ModelDB: Opportunities and Challenges in Managing Machine Learning Models

Manasi Vartak
MIT CSAIL
mvartak@csail.mit.edu

Samuel Madden
MIT CSAIL
madden@csail.mit.edu

Abstract

Machine learning applications have become ubiquitous in a variety of domains. Powering each of these ML applications are one or more machine learning models that are used to make key decisions or compute key quantities. The life-cycle of an ML model starts with data processing, going on to feature engineering, model experimentation, deployment, and maintenance. We call the process of tracking a model across all phases of its life-cycle as **model management**. *In this paper, we discuss the current need for model management and describe* ModelDB, *the first open-source model management system developed at MIT. We also discuss the changing landscape and growing challenges and opportunities in managing models.*

1 Introduction

Machine learning (ML) has become ubiquitous in a variety of applications including voice assistants, self-driving cars, and recommendation systems. Each ML-based application employs one or more machine learning models that are used to make key decisions or compute key quantities such as recognizing spoken words, detecting a pedestrian on the road, and identifying the best products for customers. Models are to ML-based applications what databases are to stateful web-applications; they are crucial for the correct functioning of these applications. Consequently, just as database management systems (or DBMSs) are used to manage state in applications, we find the need for systems to manage models in ML applications, i.e., *model management systems*.

To understand the requirements of a model management system, we begin with a brief overview of the life-cycle of a machine learning model. We divide the ML life-cycle into five phases, namely: (1) *Data Preparation* - Obtaining the training and test data to develop a model; (2) *Feature Engineering* - Identifying or creating the appropriate descriptors from the input data (i.e., features) to be used by the model; (3) *Model Training and Experimentation* - Experimenting with different models on the training and test data and choosing the best; (4) *Deployment* - Deploying the chosen model in a live system; and (5) *Maintenance* - Monitoring the live model performance, updating the model as needed, and eventually retiring the model. While we describe these phases as occurring in a linear sequence, the empirical nature of building an ML model causes multiple phases to be revisited frequently. We define a model management system as one that follows a model throughout the five phases of its life-cycle and captures relevant metadata at each step to enable model tracking, reproducibility, collaboration, and governance.

Bulletin of the IEEE Computer Society Technical Committee on Data Engineering

For instance, since the key requirement during the Experimentation phase is to enable the data scientists to choose the best model, metadata captured in this phase includes items such as performance metrics for the model, hyperparameter values used during training, etc. In contrast, for a model in the Deployment phase, metadata might include the version of the model, where it is deployed, and how to query it. Although one can imagine a model management system that directly stores models and supports prediction serving operations, the heterogeneity in models, as well as their hardware and software requirements make such a solution sub-optimal. Therefore, we take the view that model management systems are best suited to store *metadata* about models throughout their life-cycle. Thus, **we define a model management system as a system that tracks metadata about models through the five phases of their life-cycle**.

Given the rapid proliferation of machine learning applications, systems have been proposed in academia as well as industry to address different aspects of the model management problem. In this paper, we focus on MODELDB, the first open-source system we developed at MIT for model management. Other academic systems that seek to address similar problems include the ModelHub system [18] (to explore deep learning architectures and efficiently store network weights), ProvDB [19] (to manage metadata collected via collaborative data science), Ground [14] (to provide a common framework for tracking data origin and use via generic abstractions), and the work on model selection management systems by Kumar et. al. [15].

An example of a commercial model management system is the SAS Model Manager which tracks models built and deployed within the SAS platform [24]. Today, most proprietary ML platforms such as the Michelangelo platform at Uber [11] and FBLearner at Facebook [6] include a model management or model repository component. Similarly, vast majority of data science teams end up re-building a model management system to suit their needs. In [25], Sculley et. al. elegantly present the challenges with building and productionizing ML at Google and highlight the need to manage "pipeline jungles" and model configurations.

Model management systems are also closely related to workflow and experiment management systems such as Kepler [16], Taverna workbench [29], Galaxy [27], VisTrails [1, 3, 2] as well as recently introduced workflow engines tailored for data processing such as Apache Airflow [11] and Luigi [26]. While workflow systems can address some of the model management needs during the first three phases of the model life-cycle, these systems require extensions for ML-specific abstractions.

The rest of the paper is organized as follows. In Section 2, we describe the motivation behind model management systems; in Section 3, we describe the challenges faced in building model management systems; and in Section 4, we describe the MODELDB system developed at MIT. We conclude the paper in Section 5 with a discussion of how we see model management evolving in the future.

2 Why Model Management?

To understand the need for model management, we studied modeling workflows across companies and research labs ranging from large internet companies to small start-ups building ML-based applications. Across the different phases of the model life-cycle, the need for model management was apparent in three key areas: managing modeling experiments, enabling reproducibility, supporting sharing of models, and model governance.

The first and primary area where the need for model management is evident is during the Experimentation phase of the model life-cycle. The empirical nature of ML model development means that data scientists and ML developers experiment with hundreds of models before identifying one that meets some acceptance criteria. This is particularly true when data scientists perform hyperparameter optimization on models. Data about previously built models is necessary to inform the next set of models and experiments, and to identify the best model produced so far. Consequently, tracking these experiments is of paramount importance. The absence of an experiment tracking system leads to models and experiments being lost and valuable time and resources being spent in reproducing model results. For example, one ML developer at a large tech company related how she had spent over a week merely re-running a modeling experiment another employee had previously conducted since the experimental setup and results were not recorded.

The second area that requires model management is enabling model reproducibility. A key use case for reproducibility deals with the situation where the new version of a deployed model performs poorly compared to the previous one and must be reverted to the older version. If the previous model version is not available, re-creating it requires precise information about what data was used, how it was processed, libraries and versions used, and details of how the model was trained (including random seeds). A similar need becomes evident when a model has to be updated with new data or when a discrepancy is detected in an offline and live model.

The third area that requires model management is supporting model sharing. As data science teams increase in size, multiple data scientists want to collaborate on the same model or build on top of each others' work. In such cases, the lack of a centralized repository of models along with the right access control patterns hampers sharing of information about models and experiments. Finally, for software developers who wish to use existing ML models, the absence of a central repository makes it challenging to discover and integrate models into products and business processes.

The last area where we find a growing need for model management relates to model governance. For many regulated industries, (e.g., banking and healthcare), government regulations require that any model used to make automated decisions be documented and available for audit. Furthermore, government legislation now being implemented (e.g., the GDPR regulations in the European Union [21]) requires that companies be able to explain any decisions made without human intervention. These trends highlight the need for a centralized system that manages and explains models.

3 Challenges in Model Management

As defined above, model management covers the entire life-cycle of models starting with Data Processing, Feature Engineering, Experimentation, Deployment, and Maintenance. The hallmark of machine learning model development is the heterogenity in environments and frameworks used at every phase of the ML life-cycle. As a result, the primary challenge for model management is to consistently capture metadata across many environments at each phase of the modeling life-cycle. We illustrate this challenge with a few examples.

During the Data Processing or Feature Engineering phases, our goal is to track the transformations applied to data so that they may be accurately reproduced later. Since these tasks may be done in very different languages and data processing environments (e.g., Spark, Teradata, HBase), we must ensure that data transformations can be accurately recorded across a wide range of data processing environments. Moreover, even within a single environment, we must ensure high coverage for all the ways in which a specific data transformation may be applied — a significant challenge unless this functionality has been built into the system from the ground-up.

Similarly, during the Experimentation phase, there is a large diversity in machine learning frameworks and libraries in different programming languages that a data scientist may use (e.g., scikit-learn, Tensorflow, PyTorch in Python; ML libraries in R; H2O framework in Java). Each library has a unique way of defining models (e.g., graphs in Tensorflow vs. plain objects in scikit-learn) and their associated attributes. Consequently, one representation is often inadequate to capture models built across all frameworks. This diversity in machine learning frameworks during Experimentation also translates to diversity in deployment methods during the Deployment phase. For example, while TF-serving is a popular means to serve Tensorflow models, a Flask-based deployment method is most popular for scikit-learn models.

Lastly, once a model is deployed, i.e. during the Maintenance phase, while some properties are common to all models (e.g., latency of predictions, number of requests), many properties are application dependent (e.g., accuracy of model for new users on an e-commerce site) and might require input from disparate systems outside of the ML environment (e.g., prediction storage or logging systems). Thus we see that addressing the heterogeneity at every phase of the modeling life-cycle is the key challenge for model management.

While the above challenges arise from the diversity of environments for ML, two overarching requirements

```
# version 61
#   Drop fireplacecnt and fireplaceflag, following Jayaraman:
#     https://www.kaggle.com/valadi/xgb-w-o-outliers-lgb-with-outliers-combo-tune5

# version 60
#   Try BASELINE_PRED=0.0115, since that's the actual baseline from
#     https://www.kaggle.com/aharless/oleg-s-original-better-baseline

# version 59
#   Looks like 0.0056 is the optimum BASELINE_WEIGHT

# versions 57, 58
#   Playing with BASELINE_WEIGHT parameter:
#     3 values will determine quadratic approximation of optimum

...

# version 49
#   My latest quadratic approximation is concave, so I'm just taking
#     a shot in the dark with lgb_weight=.3

# version 45
#   Increase lgb_weight to 0.25 based on new quadratic approximation.
#   Based on scores for versions 41, 43, and 44, the optimum is 0.261
#     if I've done the calculations right.
#   I'm being conservative and only going 2/3 of the way there.
#   (FWIW my best guess is that even this will get a worse score,
#    but you gotta pay some attention to the math.)

# version 44
#   Increase lgb_weight to 0.23, per Nikunj's suggestion, even though
#     my quadratic approximation said I was already at the optimum
```

Figure 1: Model Versioning comments by Kaggle competitor

emerge from the data scientists' perspective as well. First, data scientists want a model management solution that minimizes developer intervention and requires minimal changes to the current developer workflow. For example, many data scientists are resistant to choosing a new ML environment or significantly changing their modeling workflow to account for model management. And second, data scientists want a vendor and framework-neutral model management system so that they are not tied into one particular provider or framework and can choose the best solutions for the particular modeling task.

To summarize, in order to build a model management system, we must address the problem of supporting a variety of ML environments, imposing common abstractions across diverse environments, and capturing sufficient metadata while requiring minimal changes from data scientists. In the next section, we describe MODELDB, an open-source model management system developed at MIT that takes the first steps in tackling the challenges identified above and focuses particularly on the Experimentation phase of the model life-cycle.

4 MODELDB

Building an ML model for real-world applications is an iterative process. Data scientists and ML developers experiment with tens to hundreds of models before identifying one that meets some acceptance criteria on model performance. For example, top competitors in the Kaggle competition to predict Zillow Home prices [5] made more than 250 submissions (and therefore built at least as many models), while those in the Toxic Comment classification competition [4] made over 400 submissions. As an example of actual experimentation performed during model building, Figure 1 reproduces code comments by an expert Kaggle competitor (ranked in top 500) written in order to track models built for the Zillow Price Prediction Challenge. As we can see from this listing,

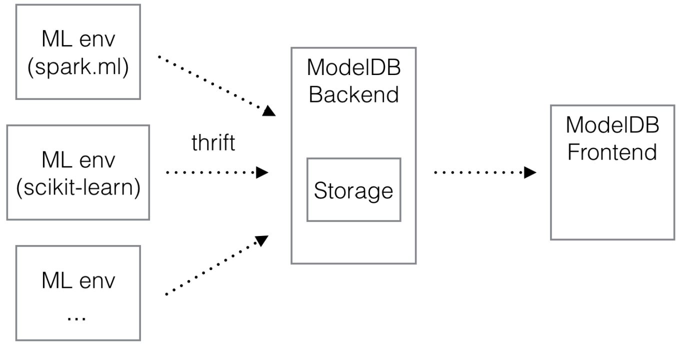

Figure 2: ModelDB Architecture

a data scientist typically tests a large number of model versions before identifying the best one. Moreover, although data scientists (and data science teams) build many tens to hundreds of models when developing an ML application, they currently have no way to keep track of all the models they have built. Consequently, insights are lost, models cannot be reproduced or shared, and model governance becomes challenging.

To address these problems, we developed a system at MIT called MODELDB [28]. MODELDB is the first open-source machine learning model management system and currently focuses on tracking models during the Experimentation phase. MODELDB automatically tracks models as they are built, records provenance information for each step in the pipeline used to generate the model, stores this data in a standard format, and makes it available for querying via an API and a visual interface.

4.1 Architecture

Figure 2 shows the high-level architecture of our system. MODELDB consists of three key components: client libraries for different machine learning environments, a backend that stores model data, and a web-based visualization interface. Client libraries are responsible for automatically extracting models and pipelines from code and passing them to the MODELDB backend. The MODELDB backend exposes a thrift[1] interface to allow clients in different languages to communicate with the MODELDB backend. MODELDB client libraries are currently available for scikit-learn and spark.ml, along with a Light Python API that can be used in any Python-based machine learning environment. This means that ML developers can continue to build models and perform experimentation in these environments while the native libraries passively capture model metadata. The backend can use a variety of storage systems to store the model metadata. The third component of MODELDB, the visual interface, provides an easy-to-navigate layer on top of the backend storage system that permits visual exploration and analyses of model metadata.

4.2 Client Libraries

Many existing workflow management programs (e.g. VisTrails [3]) require that the user create a workflow in advance, usually by means of a GUI. However, the data scientists we interviewed overwhelmingly concurred that GUIs restricted their flexibility in defining modeling pipelines and iterating over them. Moreover, we found that data scientists were resistant to significant changes in their modeling workflows. Therefore, our primary design constraint while creating the MODELDB client libraries was to minimize any changes the data scientist would need to make both to code and the existing modeling process. To meet this constraint, we chose to make

[1] https://thrift.apache.org/

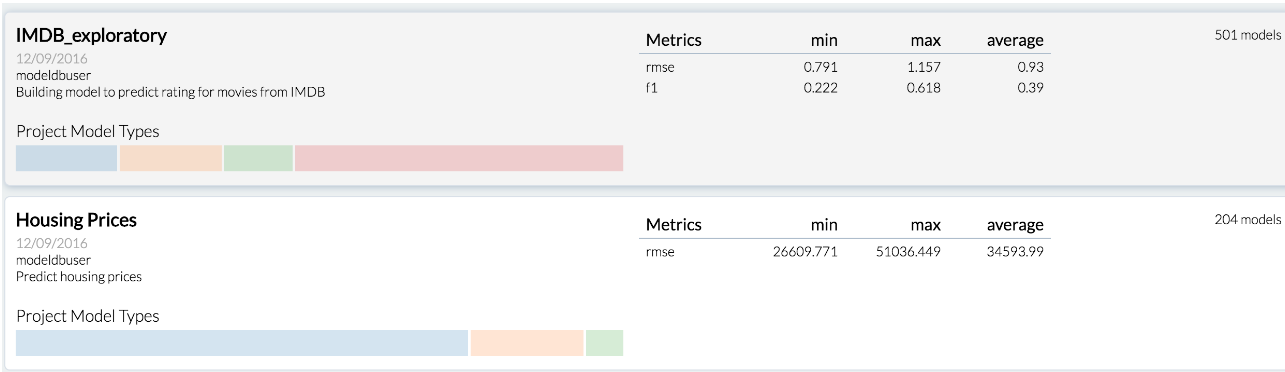

Figure 3: Projects Summary View

model logging accessible directly through code (as opposed to a GUI) and to build logging libraries for different ML environment. The spark.ml and scikit-learn libraries are architected such that data scientists can use these environments for analysis exactly as they normally would and the library transparently and automatically logs the relevant data to the backend.

4.3 Frontend

ModelDB captures a large amount of metadata about models. To support easy access to this data, ModelDB provides a visual interface. We provide three key views for exploring the data stored in ModelDB. A user starts with the projects summary page (Fig. 3) that provides a high level overview of all the projects in the system. The user can then click on a particular project to see the models for that project. We present models via two key views. The first view presents model summaries through two visualizations (Fig. 4): a summary visualization showing the evolution of model metrics over time as well as a custom visualization builder for model meta-analyses. The second is a tabular view of models in a particular project (Fig. 5) along with interactions such as filtering, sorting, grouping, and search. From any of the above interfaces, the ML developer can drill-down into a single model to visualize the model pipeline that was automatically inferred by the client library.

4.4 ModelDB Adoption and Future Work

We officially released ModelDB as an open-source model management system in Feb. 2017 and since then there has been a large amount of interest and adoption of ModelDB. Over the last year, our GitHub repository [12] has garnered $>$ 500 stars, has been cloned over a thousand times, and has been forked $>$100 times. ModelDB has been tested at multiple small and large companies and has been deployed in financial institutions. ModelDB has also served as inspiration for other model management systems such as [20]. The rapid adoption of ModelDB indicates a strong need for model management, and our current work focuses on expanding ModelDB to support other phases of the ML life-cycle.

5 Evolution of Model Management Systems

As machine learning models proliferate into every business process and product, we posit that the task of managing the life-cycle of models will become as key as managing the life-cycle of code. Just as version control systems such as SVN and Git made source code development and collaboration robust and rapid, we envision that model management systems will serve a similar purpose in the future. We imagine model management systems to become the system of record for all models and model-related information. Current model management

Figure 4: Model Summary Visualization and Custom Visualization Builder

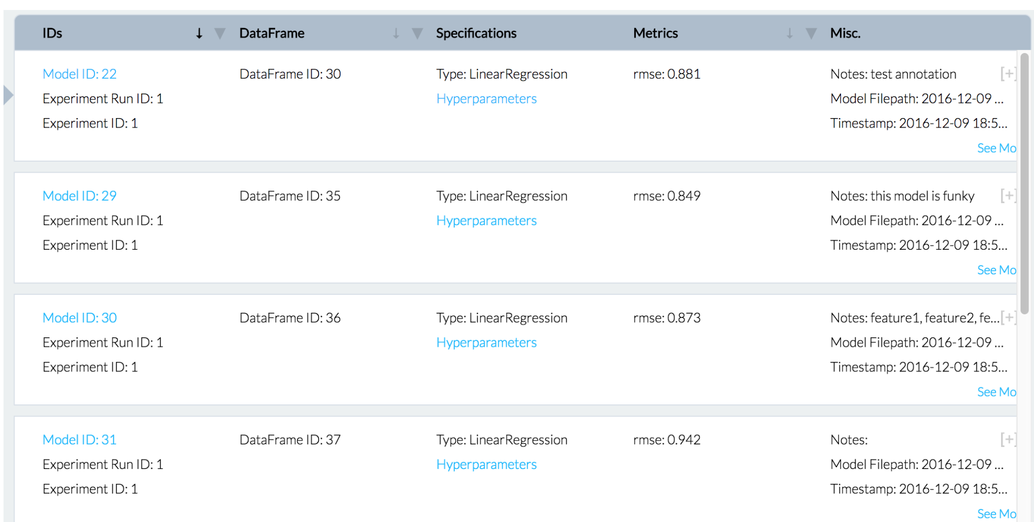

Figure 5: Tabular Models View

systems have focused on tracking models in a limited fashion; consequently, to support diverse ML applications, we expect model management systems to evolve in the following directions.

Model Data Pipelines. For many machine learning applications, the ultimate performance of the machine learning model depends on the features or data attributes used by the model. As a result, when a model is to be reproduced, it requires accurate records of the data and transformations used to produce features. We expect model management systems to evolve to accurately record data versions (train and test) as well data transformations that are used to generate features. This metadata may come from a separate data processing or workflow system as opposed to being generated by the model management system, however, the model management system would track all the metadata required to create the model end-to-end.

Model Interoperability. One of the key challenges in model management described before is the diversity in ML models and frameworks. While some frameworks support rapid development, others might be more suitable for deployment in production settings. Given the rapid proliferation of ML frameworks, we expect model management systems to support, if not provide, a level of interoperability between different ML environments and frameworks (e.g., PMML [9] and ONNX [14] provide a good start). This will enable data scientists to pick and choose the best framework for each phase of the model life-cycle without being tied to a single framework.

Model Testing. As more decisions and business logic get delegated to machine learning models, the importance of testing models (similar to code testing) will increase and will become a key part of managing the model life-cycle. For instance, defining unit tests for models and their input data will become commonplace. Similarly, we expect integration tests with models to become more prevalent as predictions from one model get used as input for other models. Finally, we note that as adversarial attacks on models increase, testing of models and edge cases will become key, requiring the development of new techniques to prevent adversarial attacks (e.g., [10, 9]).

Model Monitoring. While model testing takes place before a model is deployed, model monitoring takes place once the model is deployed in a live system. Model monitoring today is largely limited to monitoring system-level metrics such as the number of prediction requests, latency, and compute usage. However, we are already seeing the need for data-level monitoring of models (e.g., as noted in [25]) to ensure that the offline and live data fed to a model is similar. We expect model management systems to encompass model monitoring modules to ensure continued model health, triggering alerts and actions as appropriate.

Model Interpretability and Fairness. As models are used for automated decision making in regulated industries and increasingly used by non-technical users, explaining the results of models will become a key aspect of the management of deployed models (as evidenced by the rich research on interpretability [17, 6, 23]). We view a model management system as the system of record for all models and, therefore, the logical gateway for model interpretability and understanding. In the future, we therefore expect model management systems to expose interpretability functionality for every recorded model.

6 Conclusion

ML models are becoming ubiquitous is a variety of domains. This proliferation of ML brings to the forefront the need for systems that are responsible for managing models across their entire life-cycle starting with data preparation to model retirement. In this paper, we discussed the motivation for model management systems as well as challenges associated with consistently tracking models throughout their life-cycle. We described MODELDB, the first open-source model management system developed at MIT. Finally, we described the evolving opportunities in model management.

References

[1] Louis Bavoil, Steven P Callahan, Patricia J Crossno, Juliana Freire, Carlos E Scheidegger, Cláudio T Silva, and Huy T Vo. Vistrails: Enabling interactive multiple-view visualizations. In *Visualization, 2005. VIS 05. IEEE*, pages 135–142. IEEE, 2005.

[2] Steven P Callahan, Juliana Freire, Emanuele Santos, Carlos E Scheidegger, Claudio T Silva, and Huy T Vo. Managing the evolution of dataflows with vistrails. In *Data Engineering Workshops, 2006. Proceedings. 22nd International Conference on*, pages 71–71. IEEE, 2006.

[3] Steven P Callahan, Juliana Freire, Emanuele Santos, Carlos E Scheidegger, Cláudio T Silva, and Huy T Vo. Vistrails: visualization meets data management. In *Proceedings of the 2006 ACM SIGMOD international conference on Management of data*, pages 745–747. ACM, 2006.

[4] Kaggle competition. Toxic comment classification challenge. `https://www.kaggle.com/c/jigsaw-toxic-comment-classification-challenge`.

[5] Kaggle Competition. Zillow prize: Zillow's home value prediction (zestimate). `https://www.kaggle.com/c/zillow-prize-1`.

[6] Been Doshi-Velez, Finale; Kim. Towards a rigorous science of interpretable machine learning. In *eprint arXiv:1702.08608*, 2017.

[7] Facebook Engineering. Introducing fblearner flow: Facebook's ai backbone. `https://code.facebook.com/posts/1072626246134461/introducing-fblearner-flow-facebook-s-ai-backbone/`.

[8] Uber Engineering. Meet michelangelo: Uber's machine learning platform. `https://eng.uber.com/michelangelo/`.

[9] Reuben Feinman, Ryan R Curtin, Saurabh Shintre, and Andrew B Gardner. Detecting adversarial samples from artifacts. *arXiv preprint*, 2017.

[10] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *ICLR*, 2015.

[11] Apache Software Foundation. Airflow.

[12] MIT DB Group. Modeldb. `https://github.com/mitdbg/modeldb`, 2017.

[13] Alex Guazzelli, Wen-Ching Lin, and Tridivesh Jena. *PMML in Action: Unleashing the Power of Open Standards for Data Mining and Predictive Analytics*. CreateSpace, Paramount, CA, 2010.

[14] Joseph M Hellerstein, Vikram Sreekanti, Joseph E Gonzalez, James Dalton, Akon Dey, Sreyashi Nag, Krishna Ramachandran, Sudhanshu Arora, Arka Bhattacharyya, Shirshanka Das, et al. Ground: A data context service.

[15] Arun Kumar, Robert McCann, Jeffrey Naughton, and Jignesh M. Patel. Model selection management systems: The next frontier of advanced analytics. *SIGMOD Rec.*, 44(4):17–22, May 2016.

[16] Bertram Ludäscher, Ilkay Altintas, Chad Berkley, Dan Higgins, Efrat Jaeger, Matthew Jones, Edward A Lee, Jing Tao, and Yang Zhao. Scientific workflow management and the kepler system. *Concurrency and Computation: Practice and Experience*, 18(10):1039–1065, 2006.

[17] Scott M Lundberg and Su-In Lee. A unified approach to interpreting model predictions. In *Advances in Neural Information Processing Systems 30*, pages 4768–4777. Curran Associates, Inc., 2017.

[18] H. Miao, A. Li, L. S. Davis, and A. Deshpande. Modelhub: Deep learning lifecycle management. In *2017 IEEE 33rd International Conference on Data Engineering (ICDE)*, pages 1393–1394, April 2017.

[19] Hui Miao, Amit Chavan, and Amol Deshpande. Provdb: Lifecycle management of collaborative analysis workflows. 2017.

[20] MLFlow. Mlflow. `https://github.com/mlflow/mlflow`.

[21] Official Journal of the European Union. General data protection regulation. `https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=celex%3A32016R0679`.

[22] ONNX. Onnx: Open neural network exchange. `https://github.com/onnx/onnx`.

[23] Maithra Raghu, Justin Gilmer, Jason Yosinski, and Jascha Sohl-Dickstein. Svcca: Singular vector canonical correlation analysis for deep learning dynamics and interpretability. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 6078–6087. Curran Associates, Inc., 2017.

[24] SAS. Sas model manager. `https://www.sas.com/en_us/software/model-manager.html`.

[25] D Sculley, Todd Phillips, Dietmar Ebner, Vinay Chaudhary, and Michael Young. Machine learning: The high-interest credit card of technical debt.

[26] Spotify. Luigi. `https://github.com/spotify/luigi`.

[27] The Galaxy Team and Community. The galaxy project. `https://galaxyproject.org/`.

[28] Manasi Vartak, Harihar Subramanyam, Wei-En Lee, Srinidhi Viswanathan, Saadiyah Husnoo, Samuel Madden, and Matei Zaharia. Modeldb: A system for machine learning model management. In *Proceedings of the Workshop on Human-In-the-Loop Data Analytics*, HILDA '16, pages 14:1–14:3, New York, NY, USA, 2016. ACM.

[29] Katherine Wolstencroft, Robert Haines, Donal Fellows, Alan Williams, David Withers, Stuart Owen, Stian Soiland-Reyes, Ian Dunlop, Aleksandra Nenadic, Paul Fisher, et al. The taverna workflow suite: designing and executing workflows of web services on the desktop, web or in the cloud. *Nucleic acids research*, 41(W1):W557–W561, 2013.