MODELDB: A System for Machine Learning Model Management

Manasi Vartak, Harihar Subramanyam, Wei-En Lee, Srinidhi Viswanathan, Saadiyah Husnoo
Samuel Madden, Matei Zaharia
MIT
{mvartak, madden, matei}@csail.mit.edu, {hsubrama, weienlee, srinidhi, saadiyah}@mit.edu

ABSTRACT

Building a machine learning model is an iterative process. A data scientist will build many tens to hundreds of models before arriving at one that meets some acceptance criteria (e.g. AUC cutoff, accuracy threshold). However, the current style of model building is ad-hoc and there is no practical way for a data scientist to manage models that are built over time. As a result, the data scientist must attempt to "remember" previously constructed models and insights obtained from them. This task is challenging for more than a handful of models and can hamper the process of sensemaking. Without a means to manage models, there is no easy way for a data scientist to answer questions such as "Which models were built using an incorrect feature?", "Which model performed best on American customers?" or "How did the two top models compare?" In this paper, we describe our ongoing work on ModelDB, a novel end-to-end system for the management of machine learning models. ModelDB clients automatically track machine learning models in their native environments (e.g. scikit-learn, spark.ml), the ModelDB backend introduces a common layer of abstractions to represent models and pipelines, and the ModelDB frontend allows visual exploration and analyses of models via a web-based interface.

1. INTRODUCTION

Building a real-world machine learning model is a trial-and-error-based iterative process. A data scientist starts with a hypothesis about the underlying data, builds a model based on this hypothesis, tests the model, and refines the hypothesis as well as model based on the results. To understand the process of model building, we interviewed data scientists from a host of different industries ranging from social media companies to small startups in the IoT space. The process of model building across all these companies could best be described as "ad-hoc" where the data scientist often built hundreds of models before arriving at one that met some acceptance criteria (e.g. AUC, accuracy). However, the data scientist had no means of tracking previously-built models or insights from previous experimentation. Consequently, the data scientist had to *remember* relevant information about previous models to inform the design of the next set of models. As expected, however, this task was challenging for more than a handful of models and hampered the scientist's ability to derive insights from the data. Moreover, at one of the companies, we found that the lack of model and result persistence cast doubt on the conclusions of a previous experiment, leading the team to re-run a set of expensive modeling workflows.

This iterative, ad-hoc nature of model building brings to light an important and little-studied problem for machine learning tools: **model management**. Model management is the problem of tracking, storing and indexing large numbers of machine learning models so they may subsequently be shared, queried and analyzed. Lack of model management can not only waste efforts (above) and slow down the modeling process, but it can also cause insights and trends to be overlooked. We find that model management provides essential support for multiple activities: (i) It helps data scientists recapitulate insights by providing an overview of models built to far. (ii) It helps with the data scientists' *sensemaking* process [9], i.e., the process of finding and organizing information to build a mental model of the underlying phenomenon. (iii) It helps find trends and perform meta-analyses across models (e.g. "Which model worked best on American customers?"). (iv) With proper indexing, it enables data scientists to search through models (e.g."Which models used this incorrectly coded feature?"). (v) It enables easy collaboration between data scientists.

In this paper, we present ongoing work on ModelDB, a system to manage machine learning models. ModelDB automatically tracks machine learning models in their native environment, indexes them intelligently, and allows flexible exploration of models via SQL as well as a visual interface. To provide this functionality, ModelDB provides native client libraries for different machine learning environments (currently spark.ml[1] and scikit-learn[2]), a storage layer optimized to store models, and a web-based visual interface that supports meta-analyses across models.

For ease of explanation, the above discussion refers to models in isolation; however, in real-world scenarios, models are associated with multi-stage *pipelines* that perform various types of pre-processing before the model is trained. Therefore, when managing models, we are, in fact, managing multi-stage pipelines involving pre-processing, training and testing steps. Along with models and pipelines, ModelDB also manages *metadata* (e.g., parameters of pre-processing steps, hyperparameters for models etc.), *quality metrics* (e.g. AUC, accuracy), and, if required, *training and test data* for each model.

2. RELATED WORK

Scientific workflow management is a rich area of research that

HILDA'16, June 26 2016, San Francisco, CA, USA

ISBN 978-1-4503-4207-0/16/06…$15.00

DOI: http://dx.doi.org/10.1145/2939502.2939516

[1] http://spark.apache.org/docs/latest/ml-guide.html
[2] http://scikit-learn.org/

has produced systems including Kepler [8] and Taverna [12] which are popular in the scientific community. Commercial software vendors have also introduced tools such as the AzureML suite[3] and the SeaHorse suite[4] that allow the (graphical) construction of machine learning workflows. The system closest to ModelDB in terms of flavor and functionality is the VisTrails system [1, 3, 2]. VisTrails was first introduced as a solution to track visual analysis workflows and has now evolved to support generalized scientific workflows, including support for creating workflows using scikit-learn.

A drawback of almost all of the current workflow systems, including VisTrails, is that they require scientists to use a GUI to define each workflow before execution. Specifically, to build a workflow, a data scientist must find the required operators from a predefined library of operators and drag-and-drop components onto a canvas. Although almost all workflow tools are GUI-based, all the data scientists we interviewed (in industry as well as research labs) overwhelmingly concurred that GUIs were too restrictive for writing their machine learning pipelines and that most of their pipelines were constructed on-the-fly as opposed to being pre-determined. As a result, in ModelDB, we chose to infer workflows based on the data scientist's commands and use visualization only as a means to perform post-hoc analysis of the pipeline.

The importance of *history* (i.e., providing a record of past operations) as an essential operation in data analysis has been highlighted by Shneiderman and others in [11, 6]. In line with this finding, [7] developed history tools for visual data mining, while [10] developed interactive visualizations of the visual analysis process in ReVisualization. Graphical histories for visual analysis in the context of Tableau[5] were studied by Heer et. al. in [5]. We find many similarities in the design considerations for providing history in visual analysis and providing history for machine learning pipelines.

3. MODELDB ARCHITECTURE

Figure 1 shows the high-level architecture of our system. ModelDB consists of three key components: native client libraries for different machine learning environments, a backend that defines key abstractions and brokers access to the storage layer, and a web-based visualization interface. ModelDB client libraries are currently available for scikit-learn and spark.ml. Data scientists can perform experimentation and model building in their favorite ML environment as usual while, in the background, the client library automatically extracts relevant information and passes it to the ModelDB backend. The ModelDB backend exposes a thrift interface to allow clients in different languages to communicate with the ModelDB backend. ModelDB stores models and pipelines as a sequence of actions (as opposed to states) and uses a branching model of history to track the changes in models over time [4]. The backend uses a relational database to store pipelines while a custom storage engine is used to store and index models. The third component of ModelDB, the visual interface, provides an easy-to-navigate layer on top of the database that permits visual exploration and analyses of models and pipelines.

Next, we sketch the details of the user-facing components of ModelDB, namely the client libraries and the web-based frontend.

3.1 Client Libraries

Most existing workflow management programs (e.g. VisTrails) 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 pipelines and that it was difficult to specify pipelines beforehand because they changed constantly. Moreover, we found that data scientists were unwilling to change their preferred ML environment for a workflow management system. 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 model logging accessible directly through code (as opposed to a GUI) and to build native logging libraries for different ML environment. The spark.ml and skicit-learn libraries are architected such that data scientists can use the environments for analysis exactly as they normally would and the library transparently and automatically logs the relevant data to the backend.

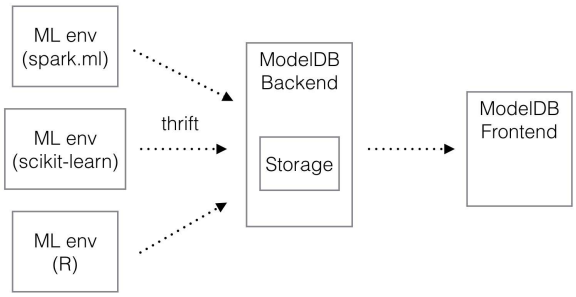

Figure 1: ModelDB Architecture

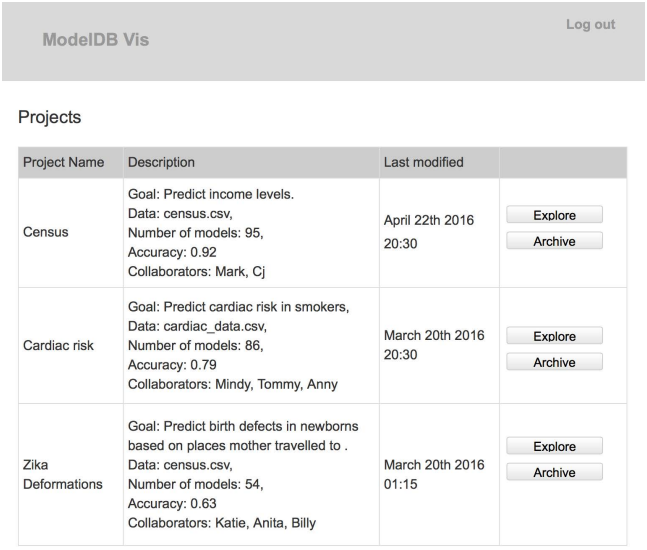

Figure 2: Overview Page

3.2 Frontend

Since data scientists build hundreds of models during a modeling project, many operators, models and pipelines are logged to the ModelDB backend. While it is possible to query the raw tables through SQL, we provide a visual interface through which data scientists can easily perform analyses without having to be familiar with the underlying database schema. Our goals for the visual interface are four-fold: (i) allow the data scientist to review all the models and pipelines built for a particular modeling project, (ii) allow data scientists and collaborators to quickly understand and inspect pipelines visually, (iii) allow the data scientist to perform different types of comparisons across pipelines and models, (iv) if

[3]https://azure.microsoft.com/en-us/services/machine-learning/
[4]https://seahorse.deepsense.io/
[5]http://tableau.com

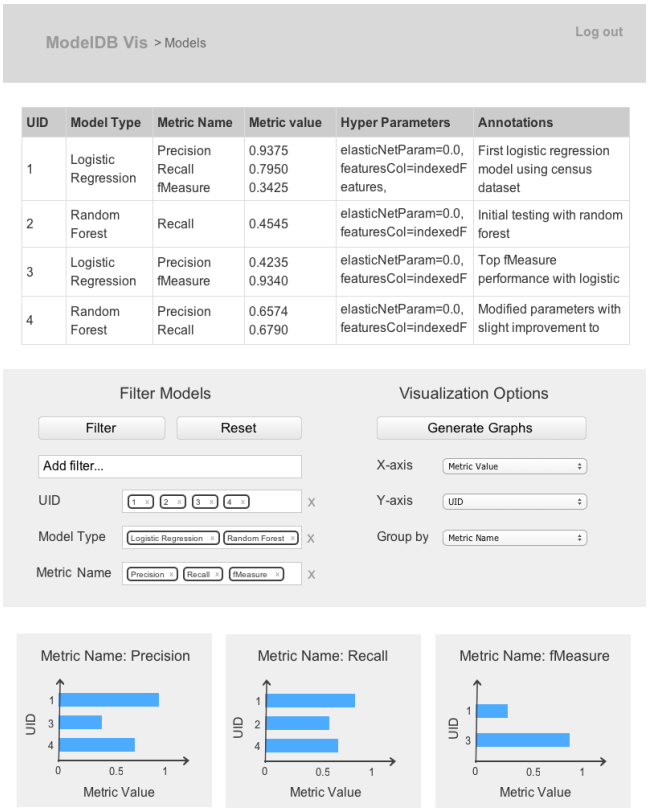

Figure 4: Models View

required, allow the data scientist to drill-down to the data itself to examine its evolution through the pipeline.

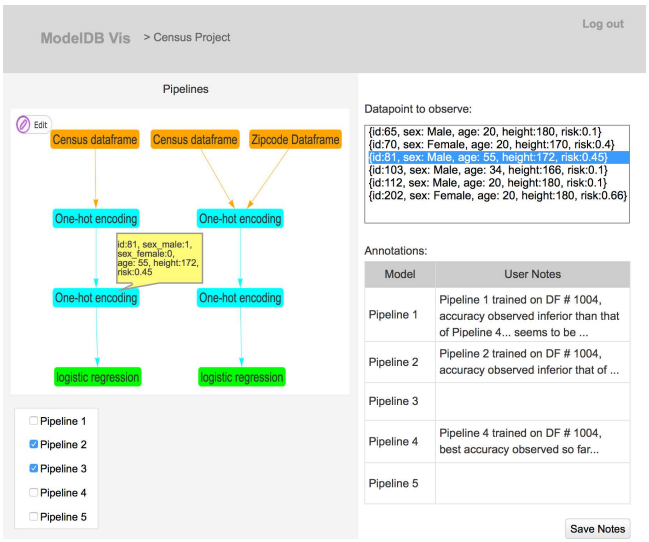

Figure 3: Pipelines View

Figures 2–4 show screenshots of the ModelDB visual interface. Figure 2 shows the overview page listing all the projects the data scientist is working on along with summary information about each project. ModelDB provides two views for exploring history related to a particular project. The first, shown in Figure 3, is called the **pipelines view**, while the second (Figure 4) is called **models view**. The *pipelines view* allows the user to explore a small number of pipelines in detail. For instance, the pipeline view depicts every stage of a pipeline along with details of the input, output and parameters for that stage. It also provides the ability to align and compare multiple pipelines to find similarities and differences. This view is particularly useful in helping data scientists and collaborators quickly understand how a pipeline was constructed. The pipeline view also provides a basic means of debugging the pipeline by allowing the data scientist to examine how the pipeline affects individual data items. The *models view*, in contrast, tabulates models generated by all pipelines built for a project. This view is best suited for obtaining a summary of the models built so far and performing comparisons across models using metadata and metrics. We support a Tableau-like interface to visually analyze models using their metadata and metrics. The functionality provided in this view, for example, makes it easy for a data scientist to graph how changes in a hyperparameter impact the accuracy of a model.

4. CONCLUSION

In this paper, we described ModelDB, a novel system to manage machine learning models. ModelDB provides native client libraries for popular ML environments including scikit-learn and spark.ml that can be used to automatically track models with minimal changes to code. The ModelDB backend indexes and stores models from multiple environments in a common format that allows easy querying. Finally, the visual interface provides a means for data scientists to visually explore modeling pipelines and run meta-analyses.

5. REFERENCES

[1] L. Bavoil, S. P. Callahan, P. J. Crossno, J. Freire, C. E. Scheidegger, C. T. Silva, and H. T. Vo. Vistrails: Enabling interactive multiple-view visualizations. In *Visualization, 2005. VIS 05. IEEE*, pages 135–142. IEEE, 2005.

[2] S. P. Callahan, J. Freire, E. Santos, C. E. Scheidegger, C. T. Silva, and H. 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] S. P. Callahan, J. Freire, E. Santos, C. E. Scheidegger, C. T. Silva, and H. 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] M. Derthick and S. F. Roth. Enhancing data exploration with a branching history of user operations, 2001.

[5] J. Heer, J. D. Mackinlay, C. Stolte, and M. Agrawala. Graphical histories for visualization: Supporting analysis, communication, and evaluation. *Visualization and Computer Graphics, IEEE Transactions on*, 14(6):1189–1196, 2008.

[6] J. Heer and B. Shneiderman. Interactive dynamics for visual analysis. *Commun. ACM*, 55(4):45–54, Apr. 2012.

[7] M. Kreuseler, T. Nocke, and H. Schumann. A history mechanism for visual data mining. In *Proceedings of the IEEE Symposium on Information Visualization*, INFOVIS '04, pages 49–56, Washington, DC, USA, 2004. IEEE Computer Society.

[8] B. Ludäscher, I. Altintas, C. Berkley, D. Higgins, E. Jaeger, M. Jones, E. A. Lee, J. Tao, and Y. Zhao. Scientific workflow management and the kepler system. *Concurrency and Computation: Practice and Experience*, 18(10):1039–1065, 2006.

[9] P. Pirolli and S. Card. The sensemaking process and leverage points for analyst technology as identified through cognitive task analysis. In *Proceedings of international conference on intelligence analysis*, volume 5, pages 2–4, 2005.

[10] A. C. Robinson and C. Weaver. Re-visualization: Interactive visualization of the process of visual analysis. In *Proceedings of the GIScience Workshop on Visual Analytics and Spatial Decision Support*, 2006.

[11] B. Shneiderman. The eyes have it: A task by data type taxonomy for information visualizations. In *Proceedings of the 1996 IEEE Symposium on Visual Languages*, VL '96, pages 336–, Washington, DC, USA, 1996. IEEE Computer Society.

[12] K. Wolstencroft, R. Haines, D. Fellows, A. Williams, D. Withers, S. Owen, S. Soiland-Reyes, I. Dunlop, A. Nenadic, P. 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*, page gkt328, 2013.