Arangopipe, a Tool for Machine Learning Meta-Data Management

Jörg Schad[1], Rajiv Sambasivan[1], and Christopher Woodward[1]

[1]ArangoDB

March 26, 2021

Abstract

Experimenting with different models, documenting results and findings, and repeating these tasks are day-to-day activities for machine learning engineers and data scientists. There is a need to keep control of the machine learning pipeline and its metadata. This allows users to iterate quickly through experiments and retrieve key findings and observations from historical activity. This is the need that Arangopipe serves. Arangopipe is an open-source tool that provides a data model that captures the essential components of any machine learning lifecycle. Arangopipe provides an application programming interface that permits machine learning engineers to record the details of the salient steps in building their machine learning models. The components of the data model and an overview of the application programming interface are provided. Illustrative examples of basic and advanced machine learning workflows are provided. Arangopipe is not only useful for users involved in developing machine learning models but also useful for users deploying and maintaining them.

Overview

The outlook for the adoption of machine learning-based solutions into technology-enabled aspects of business is strong (*The McKinsey State of AI in 2020 Survey*, n.d.). This recent survey by McKinsey suggests that companies attribute their growth to the adoption of Artificial Intelligence(AI) in their application software. The companies that see the strongest growth are those associated with a core set of best practices and processes around developing and implementing AI-based solutions. Other surveys such as (*Algorithmia, State of Enterprise ML, a 2020 survey*, n.d.),(*Kaggle, State of Data Science and Machine Learning 2020*, n.d.) and (*Anaconda, State of Data Science 2020*, n.d.) offer insights into the current state of affairs in developing enterprise machine learning solutions. These surveys suggest that reproducibility of results and tracing the lineage of activities performed on data science projects is a pain point in operationalizing machine learning solutions. This is consistent with the technical debt in the machine learning liefecycle discussed in (Sculley et al., 2014). One of the recommendations of (*Anaconda, State of Data Science 2020*, n.d.) is the integration of open-source tools to circumvent these challenges. Reproducibility of results has plagued the research community too. (Bouthillier et al., 2019) is recent work that exemplifies the brittleness of results and the need to promote rigorous methodology and tools to facilitate reproducible results. Therefore, tools that facilitate reproducibility can benefit both the practitioner and the research community. Not surprisingly, tools and methods to address this problem have attracted the attention of researchers in this area (Sebastian Schelter and Felix Bießmann and Tim Januschowski and David Salinas and Stephan Seufert and Gyuri Szarvas, 2018), (Manasi Vartak and Samuel Madden, 2018) (Matei Zaharia and Andrew Chen and Aaron Davidson and Ali Ghodsi and Sue Ann Hong and Andy Konwinski and Siddharth Murching and Tomas Nykodym and Paul Ogilvie and Mani Parkhe and Fen Xie and Corey Zumar, 2018), and (Konstantinos Katsiapis and Kevin Haas, 2019). Arangopipe is a solution that was developed to address this pain point leveraging the unique features that a multi-model database supporting a graph data model offers in developing solutions for this problem.

Much of model development for machine learning and data analytic applications involves analyzing activities and findings that went into building earlier models, such as examining distribution characteristics of features, effective modeling choices, and results from hyper-parameter tuning experiments. Similarly, when these models are deployed, there is a frequent need to review data from previous deployments to verify configuration and deployment steps. These activities are those associated with reproducibility and traceability in surveys such as (*Anaconda, State of Data Science 2020*, n.d.)and (Algorithmia, State of. . .). Therefore, applications and tools that record relevant information about machine learning model development tasks and facilitate the easy access and search of this information are of importance to teams involved in the development, deployment, and maintenance of machine learning applications. Arangopipe is a tool that provides these features. The process of developing and deploying machine learning models is often abstracted as a pipeline of activities. A graph is a natural data structure to represent the activities in the pipeline. Many machine learning tools and libraries routinely model the activities related to model development as a graph. There is great diversity in the range of machine learning applications and the complexity associated with developing these applications. Consequently, there is great diversity in data that needs to be captured from the development and deployment of these applications. A document-oriented data model is a good fit to accommodate this diverse range of data capture. With a document-oriented data model, there is no need to define the structure of the data before storing it. A database that permits both a graph and a document-oriented data model is ideal to capture data from machine learning model development and deployment activity. Arango DB provides these features. The rest of this article is structured as follows. In section 2, the nature of data collected from machine learning projects is discussed. In section 4, a series of illustrative examples illustrating the features of Arangopipe is provided. In section In section 5, work that is closely related to the problem solved by Arangopipe is discussed. In section 7, the salient observations and facts about using Arangopipe in data science projects are summarized.

Data science workflow

Arangopipe provides a data model that permits the capture of information about activities performed as part of a data science project. The information captured about the activity is also referred to as meta-data about the activity. The reference data model for Arangopipe was developed after critically reviewing the nature of meta-data captured across a range of projects from our own experience as well as standardization efforts such as (Schelter et al., 2017) and (*mlspec/MLSpec*, n.d.). While these efforts aim to standardize the operations and the data captured in productionalizing machine learning applications, it can be easily adapted thanks to the flexibility of the graph data model. A schematic representation of the data model is shown in Fig. 1. A graph data model is used to capture a comprehensive set and realtions of activities that are performed in a data science project.

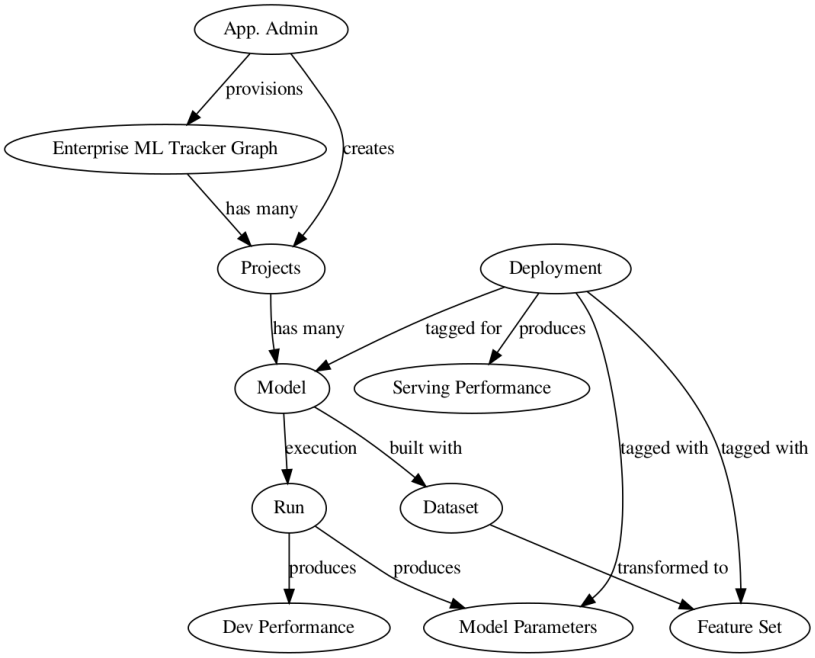

Figure 1: Reference Arangopipe Schema

The purpose of each vertex type and its relevant relationships are as follows. Arangopipe provides two interfaces to capture and track data science activities. These are the administrative interface and the project interface. The administrative interface is called *arangopipe admin* and the project interface is called *arangopipe*. The administrative interface is used to provision arangopipe for use in an organization. The project interface is used to capture data from project activities. To use Arangopipe in your organization, the graph used to track machine learning and data science activities needs to be provisioned. This graph is called *Enterprise ML Tracker Graph* and is provisioned using the administrative interface. Machine learning and data science activities are organized by *projects*. The administrative interface is used to add machine learning projects that need to be tracked. Each *project* can track multiple model-building activities. A *model* is any data science activity that is tracked by the project. This can include a wide range of tasks performed by the data science team, such as:

1. Data analysis experiments to profile attribute characteristics in a dataset
2. Experiments to evaluate candidate models for a particular machine learning task.
3. Hyper-parameter tuning experiments.
4. Experiments to evaluate *data drift*.

Machine learning models use a *featureset* to train the model. A *featureset* is constructed from a *dataset* using transformations if required. A *run* captures the execution of a *model*. It links the inputs and the outputs associated with the execution of a *model*. A *run* executes a *model* with a set of *model parameters*. The model performance observed during the *run* is captured by the *dev performance* vertex type. A model with an acceptable level of performance is deployed to production. A *deployment* is used to capture such models. A *deployment* is created using the administrative interface. A *deployment* links the assets used by the deployed *model*. These include the *dataset*, the *featureset*, the *model*, and its *model parameters*. As a deployed model serves requests, we get to observe its performance. This performance is captured by the *serving performance* vertex type. See section 4 for the details of examples that illustrate how machine learning project activity can be captured with Arangopipe.

It should be noted that it is possible to extend the graph model through the administrative interface. For example, if you need to track a new type of meta-data for your machine learning experiments and you wish to capture this as a separate vertex type and create edges from existing vertices to the new vertex type, that can be done. Note that no schema constraint enforces the kind of data that you can capture with each vertex type described above. This benefit comes from using a document-oriented model for node data. This feature can be exploited for the following purposes. The same *model* vertex type can be used to capture models from multiple machine learning libraries. The *github* repository for this project contains examples that illustrate the use of identical workflow steps for model development tasks from different libraries such as *scikit-learn*, *tensorflow*, and *pytorch*. The artifacts and results produced from modeling tasks can be stored as node data. These results can be retrieved and re-materialized as programmatic objects. An example of this feature to store and re-materialize results from *Tensor Flow Data Validation* is available in the Arangopipe repository. In general, as long as the vertex data has a JSON representation and is not too large, for example, the weights associated with a massive neural network, it can be captured as part of the vertex data. To associate large data with vertices, the data can be stored in appropriate file formats such as *HDF5*, and the URL to the data file could be stored as a node property.

Software Implementation

Arangopipe consists of the following components:

Application Programming Interface (API)

The API is primarily meant for data scientists to track and log data science activities they perform for projects. The API provides the administrative and the project interfaces discussed above. A project administrator should first provision Arangopipe for use in the organization using the administrative interface. Provisioning sets up the database and graph to track machine learning projects as well as the connection to the provisioned database. The administrator then adds projects that need to be tracked to Arangopipe. At this point, project members can use the connection and project information to start logging model development activities.

Arango DB

This is the database used with Arangopipe. Project administrators and personnel involved in deploying and maintaining machine learning models can use the *Arango Query Language (AQL)* through the web interface of Arango DB to run queries on the Arangopipe database if needed.

A Web User Interface

A web interface to track, view, and search for information about assets and model building activity is provided. The web interface is primarily meant to be used by personnel deploying or managing machine applications to obtain historic information about model development activity, deployments, serving performance details, etc.

Container Images

Docker images of Arangoipe bundled with major deep learning toolkits, *pytorch* and *tensorflow* are available. These images contain the above components along with *pytorch* or *tensorflow*. Since these toolkits are widely used, data scientists and machine learning engineers can use these images to start using Arangopipe in their organizations.

It is possible to use Arangopipe with *Oasis*, Arango DB's managed service offering on the cloud. This would require no installations or downloads. The details of doing this are provided in section

A schematic illustration of how the components discussed above are used with Arangopipe and ArangoDB is shown in Fig 2. An Arangopipe Administrator can use a Jupyter notebook to provision Arangoipe for a project. The administrator would use the *administration API* for this purpose. Project team members, for example, data scientists, can use a Jupyter notebook (or a python script) to log meta-data about project activities using the *arangopipe API*. This interface exposes two other interfaces. The *arangopipe_storage API* offers functionality related to capturing meta-data from machine-learning pipeline activities in ArangoDB. The *arangopipe_analytics API* offers functionality implemented with machine learning, such as API to identify dataset drift. Dev-Ops personnel involved in operationalizing machine learning applications can use the web user interface to look up details of deployments or assets such as models or datasets. Arango DB offers an API to develop analytic applications with ArangoDB. These applications can generate machine learning meta-data, for example, embeddings of graphs used in these applications. These applications can use the Arangopipe API to store this meta-data in ArangoDB.

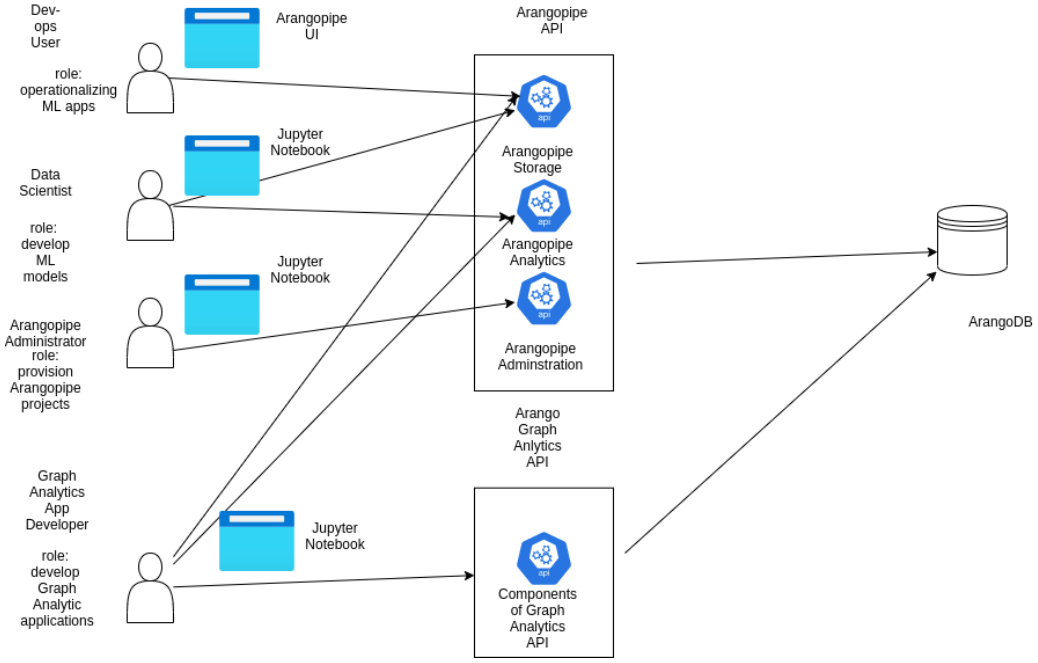

Figure 2: Using Arangopipe with Analytics Applications

For more information about these components or to evaluate Arangopipe, please visit the project github repository. The repository contains examples that illustrate the features of Arangopipe using (*Google Colaboratory*, n.d.) notebooks (see section). Working through these notebooks does not require the installation of software on your machine. These examples use Oasis, a managed service version of ArangoDB.

Illustrative Examples of Arangopipe

A comprehensive discussion of the advantages of using a graph data model to capture machine learning meta-data and a narrative detailing the progress of a data science project using Arangopipe is available in arangopipe-overview and (*arangopipe collaboration scenario*, n.d.). In this section, we present examples that illustrate the key features of Arangopipe. These features are illustrated with model building activity using the California Housing dataset (Pace & Barry, 1997)from the CMU statlib repository (*CMU statlib dataset archive*, n.d.). Each section below discusses a salient aspect of using Arangopipe with your model development efforts.

Basic Workflow

Please see Arangopipe Basic Workflow for an illustration of the basic workflow with Arangopipe. It can be run as a *colab* notebook, so no installation is necessary to work through the notebook. The notebook develops a *Linear Regression* model using the *scikit-learn (Pedregosa et al., 2011)* library and logs the results of the model development activity in Arangopipe. The notebook uses Arango DB's managed service offering, *Oasis*, as the database. The notebook execution begins with specifying a set of connection parameters to connect to the ArangoDB database instance to be used with Arangopipe. An API call is made through the administrative interface to provision the database for use with Arangopipe. This connection information can be saved and reused in subsequent interactions with the Arangopipe database instance. The project interface with which modeling activity is logged is then created with connection information used in the provisioning step. The notebook illustrates the typical administrative and project activities involved in using Arangopipe in a data science project. In particular, the following is illustrated:

Administrative Activities:

1. Provisioning an Arangopipe database
2. Adding a data science project to Arangopipe.

Data science Activities:

1. Registering assets: Assets such as a *dataset*, *featureset,* and *model* are registered with Arangopipe in the notebook.
2. Logging the model building activity: The model that is developed is executed in the notebook. This activity is captured as a *run*. The *run* links the assets used in the execution, the model parameters, and the observed development performance.

As the notebook illustrates, the process of provisioning projects to use Arangopipe and tracking machine learning assets and meta-data with it is quite straightforward.

Reusing Archived Steps

The sequence of steps illustrated with the basic workflow can be used to capture data science project activities from a variety of tasks. Data scientists can store key results from model building experiments in Arangopipe. These results can be programmatic artifacts, such as results from exploratory data analysis or a hyperparameter tuning experiment. A colleague can retrieve these artifacts from Arangopipe in a subsequent session and re-materialize them as programmatic entities. Please see using Arangopipe with TFDV for exploratory data analysis for an example of performing this in an exploratory data analysis task using *tensor flow data validation tensorflow data validation*. Please see performing hyper-parameter optimization with Arangopipe for an example with a hyperparameter tuning experiment. The capability to store node data as documents is exploited in these examples. Node data and results from modeling are easily converted into JSON, which is the format that ArangoDB stores documents in. In a subsequent project activity, these model results can be retrieved from Arangopipe and rematerialized to programmatic objects native to the

tool library that created them. These examples illustrate another aspect of the utility of Arangopipe. Not all activities in a machine learning project are targeted at developing a machine learning model. There is much effort expended in activities like descriptive and exploratory analysis of the data, transforming the data to a form amenable for model building, experiments to determine parameters for model building, etc. Meta-data from these activities can also be captured by Arangopipe.

Extending the data model

It is possible to extend the Arangopipe data model to suit the custom needs that a project may have. For example, if an organization would like to capture notebooks as a separate asset that is tracked, this can be done. Please see the "Advanced Modeling" section of performing hyper-parameter optimization with Arangopipe for an example illustrating this. Projects that need to change or define a different data model in their projects can do so as illustrated in this example.

Experimenting and documenting facts about models and data

Documenting facts about models is a routine task for data scientists. The bias and variance of a developed model are of interest to data scientists on regression tasks. Please see using arangopipe to document model bias for an example showing how model bias can be captured and stored with Arangopipe.

Checking the validity and effectiveness of machine learning models after deployment

Data drift and concept drift are known issues with maintaining and managing machine learning solutions. Arangopipe provides an extensible API to check for dataset drift. To detect dataset drift, a machine learning classifier is used to discriminate between the dataset used to develop the model and the data the model is receiving. A reference implementation using a RandomForest (Breiman, 2001) classifier is provided. By implementing the abstract class, users can use the same idea, but use a different classifier for the task. Please see capturing dataset drift with Arangopipe for an example of using Arangopipe to evaluate data drift.

Using the Arangopipe Web User-Interface

The easiest path to exploring the Arangopipe Web-User-Interface is by running one of the Arangopipe container images. Container images with pytorch and tensorflow are available. The project documentation provides detailed instructions on launching one of these containers. After running the docker image, the web-user interface should be accessible. The section "Arangopipe User Interface" on the project documentation page provides information about the organization of the user interface and the details of using key features like *asset search* and *asset lineage tracking*. The Arangopipe Web-User Interface is primarily meant to be used by dev-ops personnel who operationalize machine learning applications. Searching for assets, tracing asset lineage, and obtaining information about past deployments are common tasks for personnel involved in operationalizing machine learning models. It is also possible to use AQL to query assets tracked by Arango DB. AQL queries may be executed from the Web-User-Interface.

Storing Features From Model Development

Feature engineering is very important in many machine learning tasks (Domingos, 2012). Arangopipe can be used to capture features generated from machine development. Graph embedding can be used to obtain a Euclidean representation of graph-structured data. These embeddings can be used as features in machine learning models. A variety of techniques exist to obtain these embeddings with each finding favor in particular applications (Goyal & Ferrara, 2018). Please see IMDB-Networkx-ArangoDB-Adapter for an example of using Arangopipe along with the networkx adapter to store results from embeddings generated from node2vec(Grover & Leskovec, 2016).

Support for R Models

Using the *reticulate* R package, it is possible to capture meta-data from R data science tasks in Arangopipe. Please see an overview of reticulate (Ushey et al., 2020)for information about this package and details of type conversion between R and python. Please see using R with Arangopipe for an example of illustrating the capture of meta-data from R model development activity with Arangopipe.

Related Work

Efforts to standardize operations and data related to managing meta-data from machine learning were considered and are actively monitored in the development of Arangopipe. An example illustrating the use of the elements of ML Spec in an Arangopipe model is available. The Open Neural Network Exchange is a standard for the development of interoperable machine learning models. This standard defines a standard set of data elements and operations for a machine learning model. In this standard, as well as in most machine learning model development tools, the computation associated with developing the machine learning model is abstracted as a graph. Many tools to build data science pipelines such as Airflow and Luigi model the pipeline computation as a graph. In particular, the computation associated with the model or data science pipeline is expressed as a Directed Acyclic Graph. As execution flows through this graph, there is meta-data that is produced. This is the meta-data that is captured by Arangopipe. Since a graph is used to express computation, a graph data model is a natural fit to capture meta-data about data science workflows. The meta-data are stored as documents in the nodes and edges of the graph. If there is a need to enforce schema constraints on the meta-data obtained as the computation progresses, this can be done. If the nature of meta-data obtained from the computation is dynamic and structural constraints for the data are not known apriori, the document-oriented storage model can offer the flexibility to capture such data. There is no need to define constraints about the data before it is captured as a document. In some data models, for example, the relational model, these constraints must be defined before the system starts ingesting data. This implies that structural constraints of the meta-data must be codified and set up before using the model to capture meta-data. The multi-model aspect of ArangoDB that permits the use of both a graph and a document-oriented data model to capture data is, therefore, a data model that provides both native expressiveness and flexibility for the capture of meta-data from machine learning pipelines. For the reasons discussed in section 1, the need to develop tools that facilitate reproducibility and tracking the provenance of models has attracted the attention of the research community. (Baylor 2017), (Zaharia 2018), (Vartak 2018), and (Tsay 2018)are tools developed to solve the reproducibility problem. We believe that the graph and document-oriented storage models of ArangoDB offer considerable versatility and flexibility in capturing and analyzing meta-data from machine learning pipelines. As discussed, the provided data model was derived after reviewing solutions from our own experience, other solutions to the problem, a survey of research on this problem, and standardization efforts related to this area. If applications or projects need to use a different data model or extend the provided data model, Arangopipe provides an API for this purpose. As standardization efforts in this application area progress, new ideas that need to be captured by the data model can be added using the provided API.

Building and Testing

The project repository provides detailed information on the various options to try Arangopipe on a project. Arangopipe is open source, contributions are welcome! For details of building Arangopipe, please see the build instructions in the project repository. Unit tests that illustrate how each API in Arangopipe is to be invoked are available. For each method in the Arangopipe API, a test case is available. A review of the test cases should provide the complete details of invoking each method in the Arangopipe API. New users of Arangopipe can use the test cases as a reference for logging their project activity with Arangopipe. To facilitate exploration of the API and the Web UI, a test data generator that provides example meta-data

is also provided. The test data generation utility runs *linear regression* models on bootstrapped versions of the California Housing dataset and logs meta-data from model evaluations.

Conclusion

The capture of meta-data from the machine learning lifecycle is important to both researchers and practitioners. Tools that capture and facilitate analysis of such meta-data are therefore useful to both these communities. The multi-model feature of ArangoDB presents some unique advantages in the capture and analysis of data from machine learning pipelines. A graph is a natural abstraction for this application since most tools used in developing machine learning pipelines model the computation as a graph. The document-oriented feature of ArangoDB offers flexibility in capturing meta-data from custom machine learning pipelines. Pipelines do not have to define the structural constraints about the types of elements in the meta-data obtained from machine learning pipelines before persisting it, as is the case for example with relational databases. However, if such constraints are desired, then it is possible to enforce them with a schema. The data model offered with Arangopipe captures the basic elements of any machine learning pipeline. This model is extendible and should a particular application need it, the API offers methods to make the desired changes. ArangoDB is tracking standards development initiatives around machine learning meta-data, such as ML spec. ArangoDB is committed to the further development of Arangopipe. Feedback and questions about Arangopipe are welcome in the ArangoML slack channel and the issues section of the project repository.

References

https://www.mckinsey.com/business-functions/mckinsey-analytics/our-insights/global-survey-the-state-of-ai-in-2020

https://info.algorithmia.com/hubfs/2019/Whitepapers/The-State-of-Enterprise-ML-2020/Algorithmia_2020_State_of_Enterprise_ML.pdf

https://www.kaggle.com/kaggle-survey-2020

https://www.anaconda.com/state-of-data-science-2020

Machine learning: The high interest credit card of technical debt. (2014).

Unreproducible research is reproducible. (2019). *International Conference on Machine Learning*, 725–734.

On Challenges in Machine Learning Model Management. (2018). *IEEE Data Eng. Bull.*, *41*.

MODELDB: Opportunities and Challenges in Managing Machine Learning Models. (2018). *IEEE Data Eng. Bull.,*, *41*.

Accelerating the Machine Learning Lifecycle with MLflow. (2018). *IEEE Data Eng. Bull.*, *41*.

Towards ML Engineering with TensorFlow Extended (TFX). (2019). *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD 2019, Anchorage, AK, USA, August 4-8, 2019*.

Automatically tracking metadata and provenance of machine learning experiments. (2017). *Machine Learning Systems Workshop at NIPS*, 27–29.

https://github.com/mlspec/MLSpec. https://github.com/mlspec/MLSpec

https://colab.research.google.com/github/tensorflow/examples/blob/master/courses/udacity_intro_to_tensorflow_for_deep_learning/l01c01_introduction_to_colab_and_python.ipynb.

https://colab.research.google.com/github/tensorflow/examples/blob/master/courses/udacity_intro_to_tensorflow_for_deep_learning/l01c01_introduction_to_colab_and_python.ipynb

https://www.arangodb.com/2021/01/arangoml-series-multi-model-collaboration/. https://www.arangodb.com/2021/01/arangoml-series-multi-model-collaboration/

Sparse spatial autoregressions. (1997). *Statistics & Probability Letters*, *33*(3), 291–297.

http://lib.stat.cmu.edu/datasets/. http://lib.stat.cmu.edu/datasets/

Scikit-learn: Machine Learning in Python. (2011). *Journal of Machine Learning Research*, *12*, 2825–2830.

Random forests. (2001). *Machine Learning*, *45*(1), 5–32.

A few useful things to know about machine learning. (2012). *Communications of the ACM*, *55*(10), 78–87.

Graph embedding techniques, applications, and performance: A survey. (2018). *Knowledge-Based Systems*, *151*, 78–94.

node2vec: Scalable feature learning for networks. (2016). *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 855–864.

reticulate: Interface to 'Python'. (2020). https://CRAN.R-project.org/package=reticulate