DataLab: Introducing Software Engineering Thinking into Data Science Education at Scale

Yang Zhang[1], Tingjian Zhang[2] Yongzheng Jia[1] Jiao Sun[1], Fangzhou Xu[1], and Wei Xu[1]
[1]Institute of Interdisciplinary Information Sciences, Tsinghua University
[2]Department of Computer Science and Technology, Shandong University
zhangyang14@mails.tsinghua.edu.cn, sdubeyhhhh@gmail.com, jiayz13@mails.tsinghua.edu.cn,
j-sun16@mails.tsinghua.edu.cn, xfz@mails.tsinghua.edu.cn, weixu@mail.tsinghua.edu.cn

***Abstract*—Data science education is a new area in computer science that has attracted increasing attention in recent years. However, currently, data science educators lack good tools and methodologies. In particular, they lack integrated tools through which their students can acquire hands-on software engineering experience. To address these problems, we designed and implemented DataLab, a web-based tool for data science education that integrates code, data and execution management into one system. The goal of DataLab is to provide a hands-on online lab environment to train students to have basic software engineering thinking and habits while maintaining a focus on the core data science contents. In this paper, we present the user-experience design and system-level implementation of DataLab. Further, we evaluate DataLab's performance through an in-classroom use case. Finally, using objective log-based learning behavior analysis and a subjective survey, we demonstrate DataLab's effectiveness.**

I. Introduction

Recent years have witnessed the rapid growth of the big data industry, and the labor market consistently shows a growing demand for data scientists. According to a Glassdoor [20] survey, data scientist became the best job in the US in 2016. To meet this growing demand, many universities have initiated data science courses covering data analytics, machine learning and applied statistics. Additionally, Massive Open Online Course (MOOC) platforms such as Coursera, edX and Udacity, have also started to offer a number of online data science courses, even including data science "specializations" or "nano degrees" to meet the demand.

Unlike the scalable programming and software engineering programs, current data science educators lack good tools and methodologies. In particular, they lack the tools through which their students can acquire hands-on experience, an essential step in data science.

Many instructors formulate data science courses as math or statistics courses that do not require any software engineering tools. Thus, many students in the data science program are domain experts without formal computer science training; consequently, they often form bad coding habits. For example, one common practice is to make redundant copies of the datasets, which leads to confusion from the meaningless copied file names such as `data.csv`, `data-version1.csv`, `data-final-version.csv`, `data-last-version.csv`. Obviously, such practices are antithetical to software engineering principles. Our experience shows that this practice has been a common source of bugs.

Other educators reuse traditional software development tools in their data science courses, including various integrated development environments (IDEs) and version control tools such as Git. However, data scientists are not software engineers. Students in such classes often get confused because these tools are complex to set up and use without even considering the problems stemming from different data sources with multiple versions. The common result is that the students spend much of their time learning these tools rather than learning the data science content, the main goal of the course.

On the instructor side, maintaining scalable tools and managing the entire learning experience is also a significant commitment. Even worse, the methodology for evaluating the learning outcome of a data science course has yet to be established. Instructors want to collect as much learning behavior data about the students as possible, but traditionally, instructors see only the final assignment submission, rather than the development process.

The following characteristics make a data science course project significantly different from a traditional software engineering project:

1) Data science requires managing both data and source code together. In a typical data science project such as a regression model, the students must manage the intermediate datasets at each step while cleaning, labeling, performing other preprocessing tasks, and model training and testing, and each step usually involves managing a number of scripts.

2) Many data science tasks are primarily concerned with tuning hyperparameters (the user-configurable parameters in a machine learning model such as learning rate, initial values, regularization values and so forth). As a common poor practice, students often hard-code these parameter values into their analytics code as global constants, often creating large number of code versions that differ by only a single constant. This practice further confuses both the students and the instructors tasked with reading their code.

3) Even a simple data science assignment requires a large dataset (this has become especially true now that people have started to focus on the so-called "big data" methods). It is often necessary for the instructor or the students to set up and maintain a moderate scale distributed platform such as Hadoop

or Spark and use many libraries to complete even a simple project. Lacking system operation expertise, such requirements raise the bar of teaching and learning data science and further distract peoples attention from core data science education.

4) Data science projects often require collaboration between students from different backgrounds; therefore, teamwork is an important goal in data science education. Without a good tool for sharing datasets and code, we often see students sending code via email and through datasets copied to Dropbox, leading to wasted time, and even worse, bad habits.

In this paper, we present our experiences with DataLab, a new web-based tool for data science education that integrates code, data and execution management into a single system. The goal of DataLab is to provide a hands-on online lab environment that serves to train students to acquire basic software engineering thinking and habits while maintaining their focus on the core data science content. DataLab also reduces the setup and management overhead for instructors and provides a scalable system that can support learning at a large scale.

DataLab is tailored to meet the special needs of data science education. Using a consistent, easy to follow user interface, DataLab allows students to develop a sense of the links among code, data, parameters and their revisions. The interface combines a web-based IDE with version management tools. We integrate the widely used Jupyter Notebook UI with automatic system and environment configurations, allowing students to begin core code development with a single click.

Guided by the version management mechanism, the students can practice data analytics and software engineering skills by iteratively updating code and hyperparameters, and can continuously assess the changes in the results (e.g., accuracy). The students can see and even revert to any version in their code development histories. Using a leaderboard mechanism, we encourage students to continuously improve their results by interacting with different versions, further strengthening their parameter-tuning skills. DataLab also enables collaborative learning and group projects by allowing users to perform supervised sharing of code and data.

Instructors can easily create a data science assignment by uploading the datasets, an initial code template, and the grading policy. DataLab provides many instructor tools, including tools for permission management, student code management and results inspection, tools for learning analytics, and integration with MOOC platforms. Moreover, DataLab integrates many popular data processing back-ends such as distributed storage (Hadoop File Systems [14]) and execution frameworks (e.g., Python, R and Spark [16]).

To verify the effectiveness of DataLab, we use DataLab in a graduate-level introductory data science course with 81 students. We ask the students to finish their data science projects online using DataLab. After the assignment project, we receive $1,979$ different versions of students' submissions. We analyze the students' behavior using the logs. The result shows that DataLab can effectively keep students more active and help them to get better learning outcome. We also conduct a survey to get students' subjective feelings of DataLab, and receive positive feedback. Some students request to use DataLab as the engineering tool of their future data science research projects.

To verify the effectiveness of DataLab, we applied it to a graduate-level introductory data science course with 81 students in which the students were asked to complete their data science projects online using DataLab. After this assignment, we received $1,979$ different versions of students submissions. We analyzed the students behavior using the logs. The result shows that DataLab can effectively keep students more engaged and help them to achieve a better learning outcome. We also conducted a survey to obtain the students subjective feelings about DataLab; we received quite positive feedback. Some students even asked if they could use DataLab as the engineering tool for their future data science research projects.

The rest of the paper is organized as follows. We review related work in Section II, describe the key features in user experience design in Section III, and present the system internal design and implementation in Section IV. Then, Section V provides a case study of using DataLab with a real in-classroom data science course. Finally, we conclude the paper and introduce our future work in Section VI.

II. Related Work

In recent years, discussions concerning data science education have rippled out from the emergence of data science courses and programs in universities, as well as from data science MOOCs and online nano degrees. The authors of [30] reveal the practices of a data science program that offers courses in both traditional and MOOC formats and describe how to deliver both traditional lectures and programming laboratories online. In [29], the authors experiences of offering data analytics training programs to both on-campus students and to their potential employers at local organizations are presented. In [31], the authors discuss the creation and implementation of a data science undergraduate degree program based on the experiences of three universities. The authors of [32] show share their experiences in combining data science education with hackathons, encouraging students to solve challenging data science problems to boost the learning outcome.

Data science education, like software engineering, relies heavily on tools. However, there is little research on how to design effective data science education tools. Existing data science courses reuse the simple online IDEs and autograders designed for lightweight code completion tasks, creating a false sense that coding for data science is like introductory level programming. For instance, DataCamp [22] and Codecademy [24] provide data science courses with step-by-step tutorials and code completion tasks which can be autograded instantly. Udacity [21] offers both data science MOOCs and nano degrees and can autograde programming assignments online, but the final projects can only be graded manually. Kaggle [18] is a site for predictive modeling and analytics competitions. Participants submit prediction files and

Kaggle automatically provides real-time scoring and ranking. However, Kaggle lacks development tools for teaching.

DataLab is inspired by and built on top of many existing software engineering and data science tools, such as revision control systems, workflow tools, data management tools, and data science packages. Source code version control and management system have a long history, from CVS [26] to SVN [27] to Git [25]. They are designed to handle modest-sized files and thus not suitable for storing data.

Versioning has been a hot topic in database research recently, particularly around subjects such as arrays [4] and graphs [5]. These works implement elementary operations for comparing differences between versions. Other temporal database systems like [6], [7] provide querying versions with linear chains but do not support other complicated structures of data versions like tree-structured versions.

Some workflow tools like Chimera [8], Pegasus [9] and Vistrails [10] adopt the concept of data workflow similar to the workflow we use in DataLab. However, they all lack a clean separation of raw data, metadata and versions of datasets. Other tools such as Orchestra [11] and Fusion tables[12] use the concept of data collaboration among users, but lack the capability of data version control and management.

Some projects have focused on sharing data. The MIT DataHub [13] [17] project supports dataset revision control, but it does not manage the entire data analytics development cycle. Thus, it is more of a database management tool than a software engineering tool. The Harvard Dataverse [23] is a data publishing and sharing platform but it lacks data version control and analytics. Neither is suitable for use in data science education.

III. Key User Experience Design

DataLab provides a complete set of features that support integrating key software engineering ideas into data science courses. We first provide an overview of a typical workow in DataLab, from both the instructor perspective and the student perspective.

A. Overview of the workflow in DataLab

The DataLab workow design centers around a powerful code and data revision management system. Although DataLab can be used for open-ended course projects, here we focus on a typical homework-assignment workow. In an assignment, the workow includes an instructor part and a student part. We use a consistent UI to integrate both parts. We want to make it easy for instructors to create, manage, grade and reuse the assignments, and we want to let the students focus on the core data science materials while imparting good software engineering practices. Figure 1 provides an overview of this workow, and the following subsections highlight the key features in the workow.

B. Instructor: Creating an assignment

To create an assignment, the instructor needs to set up three parts: the datasets used in the assignment, the initial code template, the instructions, and the grading policy. The instructors can upload and import a dataset into DataLab through a web UI or upload the data using common tools such as FTP. The dataset can be stored on the local file system or in a distributed file system (we currently support the Hadoop File System (HDFS) [14]).

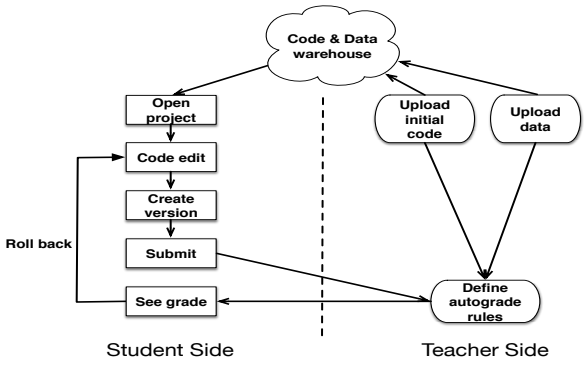

Fig. 1: A typical workflow

The instructor also uploads code templates or samples as public code in the system, and associates this code with the data. Thanks to the adoption of Jupyter-style notebooks, the instructors can combine the code templates and instructions into a single file. Initially, we provide numerous publicly available data science code examples, so it is easy for a new instructor to construct such code templates.

Finally, the instructor needs to write autograding scripts and set up the autograder. For common tasks such as data mining, we provide autograder code that can perform accuracy-based autograding.

C. Student: Doing the assignment online

The DataLab interface for students contains mainly two parts: 1) a version control system and 2) an online integrated development environment (IDE).

a) Starting with the version control UI: Upon starting a new assignment, each student receives a clone of the assignment repository, including both code and data. DataLab avoids copying the data as much as possible; instead, it uses the internal data-versioning tool discussed in the next section.

The first UI a student sees after entering DataLab is a summary page of data, source code and execution records, each of which includes the ability to make explicit version selections. Figure 2 shows the UI. We believe that letting students see this "big picture" page with versions first (instead of taking them directly into the code) serves to remind the students of two things: 1) the connections between code and data, and 2) version management of code and data. On this page, the students can inspect different versions of their code and data.

Note that Figure 2 shows some special parameters and configuration files. We encourage the instructors to extract all the hyperparameters (e.g., the learning rate in a neural network) that a mining algorithm uses into separate files. These hyperparameters are easily accessible through an intuitive API

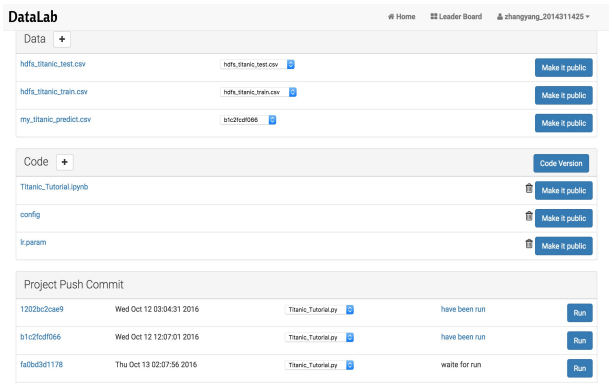

Fig. 2: Jupyter notebook interface of DataLab system

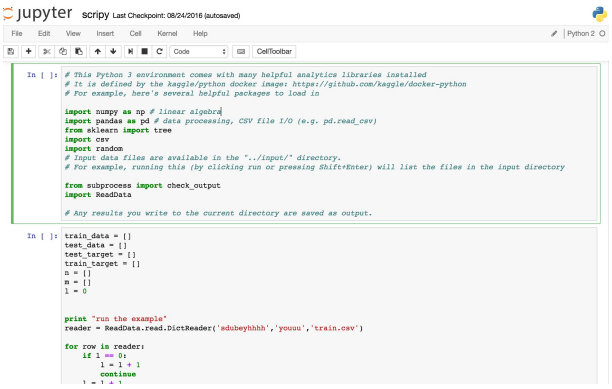

Fig. 3: Jupyter notebook integration in DataLab

in the student code. Although using this feature is not required, adopting the use of a parameter file forces the students to think separately about hyperparameter tuning vs. algorithm implementation, which is a good habit when developing data science code. As an extra benet, separating the hyperparameters allows the system to record the students learning activities more accurately. For example, it can determine whether a student is modifying the code or simply tuning the parameters. Collecting such data can further support data-driven education studies, which is an important aspect of our future work.

b) Online development environment: Students can edit code using an online IDE. Our online IDE is based on Jupyter Notebook [19], an open source, web-based interactive development environment that supports over 40 programming languages. The cloned data and code are automatically loaded into the online IDE with all proper libraries, environment variables and working directories correctly set up. Figure 3 shows a typical Jupyter Notebook interface in DataLab.

The advantage of Jupyter Notebook is that it provides an interactive programming environment that supports a combination of source code, texts, charts, figures and even multimedia files. It is popular among data scientists, and we believe it is the best format in which to provide instructions and code templates for data science students because it allows them to focus on the core data science code rather than on tedious system, package and environment configuration problems.

Students can execute their code directly within the online

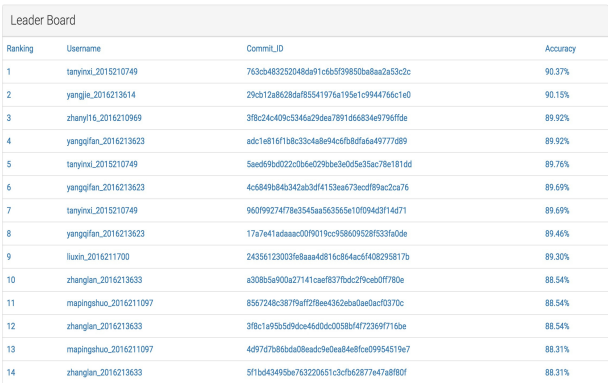

Fig. 4: An example screenshot of the leaderboard

editor UI. We modified the Jupyter backend to separate each users private environment (e.g., data paths, privileges) to ensure that students do not see each others work. We also added an auto-save functionality that periodically saves the students code to a hidden branch in our code versioning system. This feature helps when unexpected failures of the web server or the users browser occur. We use individual resource containers [28] to implement a sandbox for running each students code on a shared server. In this manner, we can provide performance and security in a multi-user environment. Of course, all these complex system configurations are transparent to the students.

c) Creating code/data versions and autograding: When a student is ready to submit a version of her code for grading, she clicks the *push* button on the version interface, at which point the system commits a new version containing the submitted code and parameters. Then, the students can optionally *run* the version, and a corresponding output dataset is created. We also capture all the console log output, save the log file and provide a link on the UI.

After DataLab completes a grading run, the system submits the code, parameters, logs, and a pointer to the resulting datasets to an instructor-dened autograder module for grading. The autograder can choose to run the code on another (secret test) dataset, or just to look at the current results. DataLab provides several widely used autograders for data science. In the case study in V, we use a grader based on the average prediction accuracy on a test dataset. This grader also comes with a global leaderboard that ranks the accuracy achieved for the dataset. The leaderboard has proven to be an effective way to encourage students to continue to update their efforts in our case study. Figure 4 shows a screenshot of a sample leaderboard. Optionally, the leaderboard can integrate with MOOC platforms, and the results can be returned using a RESTful API.

d) Version management tools: Version management is not only a good software engineering habit that the data science students need to learn about, but it also helps the instructors to better understand the students' learning process: how they improve their code and tune the hyperparameters.

On the same summary page, the student can review different versions of their code and the corresponding result datasets.

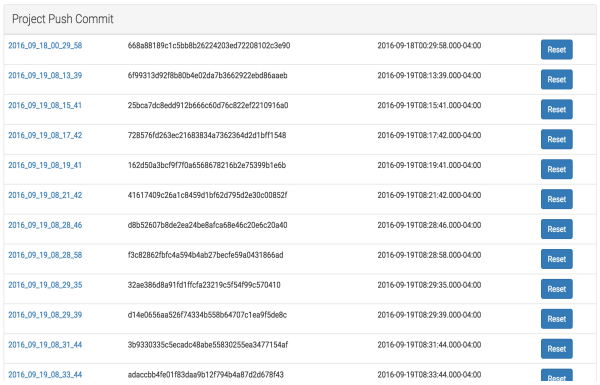

Fig. 5: Code versions

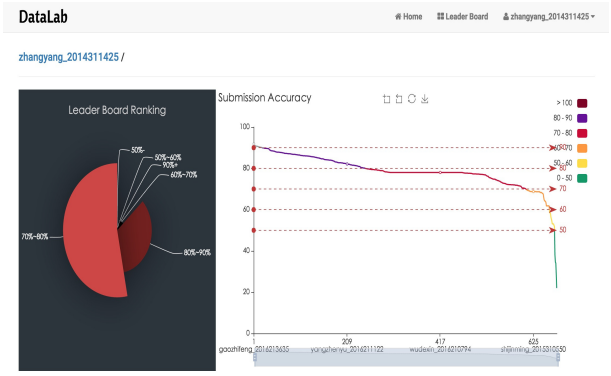

Fig. 6: Statistic interface of instructors

Figure 5 shows an example of these different versions.

Students can also compare the result datasets and the code versions. If they believe their recent changes to the code negatively impact the results (common in data science projects), they can *reset* the code to the previous version with a single click. DataLab ensures that the result data generated by the previous code version is either stored or regenerated, as we will discuss in the next section. Our experience shows that the reset feature is popular among students.

e) Student experience summary: In summary, using the version management tools and autograder as a guide, students can iterate through the process above multiple times, updating their code and hyperparameters to improve the results until they are satisfied. During the process, they can view their progress step-by-step, and the reset feature provides a safety net in the event of a mistake. All these features encourage the students to actively explore solutions for their assignments.

D. Instructor tools

In addition to the assignment creation discussed above, DataLab also provides additional tools for instructors. While we are still extending the instructor tools to provide more functionality, there are four most important tools:

In addition to the assignment creation tool discussed above, DataLab provides additional tools for instructors. We are still extending the instructor tools to provide more functionality, but the four most important tools are already in place:

1) Permission management. The instructor has full control of students permissions to access/share/write/delete files to meet the needs of different types of assignments (e.g., individual vs. group, limited-scope vs. open-ended).

2) Resource management. The instructors can shutdown students' Jupyter or autograding processes if they use too many computation resources without calling the system administrator.

3) Inspection tools for student files. The instructor can inspect code and data by acting as any student.

4) Learning statistics tools. We provide a user defined dashboard with several pre-defined plugins for the instructors to analyze the students' activities on the platform. Figure 6 shows a sample screenshot summarizing the statistics students' submission behavior.

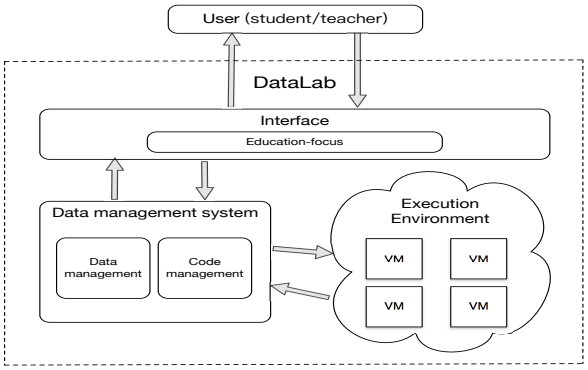

Fig. 7: System overview

IV. System Internal Design

Figure 7 provides an overview of the system architecture.

Underlying the DataLab intuitive UI is a complex internal infrastructure. The core of DataLab is a data management system that integrates the code and data revision control. In addition, we provide a secure and extensible code execution environment based on virtual machine or container technology. The underlying system provides a set of extensible APIs that allow people to build different user interfaces, of which the education-focus interface discussed previously is an important example. Furthermore, to support a large number of students with a moderate amount of computational resources, we extensively exploit the fact that most of the solutions to the assignments are similar; consequently, DataLab includes optimizations that reduce computation and storage overhead using caching and data dependency controls. Figure 7 provides an overview of the system architecture.

In this section, we describe the data management system, the execution engine and the performance optimizations for educational purposes.

A. Data management system

The data management system provides a coherent logical view of versioned code, datasets and execution records (logs). The design goals of the data management system include: 1) efficiency: being able to scale to large datasets at low cost; 2) compatibility: supporting legacy data analytics frameworks

such as Python, R and Spark; 3) extensibility: supporting rich APIs including version control and quick dataset filtering.

We first introduce the logical data model and then discuss the scalable implementation and optimizations for the code and data version management.

a) Logical data model: From a user's point of view, the DataLab data model provides an interface by which users can make queries against different versions of the code (including system environment configurations and hyperparameters), semi-structured datasets and execution records.

All three types of data are linked through a consistent version number, the Git commit ID of the code. In this model, all data are associated with executing some source code that has a unique commit ID.

Like many database systems, DataLab supports creating *logical views* of a dataset. A user can create a (named) view by either executing a query against the semi-structured datasets, or by executing a user generated program. Logical views are the first-class citizens, and the users can share and version them just like a real dataset.

Logical views are essential for both instructors and students. For example, instructors can easily select subsets of samples from larger datasets to create different assignments, and students can save, reuse and share their intermediate results (e.g., preprocessing results after data cleaning). The code is managed with an extended version of GitLab [15] that also includes all standard GitLab API support. The inclusion of the standard API allows the students to (optionally) learn standard Git code management practice.

Our extension to code revision is to separate the system configurations and hyperparameters used in data mining algorithms into two individual files: `config` and `param`. DataLab understands the semantics of these files and provides API callbacks to handle the versioning of these parameters. The students code can access these parameters the same way as it would access command line arguments. Forcing students to specify the parameters in a configuration file not only helps the students to acquire a good programming habit but also help DataLab to distinguish the students parameter-tuning efforts from code development efforts, allowing better evaluation of the students learning outcomes.

Execution logs are also important parts of the logical data and associated with the unique code commit ID. After receiving students submission request and executing their code, DataLab automatically stores the commit ID, the commit time, execution logs, etc. These logs are essential for students to debug their programs and sometimes useful for autograding. Internal to our system, the logs are important for maintaining the data workow discussed in the next subsection.

The user permission system is also managed at the logical data model level. By default, students can only see their own changes, while the instructor can see all changes from all students in the course. To avoid cheating, students cannot share code or data with anyone. The instructor can override these settings, allowing students to share the code globally or within a specific group.

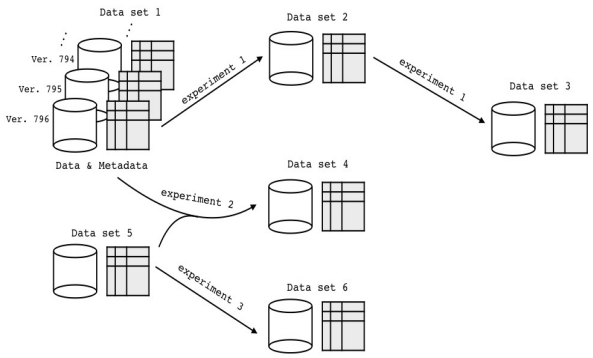

Fig. 8: An example DWF graph

b) Implementing the data versioning - Data work flow: Naively, we need to maintain a number of versions for every single dataset or logical view that students create. To manage these datasets efficiently, we introduce *data work flow (DWF)* to manage the logical relationships of all dataset versions. Two datasets are connected by an edge if one dataset is derived from the other. DWF is the core of DataLab's version management system.

In a DWF, a node represents a particular version of a dataset. A directed edge connects two nodes if one dataset is derived from the other. The labels on the edges show the code version that is used to generate the dataset. Figure 8 provides an example of the DWF. DataLab automatically constructs and maintains DWF using the execution records as students submit their experiments.

We implement these structures by adding a *parent* property to the metadata of the newly generated dataset so that when we process one dataset we can find the parent dataset. Additionally, we implement functions to compare the differences between a dataset or programs and its parent or child dataset. This feature helps students to easily see the consequences of their code changes.

The DWF graph serves three purposes: 1) it is used to allow the system to schedule the evaluations / re-evaluations of certain datasets, 2) it allows students to manage their entire experiment history, including knowing which *version* of the dataset generates which result datasets, and 3) it is used by the caching systems (discussed next) to improve cache effectiveness.

DWF is similar to the concept of data dependency graph that is common in many systems. For example, Spark uses a directed acyclic graph to manage Resilient Distributed Datasets (RDD) [2] while Dryad [3] uses a dependency graph to organize individual partitions and stages in its distributed computation. The purpose of DWF is different: we use it to keep track of the execution history and corresponding database versions instead of to track intermediate data.

c) Scalable physical data management: It is non-trivial to implement the logical data model efficiently because we must maintain all the existing versions in storage and manage a large number of datasets. Here, we introduce the general

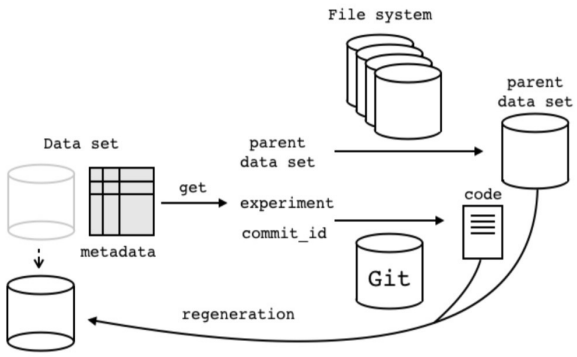

Fig. 9: Reconstruct data set from cached data sets

ideas of our physical data structures.

To manage datasets with a large variety of sizes, we separate the storage of data from that of metadata. We store metadata in a fast NoSQL database for rich query semantics, but store the large data in a distributed file system to reduce storage costs. We link the data and metadata with a system-generated ID.

We store all the students code in a GitLab server, and we use the GitLab API to communicate with it. All the execution records, except for logs from the students program executions, are imported back into the MongoDB as a special system table. The execution engine framework is mapped to the students project directories to simplify the path management for each student.

We treat DWF maintenance as a separate background task. While tracking all the execution records, we can construct the DWF, which helps students to monitor data provenance and understand their data better.

To preserve storage resources, we cannot physically keep all the versions of all the data that students generate. Additionally, if one student runs exactly the same code on the same dataset multiple times (very likely in the exploration phase in an assignment), we do not want to recompute and regenerate the datasets multiple times. Thus, we introduce a *caching* mechanism to manage datasets that we can derive anytime. Specifically, using the parent versions and the generating code version IDs in the DWF, we can recompute a dataset, if the parent dataset is available, and the computation is deterministic. In this case, the system will find the location of the parent version. Given that the dataset generation process is deterministic, we can reconstruct a dataset if it is missing. Otherwise, we can safely reuse the dataset that already exists.

We realize that many data science computations involve random factors that make the executions non-deterministic. We allow the instructors to trade off the actual randomness vs. the caching effectiveness by fixing the seed of the random number generator. The randomness setting is transparent to the students.

DataLab supports storing datasets in different distributed storage backends. For example, we run our platform on a private cloud and store the datasets in Hadoop File System (HDFS) [14]. We also support key-value storage such as Amazon S3 [33] for educational institutions using public clouds. For metadata management, we use scalable NoSQL databases (MongoDB [34] in the current implementation) instead of traditional SQL database.

B. Backend features and execution environment

a) Support for multiple data science tools: Data science projects always need many useful data science tools or packages. We support Python as our primary language given its popularity in data science. Python also provides good Spark and Hadoop[1] support, making it easy to handle large files on distributed file systems. We pre-install these data science tools as part of DataLab so that the students no longer need to configure their own.

b) Scalability to large scale: To meet the demands of multiple users editing online at the same time, we launch a separate Jupyter Notebook server for each student so that they do not conict with each other.

For deployment, we use a single machine as the web server for students and the master node. We can scale the system by adding more servers as slaves for distributed storage and execution. For the sake of security between different students, the system creates a Docker [28] container for each student so that they cannot read and write data files outside the container except through our APIs. Using the Docker container, we can also limit the resources used by each student, in case some buggy or malicious program is using up all the system resources.

By collecting the logs created when students create and manipulate data with our APIs, we can analyze students behaviors to better understand the obstacles they face during learning.

c) Supporting open-ended projects: DataLab also allows students to share a particular version of their code and results from data, making collaboration easy. With the instructors permission, students can publish their code (including the hyperparameters) so that others can import them directly as supplemental code for their projects and see the results. The sharing is efficient because no actual data copying occurs in the system. In an open-ended project setting, we also allow students to upload their own Jupyter Notebooks files into DataLab to start their project.

Because Jupyter Notebook is open-source, we can easily modify it to support multiple users and multiple backends. To ensure performance isolation and security, when students begin editing code, DataLab starts a separate Jupyter Notebook server container for code execution.

C. Extensible APIs

The core of DataLab is designed as a general data science software engineering tool meet different education and research goals. We allow users to design different user interfaces and workflows to serve different purposes. For example, the education interface extends the core API to support autograding and instructor-level permission management, while a

separate research UI focuses on managing scientific datasets. The generality of the DataLab core significantly reduces the development time required for each use case, lowering the cost of developing customizations to meet other education environments (e.g., in future work, we plan to build an environment to teach programming to K-12 students using the same system core). Table I shows the most important APIs.

V. Evaluation in a Real Classroom

In this section, we discuss a real use case of DataLab in a class of 81 students. We let the students complete a data science project and report the students' learning behavior based on analyzing the logs from DataLab. We also conduct a survey to get the students' subjective feelings about DataLab and report the key findings here.

A. Methodology and Experiment Setup

We deploy DataLab in an in-classroom graduate data science course with 81 students. The course targets first-year graduate students and senior undergraduates with diverse engineering backgrounds such as computer science, software engineering, electronic engineering and automation. Because this is an introductory course, most of the students do not have much experience with data science or related fields. We also invite another 20 volunteers who have taken the previous iteration of the course to participate. All the students know about and consent to participation in the study.

For the assignment, we adopt a classic Kaggle [18] competition project: *Titanic: Machine Learning from Disaster*. The dataset contains gender, age, cabin class, and other information about the $2,224$ persons on board the Titanic. After splitting the dataset into training data and testing data, we ask the students to build a machine learning model to predict who will survive in this disaster. There is a single metric for result evaluation: *the prediction accuracy*. Four steps are required to complete the project: data cleaning, feature selection, model selection, and hyperparameter tuning.

We provide a fairly complete code template and the students need to edit their code using our online IDE and complete the steps to clean the data, select features and build a model. They are able to run a small test within the IDE and see the results interactively. After they are satised with the results, they first need to create a new version by *pushing* the code; then, they can request autograding on a separate testing set by *submitting* the code. We do not limit how many times students submits their code.

By tuning the hyperparameters stored in a separate `param` file, such as adopting different features and different models, the students iteratively improve/reduce the prediction accuracy in each version. To encourage the students to continue improving their results, we post their accuracy results to a leaderboard, creating a competitive setting.

For this small-scale assignment, the platform runs on 3 machines which each has 8 cores, 16 GB memory and 80 GB of hard disk storage. We use HDFS for data storage.

We first report the student behavior from analyzing the DataLab logs. We show that with the help of data and code versioning, students achieve much better prediction accuracy. Then, we report findings from the student survey. We consider the first part a more objective evaluation of how DataLab affects students learning, while the second part shows the subjective view from the students.

B. Learning Behavior through Log Analysis

We analyze users' behavior logs to figure out how the students learn data science techniques and iteratively improve their algorithms. We show that DataLab can help them to learn more efficiently.

The system receives a total of $1,979$ different versions of code submissions, and the prediction accuracy from all valid submissions (run to completion without crashing) ranges from 78% to 91.83%.

Figure 10 shows the box plot of the number of times each student submits her assignments for autograding and accuracy ranking. We find that the students push their code 37 times on average with a median of 24 times. They submit an average of 42 times with a median of 24 times. Given that this is a simple assignment, we are surprised to see these many submissions. We believe a combination of the easy-to-use UI and the competition created from the leaderboard both help the student to become more engaged.

Now we want to know whether more submissions help the students to learn; otherwise, added submissions are just a waste of time. First, Figure 14 shows the relationship between the average prediction accuracy and the first 50 submissions of a sample student. It shows that the student does continue to improve accuracy results with more submissions. Sometimes the accuracy is reduced, but she can soon improve it again.

Then we analyze the relationship between students' ranking on the leaderboard with their number of activities on the platform. Figure 12 shows that students with a higher rank (top 25% on the leaderboard) submit more often than those with a lower rank (bottom 25% on the leader board). This correlation indicates that more submissions do help students to think more about the problem and find better solutions.

Many students check the historical branches that they submit, but not all of them use the *reset* function. Figure 11 shows that students check their branches four times on average. There is a large variation in the frequencies with which students use the reset function, some students use it much more than others.

To show whether using reset and branches do help improve students learn and train their techniques, we again compare the behavior of the high rank group and the low rank group. We find that more than 50% of the students with higher ranks use the reset functions at least once. In comparison, in the low rank group, less than 25% used the function. Figure 13 shows the data.

C. Survey Results

We design and conduct a survey containing 18 questions to collect their subjective opinions about DataLab, and we collect

Core APIs			
name	functionality	input	output
create	create a project	dataset name	null
upload	upload a dataset to a project	file or directory name	null
import	import a dataset to the project	file or directory name	null
push	create a new version of code and data	project name	commit ID
submit	submit code to system and execute	commit ID	null

TABLE I: Core APIs

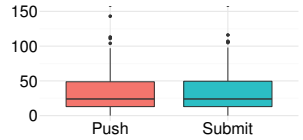

Fig. 10: How many times did students push and submit their code?

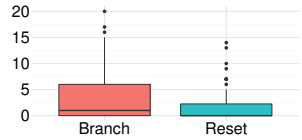

Fig. 11: How many times did students check branches and reset their code

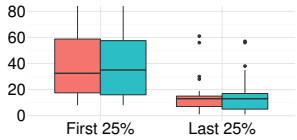

Fig. 12: How many times did students push and submit their code given their ranks?

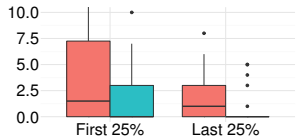

Fig. 13: How many times did students check branches and reset their code given their ranks?

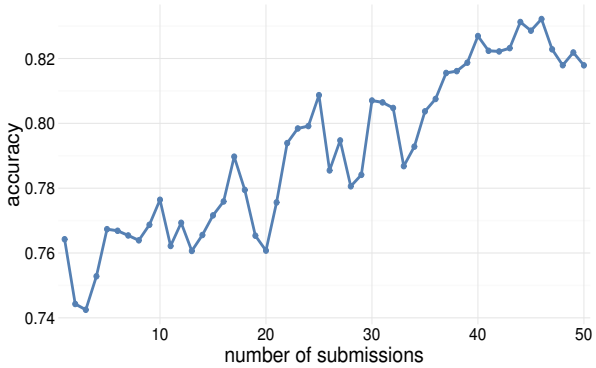

Fig. 14: Result accuracy v.s. the number of trails

all the 101 questionnaires from the 81 students enrolled in the course and other 20 non-enrolled volunteers.

The survey has three parts: 1) Questions about the students' coding experience, for example, what proportion of time do they usually spend on debugging; 2) Questions about the students' opinions DataLab, such as whether DataLab can helps them learn parameter tuning; and 3) Students suggestions about ways that DataLab can be further improved.

We conduct the survey through a third-party online survey website; each student can submit only one response. From the survey results, we find that students usually spend 50% of time on average on debugging in a project. They agree that editing and managing code online is a much more convenient and efficient way for code development. A large majority of these students acknowledge that problems stemming from having to manage many versions of code and data used to confuse them when coding (Figure 15). Many acknowledge that they had experienced bugs due to these confusing versions. Figure 16 shows how often the students face the problem of confusing data versioning.

Most students also agree that DataLab is easy to use and can improve the effectiveness and efficiency of managing code and data versions for data science projects. For collaboration, the students all agree that DataLab's sharing features are helpful for collaborating within a data science team (Figure 19).

After completing this assignment, students give a positive rate on DataLab system, and 92 out of 101 students indicate that they will continue to use DataLab for their future data science projects.

Some students suggest that it will be better if old versions can be removed, and some hope to use a more powerful online IDE than Jupyter Notebook.

We do not allow students to upload their own data in this assignment; however, after completing the assignment, a few students reach out to us and ask for permission to perform their own research on the DataLab platform, further demonstrating the positive experiences with the system. Consequently, in future work, we plan to add open-ended support for collaborative projects.

VI. Conclusion and Future Work

Having taught an introductory data science course for the past three years, we are surprised to see how poorly data science students manage their development process, especially *ad hoc* naming for versioning, defining hyperparameters as global variables, and the lengthy process of setting up an environment to run simple tasks such as a `word count`. As system builders, we build a tool to tackle the challenge of introducing software engineering thinking into data science education, making data science courses more understandable and enjoyable. DataLab manages different versions of data, code and execution records and helps students to improve their development efficiency while maintaining a focus on the core algorithms in data science. We build the entire system on scalable frameworks, allowing DataLab scale to a large number of students at low cost. In a real classroom study, behavior analyses and survey results show that DataLab helps to improve the effectiveness and efficiency of data science education.

For our future work, we realize that there is no data-driven methodology to evaluate the learning process of data science

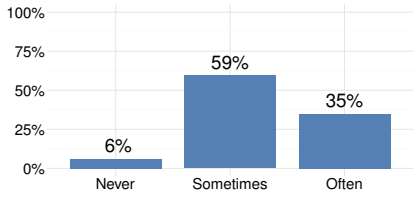

Fig. 15: How often do students get confused with the problem of code versions?

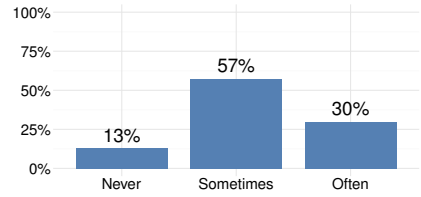

Fig. 16: How often do students get confused with the problem of data versions?

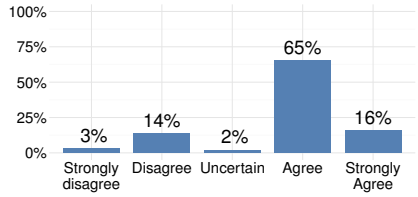

Fig. 17: Is DataLab helpful for managing code versions?

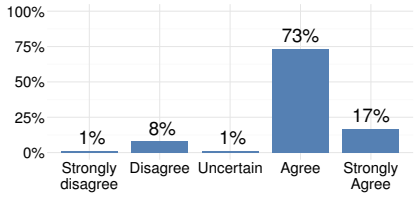

Fig. 18: Is DataLab helpful for managing data versions?

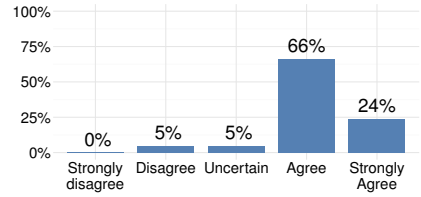

Fig. 19: Is DataLab helpful for teamwork?

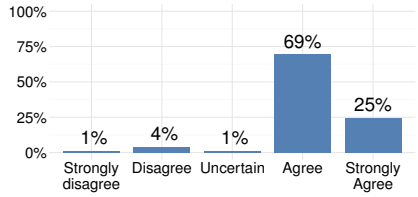

Fig. 20: Is DataLab helpful for learning data analysis techniques?

techniques. We would like to develop such a technology using the rich monitoring data that DataLab collects.

VII. Acknowledgement

This research is supported in part by the National Natural Science Foundation of China (NSFC) Grants 61361136003, 61379088, China 1000 Talent Plan Grants, Tsinghua Initiative Research Program Grants 20151080475, and a Google Faculty Research Award.

References

[1] White T. Hadoop: The definitive guide[M]. " O'Reilly Media, Inc.", 2012.
[2] Zaharia M, Chowdhury M, Das T, et al. Resilient distributed datasets: A fault-tolerant abstraction for in-memory cluster computing[C] Proceedings of the 9th USENIX conference on Networked Systems Design and Implementation. USENIX Association, 2012: 2-2.
[3] Isard M, Budiu M, Yu Y, et al. Dryad: distributed data-parallel programs from sequential building blocks[C]//ACM SIGOPS Operating Systems Review. ACM, 2007, 41(3): 59-72.
[4] Seering A, Cudre-Mauroux P, Madden S, et al. Efficient versioning for scientific array databases[C]//2012 IEEE 28th International Conference on Data Engineering. IEEE, 2012: 1013-1024.
[5] Khurana U, Deshpande A. Efficient snapshot retrieval over historical graph data[C]//Data Engineering (ICDE), 2013 IEEE 29th International Conference on. IEEE, 2013: 997-1008.
[6] Salzberg B, Tsotras V J. Comparison of access methods for time-evolving data[J]. ACM Computing Surveys (CSUR), 1999, 31(2): 158-221. MLA
[7] The TSQL2 temporal query language[M]. Springer Science & Business Media, 2012.
[8] Foster I, Vockler J, Wilde M, et al. Chimera: A virtual data system for representing, querying, and automating data derivation[C]//Scientific and Statistical Database Management, 2002. Proceedings. 14th International Conference on. IEEE, 2002: 37-46.
[9] Deelman E, Singh G, Su M H, et al. Pegasus: A framework for mapping complex scientific workflows onto distributed systems[J]. Scientific Programming, 2005, 13(3): 219-237.
[10] Callahan S P, Freire J, Santos E, et al. VisTrails: visualization meets data management[C]//Proceedings of the 2006 ACM SIGMOD international conference on Management of data. ACM, 2006: 745-747.
[11] Ives Z G, Khandelwal N, Kapur A, et al. ORCHESTRA: Rapid, Collaborative Sharing of Dynamic Data[C]//CIDR. 2005: 107-118.
[12] Madhavan J, Balakrishnan S, Brisbin K, et al. Big Data Storytelling Through Interactive Maps[J]. IEEE Data Eng. Bull., 2012, 35(2): 46-54.
[13] Bhardwaj A, Bhattacherjee S, Chavan A, et al. Datahub: Collaborative data science & dataset version management at scale[J]. arXiv preprint arXiv:1409.0798, 2014.
[14] Shvachko K, Kuang H, Radia S, et al. The hadoop distributed file system[C]//2010 IEEE 26th symposium on mass storage systems and technologies (MSST). IEEE, 2010: 1-10.
[15] GitLab. http://gitlab.com
[16] Zaharia M, Chowdhury M, Franklin M J, et al. Spark: cluster computing with working sets[J]. HotCloud, 2010, 10: 10-10.
[17] Bhardwaj A, Deshpande A, Elmore A J, et al. Collaborative data analytics with DataHub[J]. Proceedings of the VLDB Endowment, 2015, 8(12): 1916-1919.
[18] Kaggle. http://www.kaggle.com
[19] Shen H. Interactive notebooks: Sharing the code[J]. Nature, 2014, 515(7525): 151-152.
[20] 25 Best Jobs in America. https://www.glassdoor.com/List/Best-Jobs-in-America-LST_KQ0,20.htm
[21] Udacity. https://www.udacity.com
[22] Datacamp. https://www.datacamp.com
[23] Harvard dataverse. https://dataverse.harvard.edu/
[24] Codecademy. https://www.codecademy.com/
[25] Git. https://git-scm.com/
[26] CVS. http://www.nongnu.org/cvs
[27] SVN. https://subversion.apache.org/
[28] Merkel D. Docker: lightweight linux containers for consistent development and deployment[J]. Linux Journal, 2014, 2014(239): 2.
[29] Maija Marttila-Kontio, Mikko Kontio, and Virpi Hotti: Advanced data analytics education for students and companies[C]. In Proceedings of the 2014 conference on Innovation and technology in computer science education (ITiCSE '14). ACM, 2014.
[30] Geoffrey Fox, Sidd Maini, Howard Rosenbaum, David Wild: Data Science and Online Education[C]. Proceedings of IEEE 7th International Conference onCloud Computing Technology and Science (CloudCom). IEEE, 2015.
[31] Paul Anderson, James McGuffee, and David Uminsky: Data science as an undergraduate degree[C]. In Proceedings of the 45th ACM technical symposium on Computer science education (SIGCSE '14). ACM, 2014.
[32] Craig Anslow, John Brosz, Frank Maurer, and Mike Boyes: Datathons: An Experience Report of Data Hackathons for Data Science Education[C]. In Proceedings of the 47th ACM Technical Symposium on Computing Science Education (SIGCSE '16). ACM, 2016.
[33] Palankar M R, Iamnitchi A, Ripeanu M, et al. Amazon S3 for science grids: a viable solution?[C]//Proceedings of the 2008 international workshop on Data-aware distributed computing. ACM, 2008: 55-64.
[34] Membrey P, Plugge E, Hawkins D. The definitive guide to MongoDB: the noSQL database for cloud and desktop computing[M]. Apress, 2011.