can also submit any additional metadata they choose to capture. Model metadata are integrated into Competition and Experiment leaderboards, enabling users to analyze which types of models tend to perform well for a specific ML task. Users can either visually explore leaderboards on their Model Playground page or they can download leaderboards into Pandas data frames to run their own analyses. There is also a set of AIMS methods designed to visualize model metadata. For example, models can be compared in a color-coded layout showing differences in model architectures and hyperparameter settings. Furthermore, users can instantiate models from the AIMS model registry into reproducible environments. Taken together, these functions are designed to streamline the management, collaboration, and deployment of ML models, enhancing their discoverability, reproducibility, and traceability throughout their lifecycle.

3.1.3. *Instant Model Deployment:*

AIMS currently allows users to deploy ML models built in Scikit-Learn [12], Tensorflow Keras [13], [14], and Pytorch [15] into live REST APIs rapidly with minimal code. Additionally, users can deploy models from other ML frameworks by transforming them into ONNX (Open Neural Network Exchange) format - an open-source format for standardized representations of ML models with the goal of making them interoperable across platforms. Each deployed model is associated with a Model Playground page on the AIMS website and a REST API endpoint hosted in a serverless AWS backend. End-users can either manually upload data to make predictions using an automatically generated web app on the Model Playground Page, or they can programmatically query the REST API associated with the model. In addition to auto-generated web apps, AIMS enables users to submit their own Streamlit apps. Out of the box, AIMS supports models built on tabular, text, image, audio, and video data. Allowing users to deploy models with minimal effort and making those models accessible to even non-technical end-users through web apps holds the promise of making ML research applicable to real-world challenges.
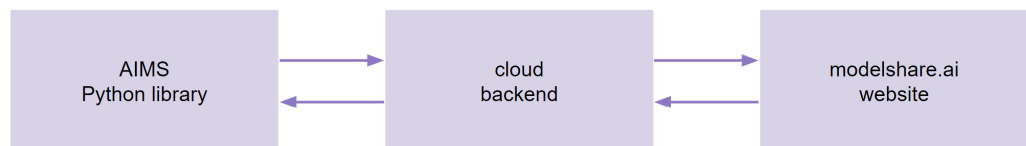

Figure 2. *Overview of the AIMS architecture. The AIMS Python library allows users to create Model Playground pages, submit and deploy models, and analyze model metadata. The modelshare.ai website provides a graphical user interface to explore model metadata and generate predictions via auto-generated web apps. All required resources are automatically generated in scalable serverless cloud infrastructure.*

3.2. Architecture

AIMS consists of three main components: an open-source Python library, user-owned cloud backend resources, and the AIMS website (see Figure 2). The AIMS Python library is the main interface allowing users to set up Model Playground pages (including Experiments and Competitions), submit and deploy models, analyze model metadata, and reproduce model artifacts. It provides an accessible layer that facilitates the creation of the cloud backend resources that power REST APIs, as well as model evaluations and model metadata extraction. The *ModelPlayground()* class acts as a local representation of a Model Playground page and its associated REST API. It provides a range of methods to configure, change, and query Model Playground resources. A detailed overview of the Python library is provided below (AIMS Workflow).

The cloud backend hosts model objects and associated artifacts in S3 storage, while REST APIs are deployed into serverless lambda functions. Lambda functions are programs or scripts that run on high-availability AWS compute infrastructure. They can be invoked by various event sources (e.g., API calls) and scale automatically based on the volume of incoming requests. This means that users do not have to maintain servers, and they only pay for the time and resources actually consumed, but not for idle time. The AIMS Python library allows users to automatically generate and deploy lambda functions based on specific properties of their ML tasks, without the need to explicitly manage any AWS resources. The most important lambda functions in the context of AIMS are the Evaluation Lambda, which computes evaluation metrics and extracts model metadata from submitted models, and the Main Lambda, which computes predictions on data submitted through the REST API. Runtime models are automatically packaged into Docker containers that run on lambda. Additionally, certain metadata are stored in a centralized Redis database that powers the modelshare.ai website.

The AIMS website hosts user profile pages, model pages, web apps, example code, and a documentation page, as well as user-generated code and documentation for their specific ML projects (see Supplementary Information A-E).

3.3. AIMS Workflow

The AIMS workflow is designed to help teams collaboratively and continuously train, evaluate, improve, select, and deploy models using standardized ML project spaces or Model Playgrounds. After training a model, users submit their model to a Competition or Experiment associated with a Model Playground. The model is then automatically evaluated, and model metadata are extracted. Evaluations and metadata are made available via a graphical user interface on the Model Playground page or can be queried using the AIMS Python library, enabling users to analyze which types of models perform well on a given learning task. This information can then be used to either improve the training process and submit more models or to select a model and instantly deploy it with a single line of code. Deployed models can easily be swapped out if they degrade over time or if they are outperformed by new submissions. An overview of the process can be found in Figure 3.

In order to use AIMS, users first need to create an AIMS user profile and generate credentials through the AIMS website. Additionally, users are required to generate AWS credentials if they wish to deploy models into their own AWS resources.

The AIMS workflow is centered around the concept of Model Playgrounds. A Model Playground is a standardized ML project space, representing an ML project that is associated with a specific dataset and a task type, such as classification or regression. A Model Playground is first instantiated locally using the *ModelPlayground()* class of the AIMS Python library. Here, users need to specify the *input_type* (tabular, text, image, audio, etc.) and
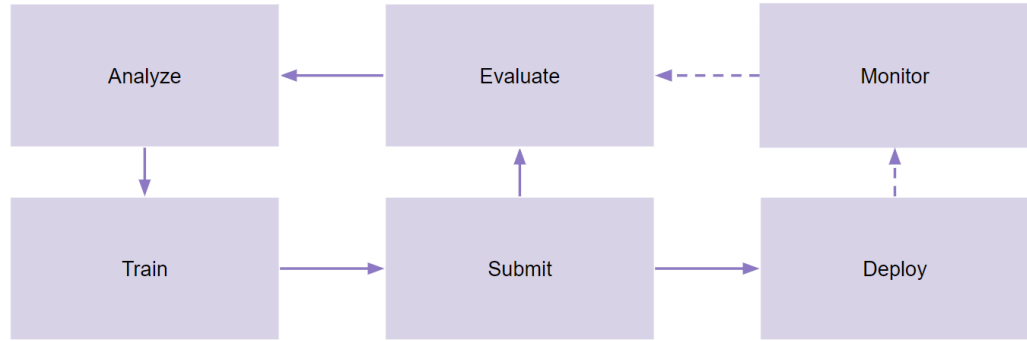
Figure 3. *MLOps workflow with AIMS. Users iteratively train, submit, evaluate, and analyze their models. As contributors have access to model evaluations, model architecture metadata, and reproducible model objects of previous submissions, they can rapidly develop high-performing models. Submitted models can easily be deployed into live REST APIs. Deployed runtime models can be monitored and seamlessly be swapped out against newly submitted models.*

the *task_type* (classification, regression), as well as select whether their Model Playground should be public or private. Private Model Playgrounds can only be accessed by invited collaborators, whereas public Model Playgrounds are open to all AIMS users. After instantiating their Model Playground object, users can create an online Model Playground page by calling the *create()* method and submitting their evaluation data. This generates a fully functioning Model Playground Page on the AIMS website, including a placeholder REST API, and enables users to submit models into associated Experiments and Competitions.

Once a Model Playground Page is created, users can start submitting models to Experiments or Competitions and deploy models into the Model Playground's REST API. A model submission includes a model object (Scikit-Learn, Keras, PyTorch, ONNX), a preprocessor function, and a set of predicted values corresponding to the previously submitted evaluation data. The predictions are then evaluated against the evaluation data, and model metadata are automatically extracted from model objects. Users can submit additional metadata in dictionary format using the *custom_metadata* argument of the *submit_model()* method. After submitting one or more models, users can explore evaluations and model metadata on the associated Model Playground page or query this information for further analysis using the *get_leaderboard()* and *compare_models()* methods. To inspect and improve models, users can instantiate models from the leaderboard using the *instantiate_model()* method. Alternatively, users can instantly deploy a model using the *deploy_model()* method by referring to the model's leaderboard version number. Additionally, users should submit example data, which will help end-users format their input data correctly, and y training data to help the web app and prediction API transform raw model outputs into the correct labels. As mentioned above, the process does not stop when a model is deployed. Users can submit and analyze more models, informed by previous submissions, and easily monitor and swap out the runtime model using the *update_runtime_model()* method. An overview of the model deployment process, including code, is available in Figure 4. Detailed tutorials and documentation can be found in Supplementary Information A.

4. IMPACT

Collaborative model improvement and crowd-sourcing are important, not only because they make teams more efficient but also because they foster diverse perspectives. Such collective efforts can contribute to the democratization of ML, provide access to resources for a wider audience, and enable community-driven innovation. For instance, the AIMS Competition feature can facilitate crowd-sourced research projects in various disciplines utilizing the common task framework. Relatedly, the AIMS model registry promotes discov-
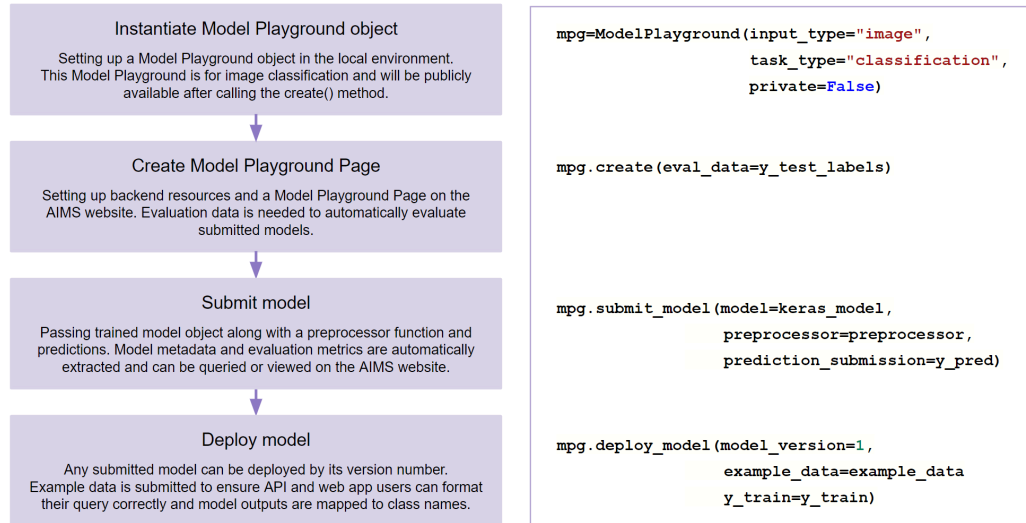
Instantiate Model Playground object	
| Setting up a Model Playground object in the local environment. This Model Playground is for image classification and will be publicly available after calling the create() method. | ```
mpg=ModelPlayground(input_type="image",
                    task_type="classification",
                    private=False)
``` |
| **Create Model Playground Page** | |
| Setting up backend resources and a Model Playground Page on the AIMS website. Evaluation data is needed to automatically evaluate submitted models. | ```
mpg.create(eval_data=y_test_labels)
``` |
| **Submit model** | |
| Passing trained model object along with a preprocessor function and predictions. Model metadata and evaluation metrics are automatically extracted and can be queried or viewed on the AIMS website. | ```
mpg.submit_model(model=keras_model,
                 preprocessor=preprocessor,
                 prediction_submission=y_pred)
``` |
| **Deploy model** | |
| Any submitted model can be deployed by its version number. Example data is submitted to ensure API and web app users can format their query correctly and model outputs are mapped to class names. | ```
mpg.deploy_model(model_version=1,
                 example_data=example_data
                 y_train=y_train)
``` |

Figure 4. *Model deployment process: Models can be deployed with just a few lines of code. After instantiating a local Model Playground object, users can create a Model Playground page that is ready for model submissions. Submitted models are automatically evaluated and can be deployed into live REST APIs.*

erability, transparency, and reproducibility by providing a centralized platform for users to find models and their associated metadata. Sharing model metadata, such as model architectures, hyperparameters, training data, and evaluation metrics, allows researchers to verify previous results and build upon each other's work. The repository also acts as an educational resource, offering students, educators, and self-learners the opportunity to study various ML models, techniques, and best practices. For example, AIMS has thus far been extensively used in classrooms at Columbia University and beyond to organize challenges, collaboratively work on ML projects, and make models accessible through API endpoints. In addition, AIMS is positioned to realize several important missions with regard to model deployment. Firstly, simplifying model deployment is important because it lowers barriers to entry, allowing more developers, researchers, and organizations to incorporate ML in their projects. Secondly, easier deployment saves resources and time, so that developers can dedicate more effort to model training, tuning, and evaluation. Ultimately, a streamlined deployment process allows users to iterate faster and explore novel ideas in real-world contexts.

5. FUTURE WORK

While AIMS provides a simple approach to collaborative model development and deployment, there are opportunities for further improvements. Firstly, it is important to strike the right balance between flexibility and ease of use. A highly flexible platform may allow researchers to use various ML frameworks and libraries, accommodate a wide range of data sources and formats, and support custom deployment strategies. However, this flexibility may come at the cost of increased complexity and a harder learning curve for users. On the other hand, a more user-friendly solution might provide a simpler, more streamlined interface but may lack the flexibility to accommodate unique or complex requirements. By default, AIMS prioritizes standardization and ease of use, making it attractive for researchers, educators, and data scientists who are interested in quick and lightweight solutions. For users who want more flexibility, AIMS provides customizations, such as custom AWS lambda functions, custom containers, and custom metadata submissions. We will continue to extend the functionality of AIMS to make it useful for a wide range of users and applications. This includes making the platform compatible with more advanced model

types, task types, and ML frameworks (e.g., PySpark). Relatedly, future work could include additional MLOps functionality, including improved pathways for monitoring, model explainability, continuous retraining, and automated ML (AutoML) [16], [17]. The latter point is of special interest, as each new model submission contributes to a growing repository of evaluated, reusable ML models, including rich model metadata. These resources can be utilized to suggest pre-trained models that are expected to work well for a given task or enable users to run their own analyses to choose pre-trained models. We hope that AIMS will become a resource for researchers interested in meta-learning [18], [19] and related problems. Additionally, we expect AIMS to be used increasingly by researchers from various disciplines, including social sciences and natural sciences, trying to solve substantive questions through ML. We are planning to further accommodate their diverse needs in order to promote cross-fertilization and widespread participation in the promise of applied ML research.

6. Conclusion

AIMS provides a versatile yet easy-to-use approach to collaborative model development, model metadata analytics, and model deployment. It enables users to quickly find, analyze, and collaboratively improve trained ML models, and to share models from their local environment directly into live REST APIs and pre-built web applications in a single tightly integrated workflow. Compared to existing solutions, AIMS is intended to lower the barriers of entry to ML research, making it attractive to a large group of researchers, educators, and data scientists. We are confident that AIMS will become a widely used tool and maximize the real-world impact of ML research.

References

[1] T. H. Davenport and D. J. Patil, "Is Data Scientist Still the Sexiest Job of the 21st Century?," *Harvard Business Review*, Jul. 2022, [Online]. Available: https://hbr.org/2022/07/is-data-scientist-still-the-sexiest-job-of-the-21st-century

[2] E. Siegel, "Models Are Rarely Deployed: An Industry-wide Failure in Machine Learning Leadership." [Online]. Available: https://www.kdnuggets.com/models-are-rarely-deployed-an-industry-wide-failure-in-machine-learning-leadership.html

[3] "Hugging Face Hub." [Online]. Available: https://huggingface.co/docs/hub/index

[4] "TensorFlow Hub." [Online]. Available: https://www.tensorflow.org/hub

[5] "MLflow." [Online]. Available: https://mlflow.org/

[6] A. Chen *et al.*, "Developments in MLflow: A System to Accelerate the Machine Learning Lifecycle," in *Proceedings of the Fourth International Workshop on Data Management for End-to-End Machine Learning*, in DEEM'20. New York, NY, USA, Jun. 2020, pp. 1–4. doi: 10.1145/3399579.3399867.

[7] M. Zaharia *et al.*, "Accelerating the Machine Learning Lifecycle with MLflow," *IEEE Data Eng. Bull.*, 2018, [Online]. Available: https://www.semanticscholar.org/paper/Accelerating-the-Machine-Learning-Lifecycle-with-Zaharia-Chen/b2e0b79e6f180af2e0e559f2b1faba66b2bd578a

[8] M. Feurer *et al.*, "Openml-python: an extensible python api for openml," *The Journal of Machine Learning Research*, vol. 22, no. 1, pp. 4573–4577, 2021.

[9] J. Vanschoren, J. N. van Rijn, B. Bischl, and L. Torgo, "OpenML: networked science in machine learning," *ACM SIGKDD Explorations Newsletter*, vol. 15, no. 2, pp. 49–60, Jun. 2014, doi: 10.1145/2641190.2641198.

[10] J. N. van Rijn *et al.*, "OpenML: A Collaborative Science Platform," in *Machine Learning and Knowledge Discovery in Databases*, H. Blockeel, K. Kersting, S. Nijssen, and F. Zelezný, Eds., in Lecture Notes in Computer Science. Berlin, Heidelberg, 2013, pp. 645–649. doi: 10.1007/978-3-642-40994-3_46.

[11] C. Olston *et al.*, "TensorFlow-Serving: Flexible, High-Performance ML Serving." [Online]. Available: http://arxiv.org/abs/1712.06139

[12] F. Pedregosa *et al.*, "Scikit-learn: Machine Learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011, [Online]. Available: http://jmlr.csail.mit.edu/papers/v12/pedregosa11a.html

[13] M. Abadi *et al.*, "TensorFlow: A system for large-scale machine learning," 2016, doi: https://doi.org/10.48550/arXiv.1605.08695.

[14] F. Chollet, "Keras: the Python deep learning API." [Online]. Available: https://keras.io/

[15] A. Paszke *et al.*, "PyTorch: An Imperative Style, High-Performance Deep Learning Library." [Online]. Available: http://arxiv.org/abs/1912.01703

[16] F. Hutter, L. Kotthoff, and J. Vanschoren, Eds., *Automated Machine Learning: Methods, Systems, Challenges*. in The Springer Series on Challenges in Machine Learning. Springer International Publishing, 2019. doi: https://doi.org/10.1007/978-3-030-05318-5.

[17] X. He, K. Zhao, and X. Chu, "AutoML: A Survey of the State-of-the-Art," *Knowledge-Based Systems*, vol. 212, p. 106622, 2021, doi: 10.1016/j.knosys.2020.106622.

[18] T. Hospedales, A. Antoniou, P. Micaelli, and A. Storkey, "Meta-Learning in Neural Networks: A Survey." [Online]. Available: http://arxiv.org/abs/2004.05439

[19] C. Finn, P. Abbeel, and S. Levine, "Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks." [Online]. Available: http://arxiv.org/abs/1703.03400