

# Pedagogy, Infrastructure, and Analytics for Data Science Education at Scale

*Vinitra Swamy*



Electrical Engineering and Computer Sciences  
University of California at Berkeley

Technical Report No. UCB/EECS-2018-81

<http://www2.eecs.berkeley.edu/Pubs/TechRpts/2018/EECS-2018-81.html>

May 19, 2018

Copyright © 2018, by the author(s).  
All rights reserved.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission.

---

**Pedagogy, Infrastructure, and Analytics for Data Science Education at Scale**

by Vinitra Swamy

---

**Research Project**

Submitted to the Department of Electrical Engineering and Computer Sciences,  
University of California at Berkeley, in partial satisfaction of the requirements for the  
degree of **Master of Science, Plan II.**

Approval for the Report and Comprehensive Examination:

**Committee:**

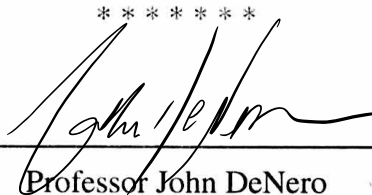


Professor David E. Culler  
Research Advisor

5/9/18

(Date)

\*\*\*\*\*



Professor John DeNero  
Second Reader

5/16/18

(Date)

**Pedagogy, Infrastructure, and Analytics for Data Science Education at Scale**

by

Vinitra Swamy

A thesis submitted in partial satisfaction of the  
requirements for the degree of  
Master of Sciences

in

Computer Science

in the

Graduate Division

of the

University of California, Berkeley

Committee in charge:

Professor David E. Culler, Chair  
Professor John DeNero

Spring 2018

I dedicate this report to aspiring computer scientists and data scientists of generations past, present, and future. Acclaimed researcher Edsger W. Dijkstra (best known for his shortest-path graph traversal algorithm) once said, “Computer science is no more about computers than astronomy is about telescopes.” Explore data that can make a difference.  
Change the world, one line of code at a time!

# Contents

<b>Contents</b>	<b>ii</b>
<b>1 Introduction</b>	<b>2</b>
<b>2 Background: Data 8 and Data 8X</b>	<b>4</b>
2.1 Related Work . . . . .	4
2.2 Data 8 . . . . .	4
2.3 The Current State of Technology MOOCs . . . . .	6
2.4 Data 8X . . . . .	7
<b>3 User Workflow, System Architecture, and Integration with Learning Tools</b>	<b>9</b>
3.1 User Experience . . . . .	9
3.2 System Architecture . . . . .	10
3.3 Usage Analysis . . . . .	10
3.4 Instructor Workflow . . . . .	16
3.5 Surrounding Suite of Learning Tools . . . . .	16
<b>4 Resource Scaling</b>	<b>20</b>
4.1 Capacity Planning . . . . .	20
4.2 Cost Estimates and Analysis . . . . .	20
4.3 Performance testing . . . . .	23
4.4 Maintenance and monitoring . . . . .	24
<b>5 Grading and Evaluation</b>	<b>26</b>
5.1 OK: Immediate feedback on tests . . . . .	26
5.2 LTI Standard: posting grades to EdX . . . . .	27
<b>6 Analytics and Prediction in Jupyter-based computing environments: Deep Knowledge Tracing</b>	<b>29</b>
6.1 Knowledge Tracing Approaches . . . . .	29
6.2 Related Work . . . . .	30
6.3 Data Context . . . . .	30
6.4 Model Methodology . . . . .	31

6.5	Expansions and Future Areas of Exploration . . . . .	32
6.6	Note on Results . . . . .	33
6.7	Chapter Acknowledgements . . . . .	33
<b>7</b>	<b>Future Work: where to next?</b>	<b>34</b>
7.1	Future Work: Autograding . . . . .	34
7.2	Future Work: Sharing the Knowledge . . . . .	34
7.3	Future Work: Deep Knowledge Tracing Models for Code Progression . . . . .	34
<b>8</b>	<b>Conclusions</b>	<b>35</b>
8.1	First Iteration of the MOOC . . . . .	35
8.2	Data 8 around the world . . . . .	35
8.3	Useful Resources / Links . . . . .	35
	<b>Bibliography</b>	<b>36</b>

## Acknowledgments

I would like to thank my advisor Dean David E. Culler for his vision, guidance, and continual belief in my abilities.

My family – Swami Ramachandran, Bhuvana Ramachandran, Varsha Swamy, and Jishnu Swamy – for their steadfast love, support, and encouragement.

Professor Ani Adhikari, Professor John DeNero, and Professor David Wagner for being incredible teaching mentors and instrumental in the success of UC Berkeley’s Data 8.

Professor Zach Pardos of the Machine Learning in Education course and Professor Marti Hearst of Intelligent User Design for Educational Technology course for their guidance on various aspects of this paper.

Yuvi Panda, Ryan Lovett, and Gunjan Baid of the Division of Data Sciences Infrastructure Team. Yuvi’s role as Operations Architect was especially crucial to the successful implementation of many technologies described in this report.

Allen Guo, Samuel Lau, Wilton Wu, and Madeline Wu for their helpful contributions to the deep learning predictive model in Chapter 6.

My wonderful friends in the UC Berkeley Data Science and Computer Science community, Division of Data Sciences, and Data 8 Course staff. You have made my three years here truly special!



## Abstract

Pedagogy, Infrastructure, and Analytics for Data Science Education at Scale

by

Vinitra Swamy

Master of Sciences in Computer Science

University of California, Berkeley

Professor David E. Culler, Chair

This report presents an educational computing environment for data science education at scale, highlighted in use at the University of California, Berkeley. With the rise of online learners in massively open computing courses (MOOCs), we detail a relevant technical case study of the decisions made in converting an introductory undergraduate data science course into a series of data science edX MOOCs. The focus of this study is on the student and instructor workflow, distributed system infrastructure, cost analysis, cloud resource allocation, and autograding integration in the scaling process. We implement an analytics pipeline for collecting data from Jupyter notebooks and propose a Deep Knowledge Tracing modification to model student progress on coding assignments.

# Chapter 1

## Introduction

Demand for courses teaching programming fundamentals has increased rapidly in classrooms online and around the world. With the popularity of these new courses comes a new set of challenges – how to enable educational computing environments at scale. This report focuses on the pedagogy, technical infrastructure, and analytics aspects of data science education at the University of California, Berkeley.

UC Berkeley’s flagship introductory data science course, Data 8 (also known as *Foundations of Data Science*), is designed to be accessible to students without a background in statistics and computing [1]. As of 2018, it serves a diverse set of  $\sim 1000$  students each semester from over 70 majors.

The Data 8 course design aims to avoid the system administration barrier of cryptic installation error messages that students often face when setting up their computing environment. We believe this to be one of the major barriers to a diverse computing classroom. In order to avoid the difficulty of managing local installations at scale, the Division of Data Sciences infrastructure team has developed a platform to enable a fully functional open source data science environment for students in the cloud. Students need nothing more than a web browser to have access to a world of computation at their fingertips. The resulting architecture outlined in this paper relies on Jupyter’s cell-based Python execution environments, JupyterHub and Kubernetes for management of thousands of student pods, and Jupyter notebook extensions that work with GitHub to enable instructor workflow [2, 3, 4].

The UC Berkeley Data Science education stack, as well as the relevance of the course material, has been crucial in making Data 8 the fastest growing class in UC Berkeley’s history. The skills covered in Data 8 appeal to the broad population of aspiring data scientists around the world. In order to address this demand, the course instructors have released the Data 8 materials publicly under a Creative Commons license, as well as developed a massively open online course (MOOC) version of Data 8 on EdX. This paper addresses the engineering challenges of enabling a data science course at the scale of a MOOC and highlights the first time JupyterHub has been scaled to 100,000 user capacity for education.

Throughout the next seven chapters, we make the following contributions:

1. Outline the architecture of the educational computing environments in Data 8 and Data 8X.
2. Explain the integration of Jupyter infrastructure and user workflow with the EdX platform.
3. Discuss the scaling of resources and capacity planning in the context of Data 8X.
4. Detail an autograding and evaluation scheme using OK, Gradescope, and other learning platforms.
5. Propose a deep knowledge tracing analytics model for prediction of student computing performance.

We begin with a discussion of Data 8 and Data 8X, UC Berkeley's thousand student data science course and MOOC online counterpart.

## Chapter 2

# Background: Data 8 and Data 8X

### 2.1 Related Work

This report provides a relevant technical case study for instructors who are interested in scaling a computing course to a MOOC. Educational computing environments have a long history, arguably beginning with the family of LOGO programming languages from Wally Feurzeig and Danny Bobrow at BBN Labs and Seymour Papert at MIT [5].

Breslow et. al's work on analytics for edX's first MOOC (MIT 6.002x: Circuits and Electronics) and its implications for digital learning has led the way for a wide variety of educational literature analyzing MOOC learners [6]. Further studies discuss scaling programming education in terms of MOOC learner analytics [7] and MOOC team development strategy [8], but do not focus on the development of student computing environments used within the context of learning platforms.

This report focuses on an innovative use case in the space of computing education at scale for Jupyter (cell-based python execution environment) [2], Kubernetes (a container management system) [3], and Docker (containerization for distributed applications) [9]. We extend the scientific and research use cases of JupyterHub discussed by Fernandez et. al by highlighting the implementation of JupyterHub in the education context [10].

### 2.2 Data 8

In this chapter, we present UC Berkeley's introductory data science course, Data 8. This course hosts over a thousand students a semester and introduces programming fundamentals, statistical inference, and prediction techniques.

The initiative began with a charge from the desk of the Chancellor and Provost to the Data Science Education "Rapid Action Team" of June 2014. The focus of the team was to "... [rethink] at a fundamental level what every educated person must know about quantitative reasoning: how to effectively understand, process and interpret information, to inform

decisions in their professional and personal lives and as citizens of the world in the 21st century.”

The result of this initiative is a data science course with a 50-50 gender ratio, students from all years and skill levels, and no prerequisite requirements. After a 100-student pilot semester in Spring 2015, demand for Data 8 enrollment has grown rapidly. The online textbook, assignments, and lecture materials can be found freely available to the public [1].

Students complete assignments using Jupyter notebooks, a cell-based Python execution environment, with problem descriptions and starter code [11]. Through a combination of technologies including Kubernetes, Docker, and JupyterHub, students are only required to have a web browser to access their Jupyter environment. The adoption of Jupyter allows programmers to display their code alongside their resulting tables and visualizations, especially useful in the pedagogical data science context. Using Python within Jupyter notebooks has rapidly gained popularity from the data science and research community, and is quickly becoming the industry standard platform for data science analysis [12].

## Data 8 Course Structure

Data 8 has three hours of lecture a week with a two hour weekly lab section, 12 weekly homework assignments, and three larger projects. These are reflected in the course resource allocation in Figure 2.1. Weekly lab sections have a ratio of 27 students to one graduate student instructor and allow students to make personal connections with instructors in a large course.

With large enrollment numbers, UC Berkeley had to address how to make class sizes in the thousands feel small. The answer is reflected in Data 8’s support resource strategy in Figure 2.1. While the lecture halls seat hundreds, the teaching staff create a personal experience for students in the course with over 24 weekly office hours for one-on-one help, weekly small group tutoring sessions (5 students and 1 tutor), and tailored exam review lectures. Guerilla sections are optional small topical worksheet-based review sessions offered by tutors once major course topics are covered. Project ”parties” allow students to work in the same room and ask staff members questions close to deadlines. With this wide array of support resources, students in a 1000-person computing course can connect with their individual staff members.

The scale of staffing for these support resources is only possible through the enthusiasm of former students in the class. An undergraduate who has performed very well in Data 8 and enjoys the course material can join the course staff as a lab assistant. Lab assistants assist graduate student instructors by answering student questions in a weekly lab section for 1 academic Pass/No Pass unit. After an interested undergraduate has maintained a role as a lab assistant for a few semesters, they can advance to a paid position as a course tutor. Tutors assist with grading written responses on assignments, holding office hours, and conducting small group tutoring sections. Experienced tutors can eventually advance to positions as (under)Graduate Student Instructors, holding a weekly lab section and helping



Figure 2.1: Data 8 Course Structure

out with various pedagogical and logistical aspects of running the course. This hierarchical course staff structure is enabled by the careful leadership of Head GSIs and Professors.

## UC Berkeley’s Data Science Course Ecosystem

Students who have taken Data 8 have the opportunity to advance their skills in data science follow-on courses Data 100 and Probability 140 [13, 14]. Data 100 offers a practical introduction to exploratory data analysis, machine learning, and scalable data processing. Probability 140 delves further into the randomness and sampling variability introduced in Data 8, and focuses on the theory of data science. Both Probability 140 and Data 100 prepare students for advanced upper division courses in data management (CS 186), machine learning (CS 189) and statistics (STAT 154). Beyond these classes, there are many relevant extensions at UC Berkeley in the form of 2 unit data science application seminars (known as data science connector courses), and throughout the departments of Statistics, Electrical Engineering and Computer Sciences, and Industrial Engineering and Operations Research.

Data 8’s course structure mirrors the structure of many of UC Berkeley’s undergraduate computer science course offerings. However, when expanding to the scale of tens of thousands of students in online educational offerings, this in-person staff support structure is not the right fit. The following section proceeds to discuss the state of technology MOOCs, as well as the benefits and concerns of teaching computing on an online medium at scale.

## 2.3 The Current State of Technology MOOCs

MOOCs are an effective medium of transmitting information to increasing numbers of learners online. According to a report analyzing the 2017 MOOC landscape, there are 81 million MOOC learners, covered primarily in the top 5 MOOC providers: Coursera (30 million users), edX (14 million users), XuetangX (9.3 million users), Udacity (8 million users), and FutureLearn (7.1 million users) [15]. The technology category of courses has the highest growth rate, with an increase by 2 percent to 19.9% of the MOOCs released in 2017 [15].

These courses come with a set of challenges and requirements for instructors. In 2013, the Chronicle of Higher Education surveyed 103 professors who had taught MOOCs. The report states that a typical professor spends over 100 hours on their MOOC before it is launched, using most of the time to record online lecture videos. However, this range is widely variable as other instructors' pre-class preparation consists of "a few dozen hours." Once the MOOC has been released and has active students, professors then spend 8 to 10 hours per week on the course, including participation in discussion forums [16].

The medians of MOOC class enrollment for the 103 professors are: 33,000 students enrollees; 2,600 passing; and 1 teaching assistant helping with the class. 74% of the classes used automated grading, and 34% used peer grading. 97% of the instructors used original videos, 75% used open educational resources and 27% used other resources [16]. These figures help estimate the instructor time and enrollment expectations for launching a MOOC.

MOOC enrollment retention and drop-out rates are two of the major challenges facing this medium. One example is the course Bioelectricity, launched in the Fall of 2012 from Duke University. 12,725 students enrolled in Bioelectricity, but only 7,761 ever watched a video, 3,658 attempted a quiz, 345 attempted the final exam, and 313 passed, earning a certificate [17, 18]. A list of other MOOC-specific challenges compiled by instructors in the crowdsourced MOOC Guide are reflected below [19].

1. Relying on user-generated content can create a chaotic learning environment
2. Digital literacy is necessary to make use of the online materials
3. The time and effort required from participants may exceed what students are willing to commit to a free online course
4. Once the course is released, content will be reshaped and reinterpreted by the massive student body, making the course trajectory difficult for instructors to control
5. Participants must self-regulate and set their own goals
6. Language and translation barriers

Specifically relevant in the context of computing MOOCs are challenges surrounding the digital literacy required to address course content, time and effort necessary for participants to successfully complete a lab assignment, and translation barriers in learning technical jargon. The design of the MOOC highlighted in this report aims to address these issues.

## 2.4 Data 8X

We proceed by introducing Data 8X, the MOOC version of Data 8 [20]. Data 8X is comprised of three 5-week Foundations of Data Science courses: Computational Thinking with Python, Inferential Thinking by Resampling, and Prediction and Machine Learning. Enrollment in

Data 8.1X, Computational Thinking with Python, has reached over 45,000 learners with over 6,500 active weekly users. Launched on April 2, 2018 and completed on May 14, 2018, Data 8.1X has finished its formative iteration. Data 8.2X and Data 8.3X will be launched on May 21, 2018 and July 9, 2018 respectively.

While finalizing the syllabus for Data 8X, course architects were limited by the EdX guideline of a three to six hour weekly commitment for online learners. After careful analysis of the material from Data 8, the Professors chose to remove the projects and weekly homework assignments, structuring the course around the weekly lab assignments instead. All three courses are freely available and open to the public, but require a fee for certification (as is traditional among EdX courses).



## Chapter 3

# User Workflow, System Architecture, and Integration with Learning Tools

### 3.1 User Experience

The mission of the Data 8 system architecture is to allow students to focus on learning data science instead of “how to install software” during the first few weeks of the course. We outline the typical user procedure in the section below.

#### Data 8 Student Workflow

Computing courses that use programming assignments, Jupyter, or other computing environments typically have the following workflow.

1. Student downloads a Jupyter notebook (.ipynb file) or set of code files (.py, .java, etc.)
2. Student completes the assignment by writing various sections of code
3. Student re-uploads the notebook or file through some online platform

In this scenario, students would have to download and install an environment (e.g. Python, a text editor, and various packages). This is a major issue for students and instructors, who have to support the issues students run into.

We provide a major simplification to this workflow: students click a link that redirects them to their hub and work directly on Jupyter in the browser. They do not lose any major functionality that could have accessed locally and can customize their Jupyter environment by importing any Python packages they need for their computation.

## Data 8X Student Workflow

Data 8X’s user workflow works in a similar way – students log in to the course on EdX, then navigate to any relevant assignment. In that assignment, they will find a “launch” button, which will enact the following steps:

1. Authenticate the user using the Learning Tools Interoperability (LTI) standard, assigning them an arbitrary user number
2. Generate an nbinteract link to pull a copy of the IPython notebook file and related file dependencies from a public Git repository into the student’s hub
3. Open the correct notebook in the browser

The end result is that in a few seconds, the student is directly editing a Jupyter IPython notebook in their browser and has a live server.

## 3.2 System Architecture

### Data 8 Infrastructure

The architecture for Data 8’s JupyterHub servers is outlined in Figure 3.1. The user prompts the system by clicking on a datahub link for a certain assignment. This link hits the proxy server, where students are asked to authenticate with OAuth, or confirmed as an existing user provided with a unique current ID. Once authentication is complete, Kubernetes creates a new Docker instance of Jupyter, and mounts that to the user’s specific disk. This then is sent back to the user, as they are free to access the Jupyter notebook environment. The culler in Figure 3.1 is responsible for deleting inactive nodes.

### Data 8X Infrastructure

Data 8X expands on Data 8’s architecture by using Kubernetes to cluster multiple hubs, each of which has a structure similar to Data 8’s datahub. The user only interacts with the outer edge proxy of Figure 3.2, which routes the user’s HTTP request to the correct Kubernetes cluster. From there, the inner edge proxy layer routes the user’s request to correct JupyterHub.

## 3.3 Usage Analysis

We proceed by examining the usage data from DataHub, the JupyterHub used for Data 8, for the Spring 2018 Semester. The goal of this section is to help provide an accurate picture for educational hub usage on the scale of 1000 students.

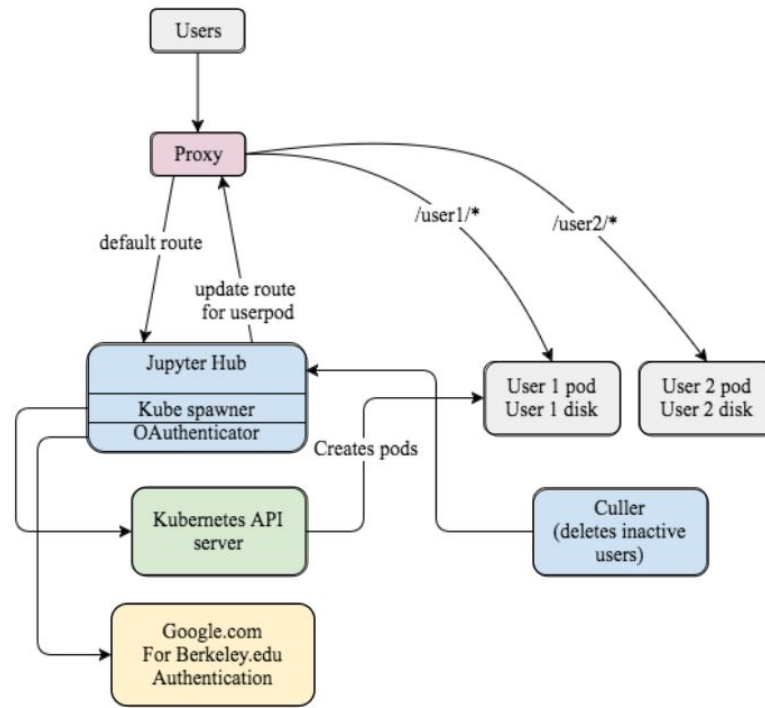


Figure 3.1: System Architecture of Data 8's DataHub

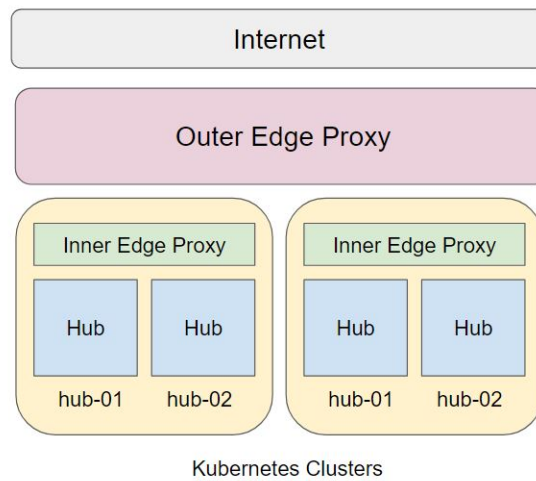


Figure 3.2: System Architecture of Data 8X

### Data 8's Spring 2018 DataHub Usage

In Figure 3.3, we look at time series data from January 17, 2018 through April 12, 2018 each of the individual spikes representing a day's usage. This graph was created through analysis

of pod stops and starts by hashed user ID to calculate a current representation of datahub state (unique active pods) at each hourly timestamp. The weekly spikes are likely due to lab sections taking place on Wednesday afternoon, the majority of Thursday, and Friday morning. During these three days each week, most students in the class will be completing their lab assignments under the guidance of a lab instructor.

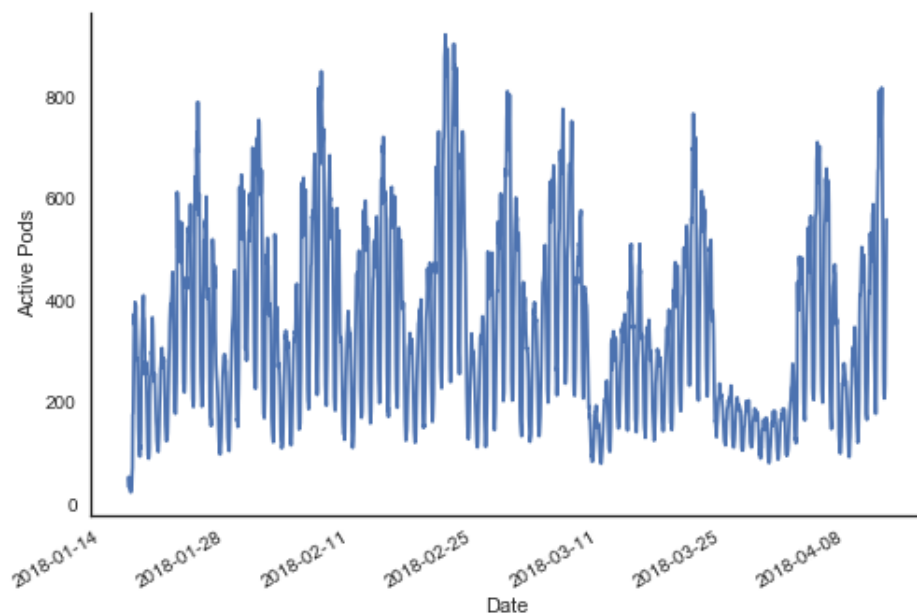


Figure 3.3: Data 8 Datahub Active Pods (Jan 17 - Apr 12)

The distribution of number of pods started by each unique user over the three month period is reflected in Figure 3.4, as a kernel density plot fit over a histogram. This figure has an expected long right tail as few users stop and start over 150 pods, and many instead create new pods about twice or three times a week. Each time a new pod is created, it means that the student has logged out of the system for long enough that the culler has determined them to be inactive, or they have manually logged out.

Diving deeper into specific weeks of the semester, we examine the two-week period that Project 1 was released in Data 8 (February 12, 2018 through February 23, 2018). Project 1, *World Progress*, is an exploration of world population, fertility rates, and child mortality rates using data from Gapminder’s Systema Globalis [21]. Looking closely at Figure 3.5, we see times in which 900 users are simultaneously using datahub as the February 23 deadline looms closer.

Data 8 Project 2, *Crime and Punishment*, is a good example of a heavy datahub usage period in the course, as students are expected to finish the project alongside a homework

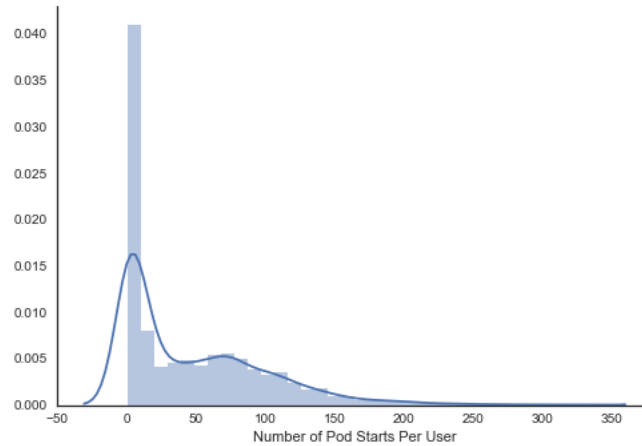


Figure 3.4: DataHub Distribution of Total Number of User Pod Starts (Jan 17 - Apr 12)

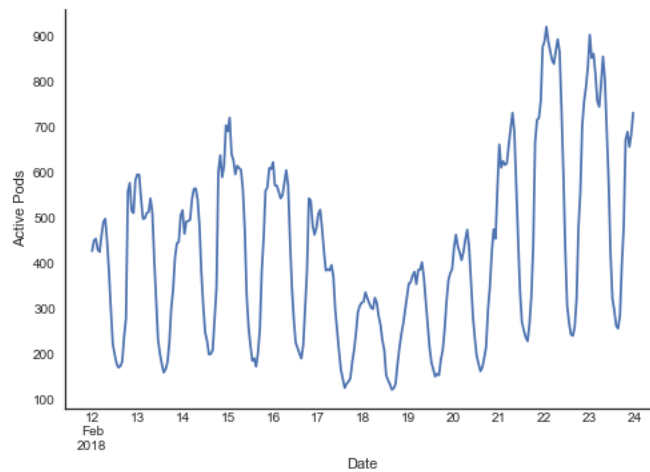


Figure 3.5: Data 8 Datahub Active Pods during Project 1 Weeks (Feb 12 - Feb 23)

and lab assignment. In the middle of this period from the project’s release (March 16, 2018) to the deadline (April 6, 2018) is a week of Spring Break. We see usage drop during Spring Break, but increase again in the following week as the Project 2 deadline approaches in Figure 3.6.

A typical lab week (the second week of lab) is reflected in Figure 3.7, and usage for a typical lab day (regression lab in Week 8) is detailed further in Figure 3.8.

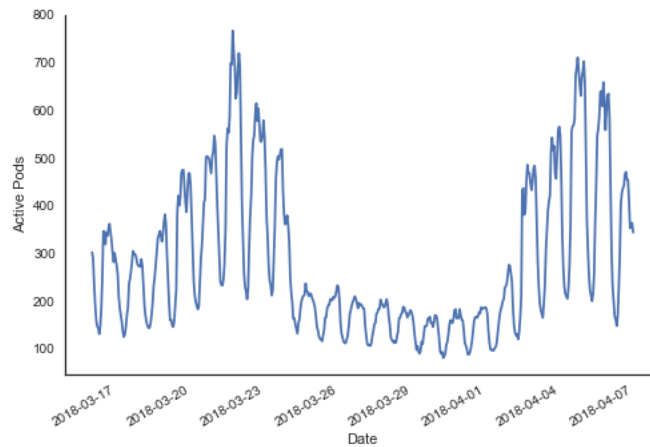


Figure 3.6: Data 8 Datahub Active Pods during Project 2 Weeks (Mar 16 - Apr 6)

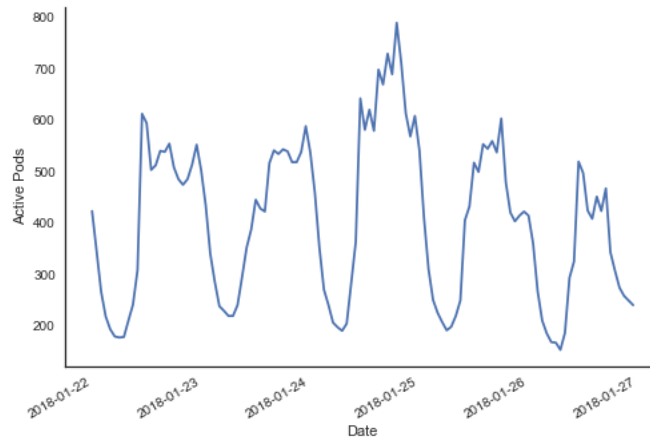


Figure 3.7: Typical Lab Week: Data 8 Datahub Active Pods during Lab 2 (Jan 22 - Jan 26)

### Data 8X’s First Few Weeks of Cluster Usage

We also present a summary of the active pod state data from the first two weeks of Data 8.1X, *Foundations of Data Science: Computational Thinking with Python*, in Figure 3.9. The alpha and beta clusters, when added together, show over 1000 simultaneous active users once the course was launched. As the first iteration of the MOOC was currently taking place at the time this report was written, analysis of usage data throughout the rest of the course and in the two follow on 5-week courses is left as future work. Based on costing estimates

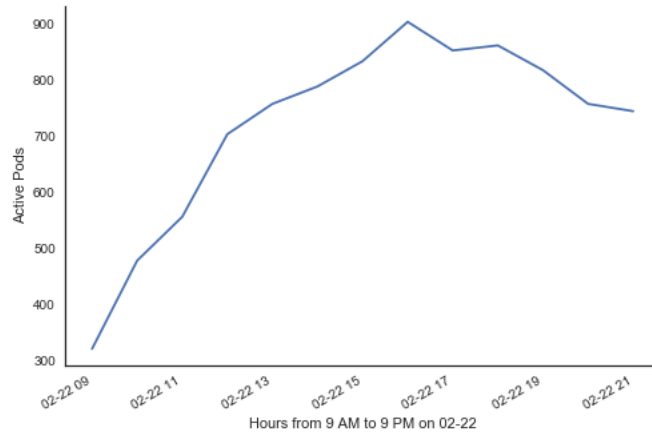


Figure 3.8: Typical Lab Day: Data 8 Datahub Active Pods on Thursday, Feb 22

detailed in Chapter 4, we can expect that the next few weeks of Data 8.1X see a similar increasing hub usage trend.

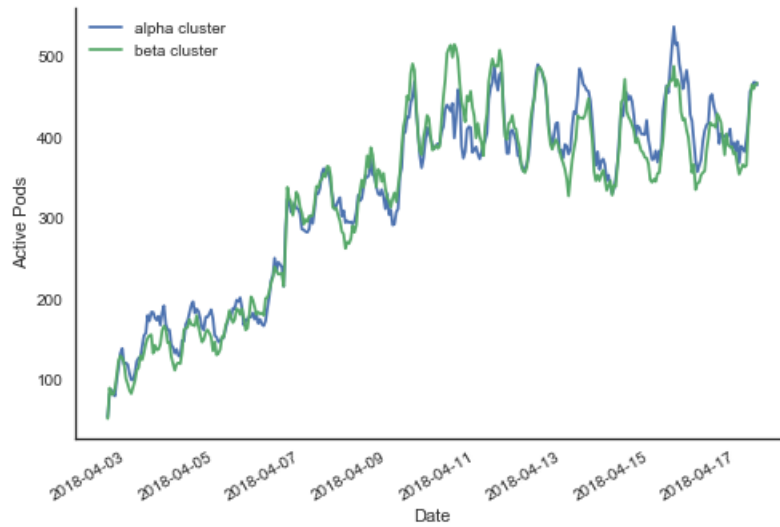


Figure 3.9: Data 8.1X Alpha and Beta Cluster Active Pods (Apr 2 - Apr 17)

## 3.4 Instructor Workflow

Instructor workflow of uploading assignments to students is simplified through a notebook extension called *nbgitpuller*, formerly called *nbinteract* [22]. This extension duplicates the contents of a specific portion of a Github repository into a user's Jupyter server [4]. The link's URL reflects the jupyterhub URL (datahub.berkeley.edu), git repository link (data-8/materials-sp18) and specific folder path (materials/sp18/hw/hw01/hw01.ipynb) to duplicate, as seen in Figure 3.10. Figure 3.11 illustrates the general frame used to create interact links with the optional arguments of selecting a git branch (default *branch* argument is master) and application (*app*) view of notebook or folder. This enables instructors to simply upload their assignments in Jupyter notebooks, CSVs, and Python files using Github, and give students a URL to duplicate all of the files from the repository so they have the exact same environment.

```
http://datahub.berkeley.edu/hub/user-redirect/git-  
pull?repo=https://github.com/data-8/materials-  
sp18&subPath=materials/sp18/hw/hw01/hw01.ipynb
```

Figure 3.10: Interact link example for Data 8's Homework 1 from nbgitpuller

```
https://myjupyterhub.org/hub/user-redirect/git-  
pull?repo=<your-repo-url>&branch=<your-branch-  
name>&subPath=<subPath>&app=<notebook | lab>
```

Figure 3.11: Interact link frame from nbgitpuller

## 3.5 Surrounding Suite of Learning Tools

Data 8 takes advantage of several other tools highlighted in this paper to maintain an effective class at over 1000 students. The Lab Assistant manager and Office Hours Queue are both built within OK, a learning tools, analytics, an autograding platform discussed further in Chapter 5.

### Discussion forums

Data 8 uses the open source forum Piazza to enable discussion between students of the class and for students to ask questions to instructors. By letting students see the questions and answers of other students at the scale of 1000 students, Piazza reduces question repetition and allows students to gain valuable learning opportunities.



Data 8X utilizes EdX’s built-in discussion forum for the same purpose. The support costs mentioned in Chapter 4’s cost analysis involve the maintenance of the forum and the technical staff operations costs of modifying and uploading course material.

## OK: Lab Assistant Manager

As mentioned in Chapter 2, Data 8 relies on two to three lab assistants in each lab section to help answer student questions. Since there are ~40 lab sections each semester, managing the corresponding ~100 lab assistants becomes a challenging task. The Lab Assistant manager was built and integrated within OK to keep track of lab assistant attendance and communication, as well as collect statistics and feedback. A lab assistant visits la.data8.org and authenticates with their OAuth Berkeley username and password to receive credit for helping out in lab (Figure 3.12). GSIs can monitor their lab assistants’ attendance directly through the same authentication scheme. This also makes it easier for course instructors to assign lab assistant grades at the end of the semester as all of the check-ins are centralized.

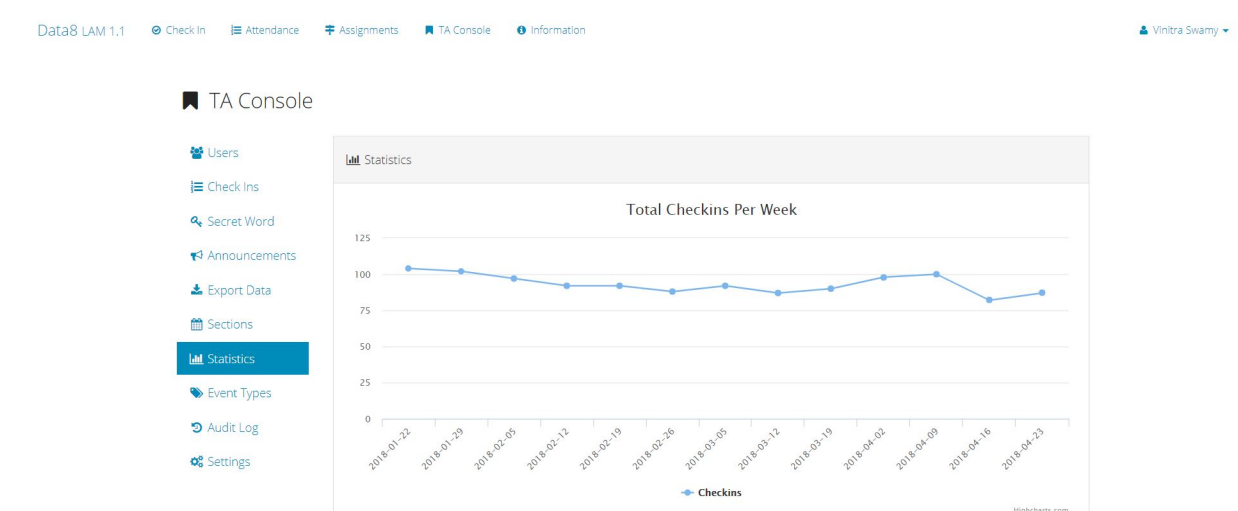


Figure 3.12: Statistics and Layout of Lab Assistant Manager

## OK: Office Hours Queue

Another useful tool integrated within OK addresses managing large quantities of students at office hours (30 students) with limited staff members (4-5 staff members). Students authenticate with their OAuth Berkeley accounts to submit a ticket requesting a staff member’s help. This ticket, reflected in Figure 3.13 and 3.14 will ask students to select a specific assignment, question number, and office hour room location. A staff member who logs into the same portal will see the screen in Figure 3.14 and answer a ticket by locating the student and helping them with their problem.

Staff can also choose to answer multiple tickets for the same question at the same time by gathering the respective students and “helping” them all at once. This allows every student to receive the help they need without having the loudest students receive the most time. We can also better anticipate demand for project weeks and see which questions are the most difficult through the statistics accumulated through the office hours queue.

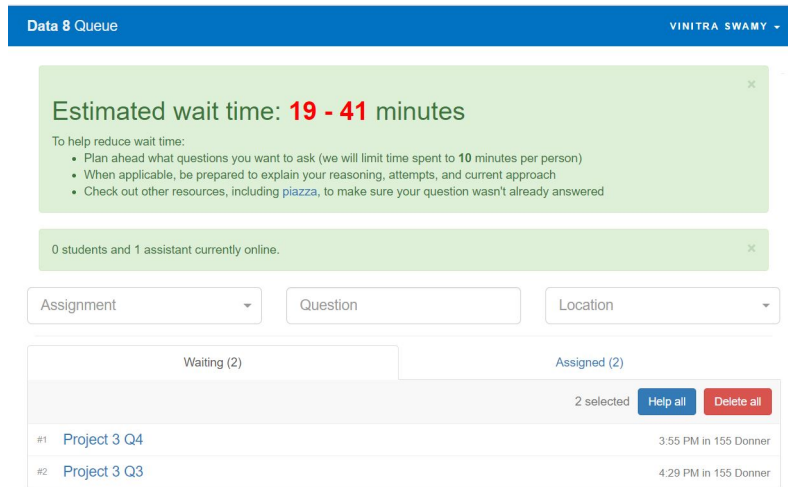


Figure 3.13: Queuing System to address students in office hours

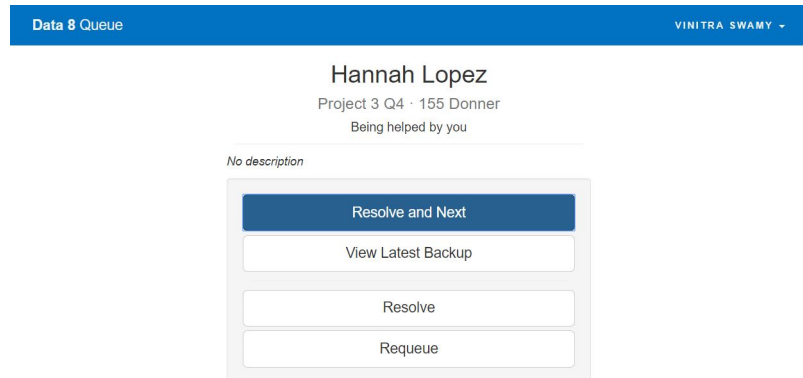


Figure 3.14: Student Ticket Layout on Office Hours Queue

## Chapter Acknowledgements

Thank you to Yuvi Panda, Ryan Lovett, Samuel Lau, Wilton Wu, Chris Holdgraf, the UC Berkeley OKPy team, Gunjan Baid, Professor John DeNero, Professor David Culler, and

the many people at the Division of Data Science and throughout Berkeley involved with implementing the architecture and learning tools described above.

# Chapter 4

## Resource Scaling

### 4.1 Capacity Planning

In scaling our computing architecture to a MOOC, the first question to consider is: “How many users are we expecting?” MOOCs in computing and data science traditionally have enrollment in the tens to hundreds of thousands [15]. With this in mind, we planned for between 20,000 to 50,000 concurrent users with a total of 100,000 to 200,000 enrolled users.

The current statistics reflect  $\sim 45,000$  students enrolled in the course with 8,000 active weekly users. In overestimating the user population, we have designed an architecture that is flexible enough to accommodate rapid growth while avoiding waste of resources in the short term.

### 4.2 Cost Estimates and Analysis

Our cost target from various funding sources allows Data 8X to budget \$2 per enrolled student per week. At the time of doing this analysis, a week before the course was set to release, we had 12,000 pre-enrolled users and a \$24,000 weekly budget. As of May 2018, we have 45,000 enrolled users. The cost analysis done in this chapter are vague estimates for hardware capacity planning purposes, not financial planning purposes. The latter would require being far more accurate than we describe here.

#### Maximum Cost

We start by evaluating a generous maximum cost estimate using the 12,000 enrolled students figure. Based on the scale of datasets used in Data 8, we estimate giving each user 1GB of RAM. Our objective is to minimize the number of nodes used. Due to a 100 pod (user) limit per node, we choose to use Google Cloud Platform’s n1-highmem-16 node type with 16 CPUs and 104GB RAM.

Table 4.1: Cost Analysis Comparison

Utilization	Number of Simultaneous Users	Number of Machines	Cost Per Student
8.75%	1000	10	\$0.11
25%	3000	30	\$0.40
50%	6000	60	\$0.83
100%	12000	120	\$1.25

Therefore, we estimate 120 machines to support our 12,000 enrolled users. This totals \$14,000 per week, with an estimated additional \$1,000 per week of support costs. Our maximum cost figure is \$1.25 per student per week.

## Expected Cost

In the maximum cost estimate, we assumed that all users would be active at once. While we can prepare for that edge case, a more realistic generous estimate is that 25% of users will be active at any given time. Even though the number of learners who enroll in the courses tends to be in the range of thousands, only a very small portion of the enrolled learners complete the course. According to the visualizations and analysis conducted by Katy Jordan in 2015, the investigated MOOCs have a typical enrollment of 25,000, even though enrollment has reached a value up to 230,000. Jordan reports that the average completion rate for such MOOCs is approximately 15% [18]. A study from Coursera in 2012 suggests an even smaller completion rate of 7% to 9% [23].

To support 25% active users at once, we require about 30 nodes. This reduces our operating cost to about \$4,000 per week. With the \$1,000 in additional support cost, we have a new expected cost estimate of \$0.40 cents per user per week. See the comparison between the maximum cost estimates and expected cost estimates in the table below. We can now confirm our estimate of 25% active users accurately reflects engagement in the current statistics of 45,000 enrolled students with 8,000 completing the weekly assignments.

## Fixed Costs: Minimum State

After estimating a maximum and expected cost, we conduct an analysis of our minimum costs to determine the number of Kubernetes clusters we need. We choose to limit each hub to 1,000 total users and each cluster to 10,000 total users as a cautionary measure. Assuming a maximum of 40,000 users, we initially considered splitting the load across 4 clusters of 10 hubs each. However, with the realization that we can easily add hubs and clusters later on, we decided to split across 2 clusters of 10 hubs each for a 20,000 simultaneous user capacity.

We also calculate the number of Network File System (NFS) hosts required, recognizing that we can resize instances without too much downtime and add new instances if needed [24]. We choose to use two nodes of n1-highcpu-8 of 1 TB each, with the additional CPU capabilities due to the high file and network I/O for this task. Due to connection limits, we



Figure 4.1: Data 8X Daily Spending for February to May

decide to use a different Postgres SQL database instance for each of the following purposes: NFS Sharding, Hub Sharding, and the Hub Database.

### Cost saving strategies, Autoscaling

The most effective way to save costs is to size the architecture correctly [25]. Capacity planning and calculating the estimates in the section above go a long way towards not wasting resources. Another important cost saving strategy is utilizing the Kubernetes autoscaler. We set autoscaling limits close to 10% utilization for lower bounds and 100% utilization as an upper bound. As 100% utilization still falls well under budget, we did not prioritize autoscaling during this initial MOOC iteration. Implementing a more effective autoscaling procedure remains in future work.

### Comparison of Estimates with Actual Cloud Scaling Costs

In the month of April 2018, there were  $(6720 + 9176 + 6443 + 5543)/4 = 6970.5$  learners in Data 8X each week that played at least one video. Overall compute costs average to \$2.8K per week in April. This results in the estimate of \$0.40 per active video learner per week, which justifies our earlier calculations. The daily spending expenditure is reflected in Figure 4.1.

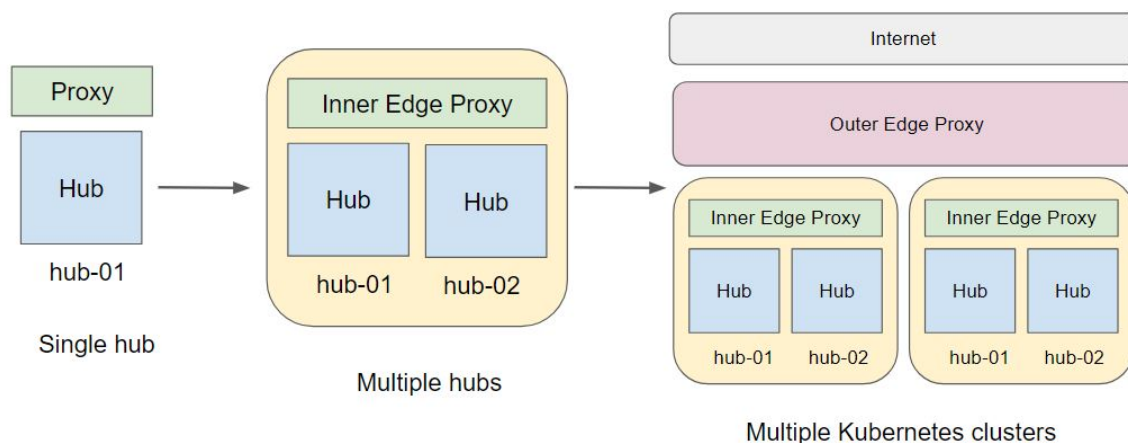


Figure 4.2: Evolution of performance testing architecture

### 4.3 Performance testing

In testing the performance of scaling JupyterHub instances, we first identify the limits of a single hub, design a sharding procedure allocating users to multiple hubs, and expand on that strategy to shard users across multiple clusters. This evolution is reflected in Figure 4.2. We conclude this section by discussing the merits of sharding over load balancing strategies. At the time of developing this architecture, we tested all of our procedures with JupyterHub 0.8. Future work lies in upgrading to JupyterHub 0.9, which has massive performance improvements.

#### Identifying limits of a single hub

Our original explorations begin with identifying the limits of a single hub, as is used in the architecture of the in-person Data 8 course at UC Berkeley. The components of the simplest scalable JupyterHub instance consist of a proxy layer and a hub, as shown in the first section of Figure 4.2. The limits of this hub are hit at about 5,000 users, which is far below our target of 100,000 users. Additionally, the architecture of a single hub and a single NFS (network file system) disk causes us to have a system with a single point of failure. We want to expand upon that in our next design.

#### Sharding users across multiple hubs

Our next scaling strategy attempts to allocate users across multiple hubs instead of a single hub. The second section of Figure 4.2 highlights the components of multiple hubs and an

“inner edge” proxy layer. While this more effective than a single hub, we still cannot reach the scale that we need while effectively monitoring each hub.

## Sharding hubs across multiple clusters

We move on to our current architecture, which involves using Kubernetes clusters to manage multiple instances of Jupyterhub. Kubernetes is an open-source container-orchestration system for automated deployment [3]. In this case, we use Kubernetes to manage containers created in Docker for each student running an instance of Jupyter. The components highlighted in the third portion of Figure 4.2 include the “outer edge” proxy, Kubernetes clusters containing hubs, and each cluster’s respective “inner edge” proxies. In this architecture, we shard user home directories across multiple NFS servers.

## Sharding vs Load Balancing

In this chapter, we detail system architectures that use sharding approaches as opposed to load balancing. When employing a sharding strategy, a user is assigned to a hub the first time they log in. Every time that specific user logs in afterwards, they are assigned to the same hub. A load balancing strategy assigns a user to the least loaded hub every time they log in. It is clear that load balancing is a more effective strategy, and is left to be implemented in future work. Sharding was used for our initial architecture because it is easier to implement and avoids dealing with the problem of finding a live count of users, which requires expensive operations, and avoids crowding due to correlated starts. Since we are far under budget and are not concerned directly with cost saving, we chose to prioritize sharding – however, in other courses and in future iterations of Data 8X, load balancing is a priority.

## 4.4 Maintenance and monitoring

In designing a new architecture, we prioritize observability in our monitoring scheme.

1. We want observability at each layer of the architecture. It is important to see the status at each NFS server, as well as the errors at the inner and outer edge level.
2. We want to reduce the possibility of bottlenecks or single points of failure. This is why our architecture prioritizes multiple clusters of users, and requests statistics/errors/active connections separately at each edge.
3. We do not want the user’s interactions to deal with the inner architecture directly. The user only interacts with the outer edge layer of this architecture. Upon receiving a HTTP request, the outer edge proxy layer routes the user to the correct cluster and the inner edge routes the request to the correct hub.





Figure 4.3: Data 8X Grafana dashboard for Alpha and Beta clusters

## Logging, Monitoring, and Metrics aggregation

Metrics are monitored using Prometheus and the real-time Grafana dashboard shown in Figure 4.3. Prometheus is an open-source monitoring system and time series database used to monitor system architectures at scale [26]. Grafana is a platform to visualize analytics for time series data [27]. We use Prometheus to extract metrics from each layer of the architecture and Grafana to display queries regarding memory usage, requests at HTTP layers, CPU usage, disk usage, users currently running, and RAM over time. The most important metric we watch for is the error rate of running pods as this reflects the status of our architecture as well as the user experience.

# Chapter 5

## Grading and Evaluation

Countless learning theory studies have demonstrated the efficacy of evaluation for student understanding [28, 29]. The design of Data 8 embodies constructivist philosophies in assignment design, a teaching approach in which assessment is a key factor [30]. When expanding from a manageable class size to the thousands of students in a MOOC, evaluation in terms of autograding capabilities become increasingly important. In this chapter, we discuss the autograding and evaluation schemes of Data 8 and Data 8X.

### 5.1 OK: Immediate feedback on tests

In Data 8, we use the OK autograding platform within python cells of Jupyter notebooks to evaluate student feedback immediately. OK autogrades programming assignments and facilitates submission, composition feedback, and analytics for a course [31]. OK was initially developed for use in CS 61A, UC Berkeley’s introductory computer science course. It has since expanded to include integration in Data 8 Jupyter notebooks and a variety of other courses using its open-source APIs.

For each question presented on a Jupyter Notebook assignment, the instructor writes a companion series of python autograder tests to assess accuracy. As there are infinite ways students can arrive at a correct answer in most programming problems, OK tests the content of the variables in the student’s environment instead of the exact code the student writes. Using the `ok.score()` command, the total score is provided to students as soon as they run the cell as shown in Figure 5.2. The novelty in this approach is that the instructor can create complex visible and hidden tests in the industry-standard Jupyter notebook environment. Some MOOCs use existing browser based editors like DataCamp or ProcessingX that do not allow students to modify their programming environment [32, 33], or a combination of peer review and multiple choice questions that limit the scope of an instructor’s creative freedom.

In the long term, a graphical user interface and user-friendly autograding output in the context of the notebook environment could help ease students into debugging failed tests. The merits of autograding interactivity within a Jupyter Notebook and the implementation

```

jupyter project3_master Last Checkpoint: in a few seconds (autosaved)
File Edit View Insert Cell Kernel Widgets Help Trusted Python 3 O Mem: 179 / 1024 (MB)

Question 1.1.3
Assign percent_unchanged to the percentage of words in vocab_table that are the same as their stemmed form (such as "blame" above).

In [40]: num_unchanged = sum(vocab_table.column(0) == vocab_table.column(1))
percent_unchanged = num_unchanged / vocab_table.num_rows * 100
print(round(percent_unchanged, 2), '% are unchanged')

30.1 % are unchanged

In [41]: _ = ok.grade("q1_1_3")
_ = ok.backup()

~~~~~
Running tests
-----
Test summary
  Passed: 2
  Failed: 0
[ooooooooook] 100.0% passed

```

Figure 5.1: OkPy Test in the Jupyter Notebook Environment for Data 8 Project 3

of an inline graphical autograder is left as future work.

OK's autograding capabilities are part of a larger ecosystem including UI authentication, management of lab assistants, analytics for course assignments, an office hour management queue, and several others. We take full advantage of this ecosystem in Data 8 to improve the student experience. However, Data 8X's online environment has a different set of needs that are not served by OK's surrounding tools, so implementing OK in full adds unnecessary complexity.

## OK and Gradescope

The data science concepts of hypothesis testing, confidence intervals, and inference tested throughout Data 8 necessitates written response questions as well as programming questions. Therefore, while the majority of questions are autograded using instructor-written hidden tests in OK, the written-response answers are scraped and uploaded to Gradescope for Data 8 tutors to evaluate. Gradescope streamlines the tedious parts of grading through AI-assisted answer grouping, parallelized rubric creation, and parallelized evaluation by question [34]. Once the tutors have graded the assignment, the grades are persisted back to OK.

## 5.2 LTI Standard: posting grades to EdX

We reintroduce the Learning Tools Interoperability (LTI) standard, created by the IMS Global Learning Consortium to link content and resources to learning platforms. The LTI standard, mentioned briefly in Chapter 3, provides a secure solution to post grades back to the EdX platform without forcing users to re-authenticate or require active user interaction.

The alternative OAuth standard used for posting grades in Data 8 only saves a user's identity for a short period of time. However, with the LTI standard, it is possible to post grades even a week after the student submits their code.

## **Autograding in EdX**

Using the LTI standard for authentication enables users to have a one-click seamless integration of autograding in EdX. Integration was not possible directly with OK, so we use a script written by Operations Architect Yuvi Panda to circumvent this problem. Data 8X users are not asked to reauthenticate when they try to access a Jupyter assignment or when enabling autograding since they are already authenticated with LTI. However, with the OAuth standard used in Data 8, students are asked to authenticate for the first time after clicking a datahub link. Had OAuth been implemented for Data 8X, students would have to log in twice. Another advantage of using LTI in Data 8X context is that grades can be persisted directly through the notebook instead of being uploaded to OK servers and transferring to grading portals later on.

## Chapter 6

# Analytics and Prediction in Jupyter-based computing environments: Deep Knowledge Tracing

The motivation for this chapter is to provide an analytics extension in the context of Data 8's iterative educational coding environment at scale. This portion of my thesis is a collaborative work within the UC Berkeley EECS department and Division of Data Sciences. I would like to thank Allen Guo, Samuel Lau, Madeline Wu, Wilton Wu, Professor Zachary Pardos, and Dean David E. Culler for their contributions to this chapter. The final version of this work is published as a short paper in Artificial Intelligence in Education (AIED) 2018's Springer proceedings.

Knowledge Tracing, and its recent deep learning variants, have made substantial progress in modeling student knowledge acquisition through interactions with coursework. In this chapter, we present a modification to Deep Knowledge Tracing to model student progress on coding assignments in large-scale computer science courses. The model takes advantage of the computer science education context by encoding students' iterative attempts on the same problem and allowing free-form code input. We implement a workflow for collecting data from Jupyter Notebooks and suggest future research possibilities for real-time intervention.

### 6.1 Knowledge Tracing Approaches

With students far outnumbering teachers in online learning platforms, there is increased demand for tools to maintain and improve learning. Since 1-on-1 instructor support is not feasible at this scale, educational technology seeks to use artificial intelligence to provide similar guidance and model students' knowledge.

A popular approach called Knowledge Tracing models students' knowledge as they cor-

rectly or incorrectly answer exercises. Accurate modeling allows students to spend more time working on questions that are suited for their level of understanding.

Recent work uses recurrent neural networks to more effectively encode representations in an approach called Deep Knowledge Tracing (DKT). DKT models use the accuracy of prior attempts to predict students' future performance on questions, as well as automatically learn question clusterings [35].

As instructors for large scale computer science courses at UC Berkeley, we are particularly interested in modifying DKT for the computing education context. Traditional knowledge tracing techniques succeed in intelligent tutoring contexts, where students attempt each exercise once. In computing education, however, students often attempt a single exercise multiple times until it is solved. We modify DKT to take this into account and make the following contributions:

1. We modify the model to take vectorized free-form student code as input.
2. We map questions to multiple skills and train one model per skill.
3. We implement a workflow for collecting data from Jupyter Notebooks and OkPy autograding.
4. We suggest possible applications of this model for determining a point of intervention in real-time.

## 6.2 Related Work

Knowledge Tracing traditionally uses Hidden Markov Models to track student knowledge as they solve exercises [36]. Deep Knowledge Tracing (DKT), introduced by Piech et al. [37], is an approach to knowledge tracing that utilizes recurrent neural networks, specifically Long-Short Term Memory (LSTM) cells, to produce improvements in model capabilities over previous methods. Piech et al. [35] also use DKT to automatically cluster math exercises into skill groups.

Blikstein and Piech [38, 39] show that a student's trajectory of attempts while solving programming exercises is predictive of their success. Wang et al. [40, 41] expand on this work by using DKT to model student trajectories as they solve programming exercises in the block-based programming language Scratch. We replicate the Wang et al. work in our setting, making a modification of input to the model in order to formulate the problem in terms of free-form code.

## 6.3 Data Context

We examine data from student code submissions from UC Berkeley's introductory data science course, Data 8.

Students complete assignments using Jupyter notebooks, a cell-based Python execution environment, with problem descriptions and starter code [11]. Instructors manually create assignment skeletons and autograder tests for each assignment. Every time the autograder is run, students' code and accompanying notebook metadata are backed up to the OkPy server [42]. This creates time series data recording a student's progression through a given assignment. The Jupyter environment enables rapid, iterative learning—students can write, run, and check their programs' correctness in near real-time.

The OkPy API allows us to retrieve one student backup for each autograder run, final submissions, and OkPy-specific assignment metadata. The raw submission data contains question numbers, student code responses, and accompanying autograder test results.

All student code is written in Python 3. Students use both *NumPy* and UC Berkeley's in-house *datascience* package, a tabular data manipulation library that draws inspiration from the *pandas* library.

## 6.4 Model Methodology

We make a number of data modifications to make raw student code amenable for DKT techniques. We also modify the baseline DKT approach to model student knowledge using code submissions from failed attempts.

### Data Featurization and Sanity Check

To featurize code submissions, we use the default tokenization scheme in *scikit-learn* [43], which splits on non-alphanumeric characters. For example, consider the following code submission:

```
murder_rates.join('State',death_penalty,'State')
    .pivot('Death Penalty','Year','Murder Rate', np.average)
    .select(0,2,1)
```

The first 10 extracted tokens are:

```
['murder_rates','join','state','death_penalty',
 'state','pivot','death','penalty','year','murder']
```

For each distinct word, we compute the term frequency-inverse document frequency (tf-idf) score across all submissions. Each code submission is represented as a vector of length  $K$ , where  $K$  is the vocabulary size: the total number of distinct words across all code submissions.

## Model Architecture

We train a LSTM network on sequences of student code submissions. Instead of a single network as in DKT, we train a separate neural network for each skill (as shown in Figure 6.1) using the vectorized code representation described in the previous section.

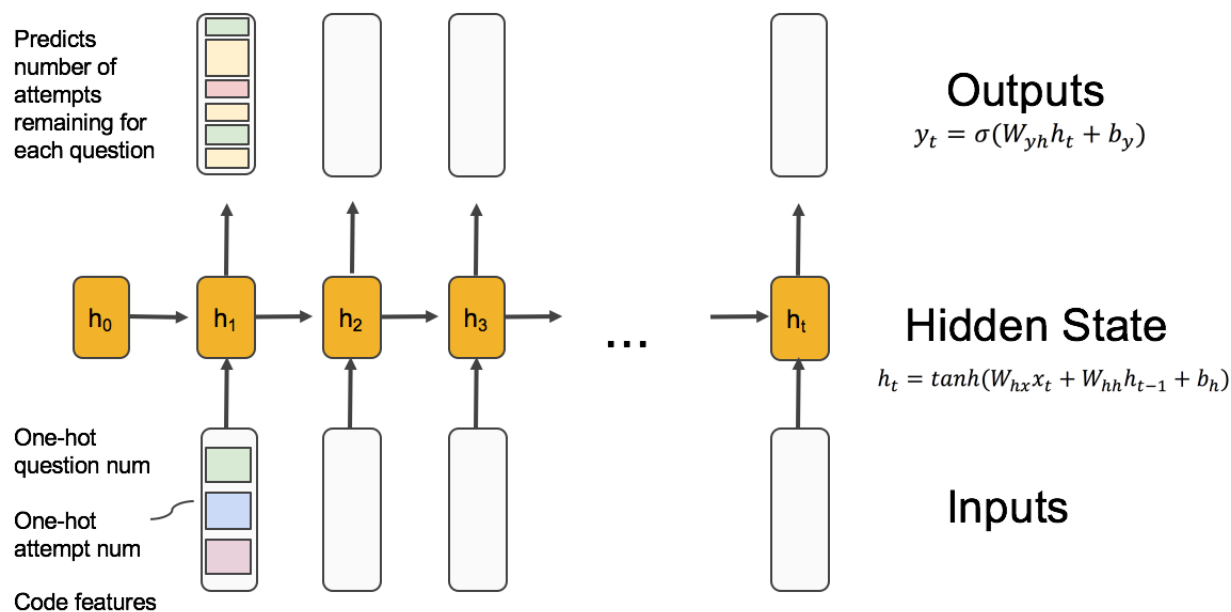


Figure 6.1: Architecture of Our DKT-Inspired LSTM Network

The inputs to the LSTM network are a vector containing the one-hot encoded anonymized student identifier, the one-hot encoded question number, the one-hot encoded attempt number, and the vectorized representation of the code. The output of the model at each time-step is a vector of predictions for the number of attempts remaining for each question of the same skill.

Each of these arrays have taken the students' progression (question each student attempted at each attempt number) and one-hot-encoded the inputs to create these arrays. We deal with the problem of each student having a different number of attempts on a skill by padding our data to the max number of attempts a student has for a skill, but we can also truncate data to an arbitrary "pad number" as well (e.g. 100 attempts).

## 6.5 Expansions and Future Areas of Exploration

Our modifications to DKT motivate several lines of future research in intervention and hint generation. For each student, our model predicts the number of attempts remaining to complete the questions for each skill. This suggests a simple way to provide live feedback by



subject or question for instructors in a lab setting. For example, an instructor can identify a potentially useful time to intervene when a student's total predicted number of attempts remaining is consistently higher than those of other students (i.e., falling in the top  $k$ -th percentile for some manually tuned  $k$ ).

The learned representation of student knowledge suggests a method to automatically provide hints to students. By simulating modifications to student code and checking the predicted attempts remaining, we can use the trained networks to find the keywords that most quickly help the student complete the exercise. Implementation of this hint generation model in a MOOC (perhaps in future iterations of Data 8X) can help learners adjust to the lack of in-person support of tutors, lab assistants, or instructor office hours in an online setting.

## 6.6 Note on Results

We have run preliminary models on mock data and have found that DKT with free-form code reduces error significantly in comparison to baseline DKT. Future work will involve writing up the analysis and insights gained from student code attempts as well as the specifics of our trial results, pending IRB approval.

## 6.7 Chapter Acknowledgements

Thank you to the UC Berkeley Machine Learning in Education course, the Jupyter team, and the OkPy team for their support throughout this process.

# Chapter 7

## Future Work: where to next?

### 7.1 Future Work: Autograding

In Chapter 5's autograding section, we explain how each day our Operations Architect runs a script to post final grades from each submission back to EdX. This approach is clearly untenable in the long run. The goal is that users should be able to run a button and get their final score immediately posted to EdX. EdX will only keep the maximum score it receives, and handing off the autograding computation to the user's discretion allows us to balance the load more evenly.

### 7.2 Future Work: Sharing the Knowledge

A common cautionary note throughout designing large systems emphasizes that there should not be a single point of failure. We attempt to keep this in mind in our system design, but it is important to keep that in mind in terms of support structure as well. A lot of this information is currently silo-ed in the minds of our course developers. With enough training and documentation, we will be able to make sure there are multiple people who can come to the rescue in case something goes wrong.

### 7.3 Future Work: Deep Knowledge Tracing Models for Code Progression

In the predictive model proposed in Chapter 6, there is future work to be done in terms of classroom implementation, official trials, and analysis across semesters. As discussed at the end of the chapter, there is also potential for recognition of students' conceptual misunderstandings and intervention through an instructor or generated hints to rectify the students' trajectories.

# Chapter 8

## Conclusions

### 8.1 First Iteration of the MOOC

As the team moves forward to Data 8.2X and Data 8.3X, as well as future iterations of the MOOC, we have a lot to learn and analyze. The current takeaways reflect that the architecture was executed correctly, and everything is going according to plan. More importantly, the intuition that JupyterHub and the surrounding Berkeley Data Science Education Stack can scale in this situation was correct.

### 8.2 Data 8 around the world

Many universities around the world are implementing their own versions of Data 8. The novel curriculum and unique case-study focused approach to data science education has seen great success. This report will hopefully become a resource for instructors interested in starting their own versions of Data 8, or adapting computing courses of their own into MOOCs.

### 8.3 Useful Resources / Links

1. Zero To JupyterHub Guide – Contributed to by members of the UC Berkeley Jupyter team, this guide fully walks users through setting up their own JupyterHub instance.
2. Division of Data Sciences – The UC Berkeley Division of Data Sciences offers a wide variety of data-related courses, events, and research opportunities.
3. Datascience Package Documentation – The docs for the datascience package, developed in-house at UC Berkeley by John DeNero, David Culler, Alvin Wan, and Samuel Lau, are helpful in understanding the programming used in Data 8 and Data 8X.

# Bibliography

- [1] *Data 8: The Foundations of Data Science*. <http://data8.org/>.
- [2] *Project Jupyter*. URL: <http://jupyter.org/>.
- [3] *Production-Grade Container Orchestration*. URL: <https://kubernetes.io/>.
- [4] *Build software better, together*. URL: <https://github.com/>.
- [5] Seymour Papert. *Mindstorms: Children, computers, and powerful ideas*. Basic Books, Inc., 1980.
- [6] Lori Breslow et al. “Studying learning in the worldwide classroom: Research into edX’s first MOOC”. In: *Research & Practice in Assessment* 8 (2013).
- [7] Heather Miller et al. “Functional Programming for All! Scaling a MOOC for Students and Professionals Alike”. In: *Companion Proceedings of the 36th International Conference on Software Engineering*. ICSE Companion 2014. Hyderabad, India: ACM, 2014, pp. 256–263. ISBN: 978-1-4503-2768-8. DOI: 10.1145/2591062.2591161. URL: <http://doi.acm.org/10.1145/2591062.2591161>.
- [8] Natalia Spyropoulou et al. “Developing a computer programming MOOC”. In: *Procedia Computer Science* 65 (2015), pp. 182–191.
- [9] *Docker - Build, Ship, and Run Any App, Anywhere*. <https://www.docker.com/>. (Accessed on 05/14/2018).
- [10] L. Fernandez et al. “Jupyterhub at the ESS. An Interactive Python Computing Environment for Scientists and Engineers”. In: *Proc. of International Particle Accelerator Conference (IPAC’16), Busan, Korea, May 8-13, 2016*. (Busan, Korea). International Particle Accelerator Conference 7. doi:10.18429/JACoW-IPAC2016-WEPOR049. Geneva, Switzerland: JACoW, 2016, pp. 2778–2780. ISBN: 978-3-95450-147-2. DOI: doi:10.18429/JACoW-IPAC2016-WEPOR049. URL: <http://jacow.org/ipac2016/papers/wepor049.pdf>.
- [11] Thomas Kluyver et al. “Jupyter Notebooks—a publishing format for reproducible computational workflows.” In: *ELPUB*. 2016, pp. 87–90.
- [12] Helen Shen. “Interactive notebooks: Sharing the code”. In: *Nature News* 515.7525 (2014), p. 151.
- [13] *DS100*. <http://www.ds100.org/>.

- [14] *Prob 140 - Probability for Data Science*. <http://prob140.org/>.
- [15] *A Product at Every Price: A Review of MOOC Stats and Trends in 2017*. <https://www.class-central.com/report/moocs-stats-and-trends-2017/>.
- [16] *The Professors Behind the MOOC Hype - The Chronicle of Higher Education*. <https://www.chronicle.com/article/The-Professors-Behind-the-MOOC/137905#id=overview>.
- [17] *Big (MOOC) Data StratEDgy*. <http://www.insidehighered.com/blogs/stratedgy/big-mooc-data>.
- [18] *MOOC completion rates*. <http://www.katyjordan.com/MOOCproject.html>.
- [19] *MoocGuide - 2. Benefits and challenges of a MOOC*. <http://moocguide.wikispaces.com/2.+Benefits+and+challenges+of+a+MOOC>.
- [20] *Foundations of Data Science edX*. <https://www.edx.org/professional-certificate/berkeleyx-foundations-of-data-science#courses>.
- [21] *GapMinder*. <https://www.gapminder.org/data/>.
- [22] *data-8/nbgitpuller*. <https://github.com/data-8/nbgitpuller>. (Accessed on 05/14/2018).
- [23] *MOOCs on the Move: How Coursera Is Disrupting the Traditional Classroom - Knowledge@Wharton*. <http://knowledge.wharton.upenn.edu/article/moocs-on-the-move-how-coursera-is-disrupting-the-traditional-classroom/>.
- [24] *Network File System (NFS) - Red Hat Customer Portal*. [https://access.redhat.com/documentation/en-us/red\\_hat\\_enterprise\\_linux/6/html/storage\\_administration\\_guide/ch-nfs](https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/6/html/storage_administration_guide/ch-nfs).
- [25] *Right Sizing: Provisioning Instances to Match Workloads - AWS Whitepaper*. <https://docs.aws.amazon.com/aws-technical-content/latest/cost-optimization-right-sizing/cost-optimization-right-sizing.pdf>.
- [26] *Prometheus - Monitoring system & time series database*. <https://prometheus.io/>.
- [27] *Grafana - The open platform for analytics and monitoring*. <https://grafana.com/>.
- [28] Mamta Agarwal. "Curricular reform in schools: the importance of evaluation". In: *Journal of Curriculum Studies* 36.3 (2004), pp. 361–379. DOI: 10.1080/0022027032000152987. eprint: <https://doi.org/10.1080/0022027032000152987>. URL: <https://doi.org/10.1080/0022027032000152987>.
- [29] Charalambos Vrasidas. "Constructivism versus objectivism: Implications for interaction, course design, and evaluation in distance education". In: *International journal of educational telecommunications* 6.4 (2000), pp. 339–362.
- [30] Chris Rust, Berry O'Donovan, and Margaret Price. "A social constructivist assessment process model: how the research literature shows us this could be best practice". In: *Assessment & Evaluation in Higher Education* 30.3 (2005), pp. 231–240.

- [31] Sumukh Sridhara et al. “Fuzz testing projects in massive courses”. In: *Proceedings of the Third (2016) ACM Conference on Learning@ Scale*. ACM. 2016, pp. 361–367.
- [32] *HarvardX - Free Courses from Harvard University — edX*. <https://www.edx.org/school/harvardx>.
- [33] *Programming for Data Science*. [https://www.edx.org/course/programming-data-science-adelaidex-programx?utm\\_campaign=adelaidex&utm\\_medium=partner-marketing&utm\\_source=direct&utm\\_content=direct-programx-winter2017](https://www.edx.org/course/programming-data-science-adelaidex-programx?utm_campaign=adelaidex&utm_medium=partner-marketing&utm_source=direct&utm_content=direct-programx-winter2017).
- [34] Arjun Singh et al. “Gradescope: a Fast, Flexible, and Fair System for Scalable Assessment of Handwritten Work”. In: *Proceedings of the Fourth (2017) ACM Conference on Learning@ Scale*. ACM. 2017, pp. 81–88.
- [35] Chris Piech et al. “Deep knowledge tracing”. In: *Advances in Neural Information Processing Systems*. 2015, pp. 505–513.
- [36] Albert Corbett. “Cognitive computer tutors: Solving the two-sigma problem”. In: *User Modeling 2001* (2001), pp. 137–147.
- [37] Felix A Gers, Jürgen Schmidhuber, and Fred Cummins. “Learning to forget: Continual prediction with LSTM”. In: (1999).
- [38] Paulo Blikstein. “Using learning analytics to assess students’ behavior in open-ended programming tasks”. In: *Proceedings of the 1st international conference on learning analytics and knowledge*. ACM. 2011, pp. 110–116.
- [39] Chris Piech et al. “Autonomously Generating Hints by Inferring Problem Solving Policies”. In: *Proceedings of the Second (2015) ACM Conference on Learning @ Scale*. [email protected] ’15. Vancouver, BC, Canada: ACM, 2015, pp. 195–204. ISBN: 978-1-4503-3411-2. DOI: 10.1145/2724660.2724668. URL: <http://doi.acm.org/10.1145/2724660.2724668>.
- [40] Lisa Wang et al. “Deep Knowledge Tracing On Programming Exercises”. In: *Proceedings of the Fourth (2017) ACM Conference on Learning @ Scale*. L@S ’17. Cambridge, Massachusetts, USA: ACM, 2017, pp. 201–204. ISBN: 978-1-4503-4450-0. DOI: 10.1145/3051457.3053985. URL: <http://doi.acm.org/10.1145/3051457.3053985>.
- [41] Lisa Wang et al. “Learning to represent student knowledge on programming exercises using deep learning”. In: *Proceedings of the 10th International Conference on Educational Data Mining; Wuhan, China*. 2017, pp. 324–329.
- [42] John DeNero et al. “Beyond Autograding: Advances in Student Feedback Platforms”. In: *Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education*. ACM. 2017, pp. 651–652.
- [43] F. Pedregosa et al. “Scikit-learn: Machine Learning in Python”. In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.