Predicting Factors that Affect Student Performance in MOOC and On-Campus Computer Science Education

by

Vincent C. Vostatek

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Master of Engineering in Electrical Engineering and Computer Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

May 2020

Author .....
Department of Electrical Engineering and Computer Science
May 12, 2020

Certified by .....
Ana Bell
EECS Principal Lecturer
Thesis Supervisor

Accepted by .....
Katrina LaCurts
Chair, Master of Engineering Thesis Committee

Predicting Factors that Affect Student Performance in MOOC and On-Campus Computer Science Education

by

Vincent C. Vostatek

Submitted to the Department of Electrical Engineering and Computer Science
on May 12, 2020, in partial fulfillment of the
requirements for the degree of
Master of Engineering in Electrical Engineering and Computer Science

Abstract

We are analyzing the performance of students in both Massive Open Online Courses (MOOCs) and in residential courses. To do so, we are analyzing data gathered on two edX programming courses, as well as the residential counterparts for those courses that are administered in person at MIT. A large part of the research performed involved building out a data analysis educational software platform on which students taking the residential version of the course at MIT perform all of their work. Using this software, we gather data on students analogous to the data gathered on the MOOC students. Using this data, we will search for behaviors that lead to better performance in the courses. In addition, we use machine learning algorithms in order to be able to create predictive models that can determine how a student will perform in a course given their behaviors and background information. Finally, a major contribution of this paper is applying those machine learning models to the course software in order to provide the students with counseling as to which behaviors leads to increased performance and learning for students of their backgrounds.

Thesis Supervisor: Ana Bell
Title: EECS Principal Lecturer

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background

The modern world's most precise indicator of socioeconomic status is no longer ancestry or birth rite. The most valuable commodity of today's age is, in fact, not a material commodity at all. Rather, we are living in the age of intellect, where our hierarchies are built upon competence more so than ever before. It is shown that investing money in one's education provides a return on investment at nearly 3 times the return on investment as does investing in the stock market. People with at least a Bachelor's degree can expect to earn approximately double the income as people with just a highschool diploma. In addition, more and more jobs are requiring some form of higher education, forcing people to earn advanced degrees if they want to keep up with the ever advancing economy. This requirement for higher education and possessing advanced skills is only going to continue to increase as automation replaces more low skill jobs. The benefit of accruing intellectual capital is matched only by the necessity.

The advent of the internet provides a very interesting opportunity to overcome educational obstacles. Over the course of the past few decades, the average person has been turning to the internet for research and education over the libraries and, in some cases, over formal education. One new idea that has become popularized is Massive Open Online Courses (MOOCs). MOOCs are oftentimes complete college

courses that are made available to anyone online for either no charge at all, or a small charge in order to obtain an official certification.

1.2 Overview

With access to data collected by the tow MOOC runs, as well as data collected through a course website that we have built, we have conducted research into students' behavior in order to be able to make predictions on student performance given their respective patterns of behavior, and to discover what leads to increased learning and better performance in a programming course given the background of the student.

We will be analyzing two hypotheses. The first hypothesis is that having in person teaching provides benefits to the students and their ability to learn. This is in contrast with taking purely online courses – even those with online forums to ask and answer questions. In large part, we have taken a survey of contemporary literature on this topic [5] [3] [7] and have summarized the results. Then, we examined the data from the residential course that we have access to, and compared the results.

The second hypothesis that we will be looking into is that the performance of each profile of student can be predicted based on measurable actions taken by that student over the course of the semester. By looking at both the profile of the student and the behaviors that each student exhibits while they take the course, we have been able to pinpoint certain behaviors that benefit students with specific backgrounds. This enables us to guide them towards a set of behaviors that are shown to increase the likelihood of their success. This can be done using the course website that we have built for this research. On the website, students will be able to see their predicted final grade given their profile and performance thus far, and any tailored suggestions that the model finds that will most increase their predicted performance.

A few definitions are in order. Firstly, the residential classes that we are observing will be 6.0001 (Introduction to Computer Science and Programming in Python) and 6.0002 (Introduction to Computational Thinking and Data Science). A residential class is a class that is administered in person - in this case at MIT. There is a version

of both of these classes that is offered as a MOOC for the general public. The names of these courses are 6.00.1x and 6.00.2x respectively. Secondly, when referring to a student's profile, we are referring to data gathered through surveys that shed light on their knowledge of course material upon entering the class, as well as data gathered on their behavior during the course of the term. Finally, success is measured based upon the final grade earned in the class. This grade is represented as a percentage, and is determined by the instructors of the courses.

In order to shed light on these hypotheses, we have examined data from several sources. Firstly, we have analyzed data taken from the 2019 Spring runs of 6.00.1x and 6.00.2x. Secondly, we collected data on the approximately 400 students that enroll in 6.0001 and 6.0002 at MIT each semester. These two data sets provided us with a very important differentiation that we used to compare one method of learning to another. The edX data provided us with data on students that do not have access to in person assistance and simply complete all assignments online and submit them online for grading. Contrarily, the MIT student data provided us with data from students who also have access to face to face lectures, recitations, and office hours.

1.3 Related Work

1.3.1 MOOC Research

As mentioned before, there has been extensive research performed on student performance in MOOCs, and what leads to better learning in online classes. This is a particular field of interest because of the high potential value in online free education. When looking at previous runs of 6.00.1x, findings from [7] suggest that student performance is determined more so by the amount of exercises that they perform rather than videos watched. They, as well as [3], suggest that students learn how to program better through actual practice than any other methods. This is termed the doer effect. This one of the various findings that we have attempted to replicate in the residential version of the data.

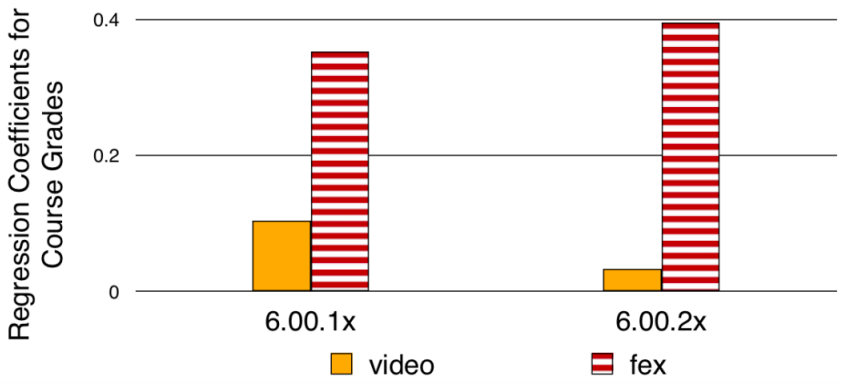

Figure 1-1: Figure from [7] on the doer effect. A higher regression coefficient indicates that the feature is a better indicator of how a student will perform in the class overall. This graphic illustrates that practical exercises more strongly related than watching videos to performance

In addition, [10] performed research into the effect that grading has had on student performance, and has found that grades have a significant impact on a student's learning in a course. We mirror this research in our residential data by examining the large number of freshmen that take 6.0001 and 6.0002 at MIT in the fall in comparison to the spring. Just as [10] examined the difference between certified and non-certified learners in the MOOCs, we examine the difference between students who are not graded at the end of the course (MIT Freshmen do not receive letter grades for their first semester) and students who are being assigned grades at the end of the course (Freshmen in their second semester).

Finally, [5] has performed a very broad survey of MOOC learning and effects on student performance in programming classes. We take some of the work done in that paper and expand it. [5] has examined the factors that are most strongly correlated with high performance overall in the class, on exams, and successful completion of the course. Once again, we apply and expand those ideas to the residential data.

1.3.2 Predicting Student Performance

Additionally, there has been a large push to start attempting to leverage Machine Learning techniques to be able to aid in the educational research field. One such use

for machine learning in educational research is to able to predict student performance in a course. [11] and [9] have performed an analysis of several existing works that have tried to apply machine learning techniques to educational problems. They have found several successful examples of machine learning being applied to attempt to predict student performance.

On top of that, [4] has performed similar research to ours. They have built out student profiles for groups of students at several universities, and have predicted both next term grades that the students will earn, and assignment grades. They also did these predictions on MOOCs as well, showing that machine learning algorithms can successfully be used to predict student performance in residential and online classes alike.

1.4 Course Summaries

For all 4 courses being looked at (6.00.1x, 6.00.2x, 6.0001, and 6.0002) there are three different types of assignments. Problem sets are long assignments that can be completed over the course of one to two weeks. They are typically challenging programming projects which cover the course material that has been reviewed before the due date. Finger exercises are shorter, more frequent assignments that are meant to test single topics in the course. Typically, they are less than 10 lines of code and are associated with specific lectures. There are also quizzes which are administered. In the residential version there are "micro quizzes" which are short in class assignments that are done online and taken for a grade. They usually test topics that have been taught in the previous 1 to 2 weeks. Longer quizzes are also administered in the MOOC version and in 6.0001. The final exam is one such instance.

1.4.1 MOOC

6.00.1x

This course is designed for students ranging from limited programming experience prior to taking it (i.e. basic online self-training, high school level programming education, etc.) to students with no previous programming experience. The topics taught in this class include the basics of programming, syntax, common data structures, a brief introduction to object oriented programming, recursion, runtime analysis, and implementation of simple algorithms.

6.00.2x

This course is designed for a more advanced programmer. It is highly recommended for students to have taken at least 6.00.1x before attempting 6.00.2x. The concepts that are taught in this class included dynamic programming, search algorithms, stochastic algorithms, dealing with experimental data and the fundamentals of machine learning.

The following assignments contribute to the overall grade of the students for both 6.00.1x and 6.00.2x

1. 4 Problem Sets
2. Midterm Exam
3. 2 Problem Sets
4. Final Exam
5. Finger Exercises (throughout the term)

1.4.2 Residential

6.0001

This is the residential version of 6.00.1x. This course is also designed for students with little to no programming experience. It is a graduation requirement for all computer science majors, as well as majors in other disciplines such as Biological Engineering.

The assignments for 6.0001:

1. 5 Problem Sets (30%)
    (a) 5 Autograder Grades (70% of each problem set)
    (b) 4 Checkoff Grades - Discussed in person with a staff member (30% of each problem set)
2. Mandatory Finger Exercises (10%)
3. 3 Micro Quizes (20% taken from best 2 of 3)
4. 1 Final (40%)

6.0002

This is the residential version of 6.00.2x. This course is designed for students with previous programming experience. It is a graduation requirement for Biological Engineering majors, among other departments.

The assignments for 6.0002:

1. 5 Problem Sets (35%)
    (a) 5 Autograder Grades (70% of each problem set)
    (b) 4 Checkoff Grades (30% of each problem set)
2. Mandatory Finger Exercises (10%)
3. 3 Micro Quizes (30% taken from best 2 of 3)

4. 1 Final (25%)

The main differences between that residential and the MOOC versions are in the fact that there are 2 lectures each week for the residential version, each of which lasts for one hour and a half. In addition, there are optional recitations that are held in person once a week, and there are office hours that are held every day. Students can walk into office hours at any time and get help from either a lab assistant (LA) or a teaching assistant (TA).

Also, it must be noted that the grading for the spring runs of 6.0001 and 6.0002 have been drastically altered. Due to the disruption caused by covid-19, the final exam for 6.0001 got cancelled, as did the last problem set and the checkoff grade for the 4th problem set. In response to these disruptions, we decided to change the grading weights for the spring run of 6.0001 to be 45% problem sets, 45% microquizzes and 10% finger exercises.

In addition, grading for all subjects in the second half of the spring semester got switched to Pass or No Record. Therefore, there were no final grades given for 6.0002.

We will be discussing our findings related to these courses in later sections.

Chapter 2

Datasets

Educational Data Mining (EDM) is a rapidly expanding field. Machine learning is quickly being applied to nearly every industry in order to improve processes or expand capabilities. There have been major leaps in EDM efforts over the past few decades, but the efforts are always limited by the ability to gather specific information about the student's behaviors on a large enough scale [8]. Here, we will discuss the two datasets that we have been performing a range of analyses on.

2.1 MOOC Dataset

Although there have been many runs of 6.00.1x and 6.00.2x, we are limiting our analysis to just one run of the course. Specifically, we are looking at the most up to date data, the Spring term of 2019. The reason that we are looking at a limited amount of the data is because there has already been extensive research performed on MOOC data, and even more specifically MOOC data from these course [3] [7] [5].

In the datasets that we are using, 6.00.1x has 57,418 students initially enrolled in the course. 6.00.2x has 7,692 students initially enrolled. However, students can have three different degrees of participation. The first level is simply having had viewed the course. The second level is having had explored the course, which means that they visited over half of the chapters. The highest degree of participation is having completed the course. When trimming the datasets based on participation in the

course, we can see that these numbers rapidly decrease. Out of the 57,418 students enrolled in 6.00.1x only 35,874 viewed the course, 4,286 explored the course, and only 1190 completed the course. In 6.00.2x, out of the 7,692 enrolled students, 3,656 students viewed the course, 582 students explored the course, and only 309 completed the course.

The features that are included in the MOOC data include: name, geographic location, amount of pages visited, number of problems completed, amount of time spend watching videos, number of videos watched, how many times the videos were paused or fast-forwarded or re-winded, grades on each assignment, number of attempts for each problem, and many other relevant features to student behavior in a course. In addition, the data includes their result in the course. Results included final overall percentage, whether they completed the course or not, and whether they passed the course or not.

In addition to all of the data gathered in regards to the behavior of the students and their background, there is also data gathered on their forum activity. In both 6.00.1x and 6.00.2x, there is a forum on which students can post questions, and either staff members or other students can answer the questions or comment on posts.

2.1.1 Gender

We later investigate whether or not gender is a determining factor for performance in 6.00.1x, 6.00.2x, 6.0001 and 6.0002. Prior work has been done in investigating the effect of gender in other programming classes, and it has shown that gender, when isolated as a feature, does not have a causal linkage to performance in a course [2].

2.1.2 Prior Experience

Previous research into 6.00.1x and 6.00.2x also centered around measuring student performance based upon previous experience that they had programming. Although the metrics that they used differed from ours, there are certainly links that can be drawn between the two datasets. Figure 2-2 shows the breakdown of experience by

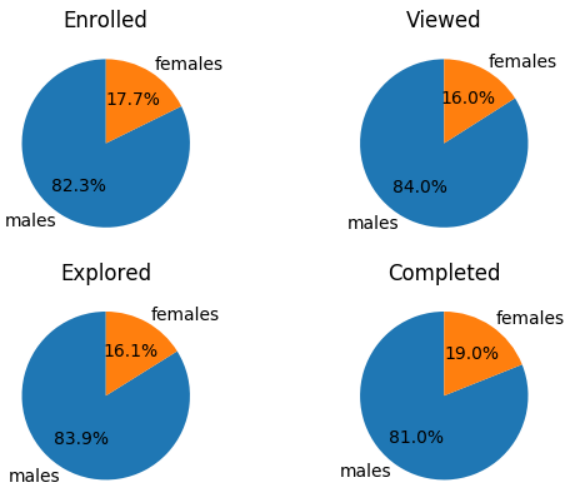

Figure 2-1: Gender breakdown for 6.00.1x and 6.00.2x

student for the MOOC versions.

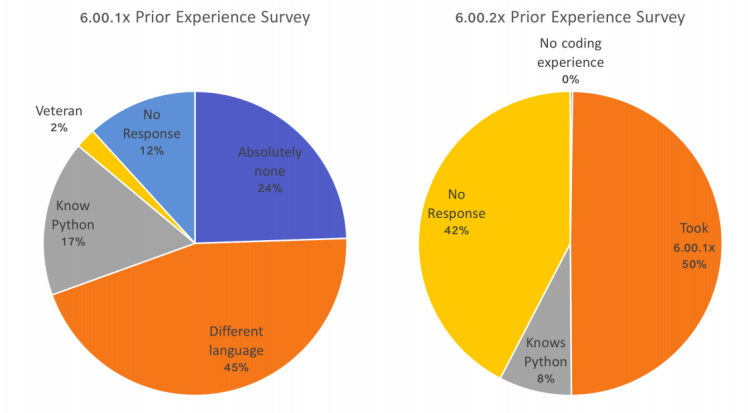

Figure 2-2: The distribution of prior experience in 6.00.1x and 6.00.2x. Figure taken from [7]

2.2 Residential Dataset

The residential datasets are significantly smaller than the MOOC datasets, however, the data is more rich given the fact that we were able to design which features were being collected, and there is more depth to the course when being taught in person.

Given the fact that not all students completed the background survey, we were able to record the profiles of 274 students for the fall run of 6.0001. For the fall run of 6.0002, there were 214 students profiles that we collected.. For the spring run of 6.0001, we collected 282 profiles.

There were several sources from which we mined student data:

- The course website: From here we were able to gather information on the student's autograder score for each problem set, their activity on the website, how often they clicked on buttons on the website and what they clicked on, when they downloaded problem sets, when they downloaded lecture notes, which pages they visited and when, etc.

- The help queue: From here we were able to see each time a student went to office hours and when, for what assignments they asked for help, and when they received their checkoffs for the problem sets.

- Forums:

  - Piazza: for the fall semester, we used a forum website called Piazza. On Piazza, we were able to see how many days they were active on the website, how many questions they posted, how many comments they posted, how many answers they contributed, and how many questions they viewed.

  - Ed: for the spring semester we switched to use a new forum website. The data that we were able to collect from Ed was the same as the data collected from Piazza.

- Background Survey: As part of problem set 0 (an ungraded problem set that all of the student do in order to make sure that their computer systems are ready for the start of the course) the students are all asked to fill out a survey detailing some of their background knowledge on the material in the course. Background programming knowledge is extremely important when trying to find causally linked factors in performance analysis. The questions include:

1. What is your Grade Level?
2. Approximate Lines of Code written before enrolling in 6.0001?
   - options are 0, 50, 100, 200, 300, 500, 1000, 5000, 10000
3. Prior Programming Experience?
   - None
   - HTML
   - AP Computer Science
   - Onlince coding course
   - Programming in another language
   - Programming experience in python
   - OCW or edX
   - College education in another language
   - college education in Python
4. Why did you enroll in 6.0001?
   - To learn how to program
   - To fulfill a course requirement
   - To get a good grade
   - other
5. Select all of the resources that you are aware of or have used to help learn how to program.
   - Google
   - Online Coding Courses
   - Stack Overflow
   - Friends

Here we must also note that the data for the spring semester got corrupted due to the covid-19 pandemic. Part way through the semester, all students were required to leave campus. This caused many anomalies in the data.

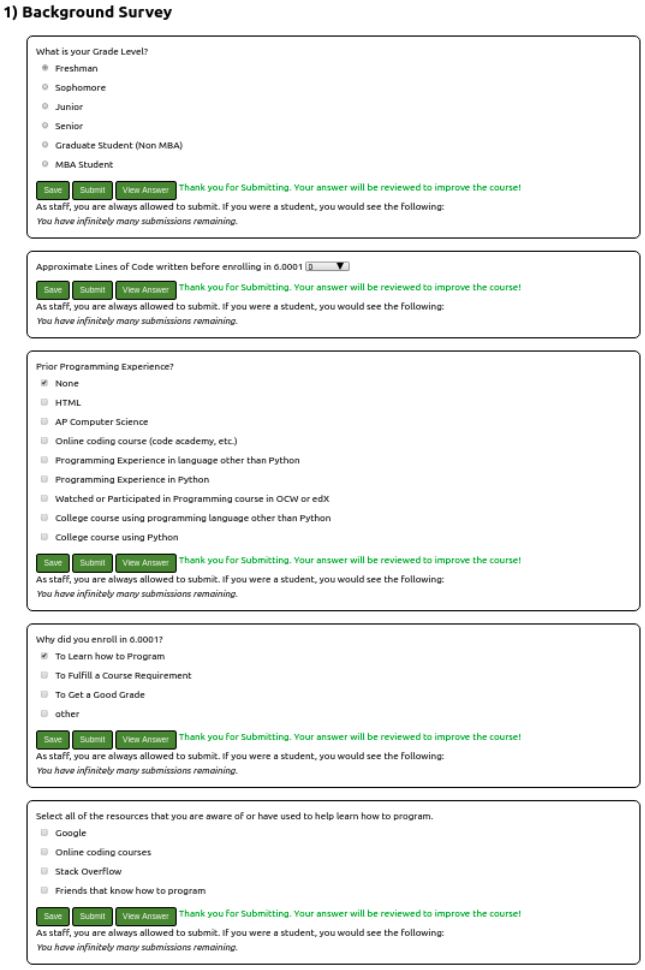

Figure 2-3: The survey that all students must take at the beginning of the course

2.2.1 Residential Student Profile

In designing the course website, which we will talk about in more detail in the next section, there were many careful design decision that needed to be made in order to gather data that would be useful in the analysis and prediction parts of our research. While we did not want to gather information that is superfluous, we also decided to lean towards the side of overcollection rather than undercollection of data.

Therefore, the profiles that we collected on each student ended up being very comprehensive. This allowed us to analyze many factors that could possibly contribute to increased learning and performance. See table 2.1 for a complete list of features and table 2.2 for a complete list of the background information that we collect.

Overall	course
FE	PS1
activity_count	avg_pset_time_remaining_after_last_submit
avg_submits	avg_workahead_time
cheat	click_count
forum_contributions	forum_days
forum_endorsements	forum_notes
forum_questions	forum_views
queue_add	queue_all
queue_count	queue_help
queue_pset1	queue_pset2
queue_pset3	queue_pset4
queue_pset5	semester
unique_problems_attempted	number_of_correct_problems
avg_attempts_per_problem	total_attempts
l1_unique_problems_attempted	l1_number_of_correct_problems
l1_total_attempts	l2_unique_problems_attempted
l2_number_of_correct_problems	l2_total_attempts
l3_unique_problems_attempted	l3_number_of_correct_problems
l3_total_attempts	l4_unique_problems_attempted
l4_number_of_correct_problems	l4_total_attempts
l5_unique_problems_attempted	l5_number_of_correct_problems
l5_total_attempts	l6_unique_problems_attempted
l6_number_of_correct_problems	l6_total_attempts
l7_unique_problems_attempted	l7_number_of_correct_problems
l7_total_attempts	l8_unique_problems_attempted
l8_number_of_correct_problems	l8_total_attempts
l910_unique_problems_attempted	l910_number_of_correct_problems
l910_total_attempts	l11_unique_problems_attempted
l11_number_of_correct_problems	l11_total_attempts
nevents	ndays_act
nplay_video	nchapters
nproblem_check	language_nevents
nshow_answer	nvideo
nvideos_unique_viewed	nvideos_total_watched
nseq_goto	nseek_video
npause_video	avg_dt
sdv_dt	max_dt
n_dt	sum_dt

Table 2.1: List of all of the features that have been gathered on the student **behavior** for analysis and prediction

grade_level	lines_of_code
exp_None	exp_ap_comp_sci
exp_html	exp_non_python
exp_non_python_college	exp_ocw
exp_online_class	exp_python
exp_python_college	forum_answers
awr_friends	awr_google
awr_online	awr_stack_overflow
mot_gpa	mot_learn
mot_other	mot_requirement
dep_AeronauticsAndAstronautics	dep_Architecture
dep_BiologicalEngineering	dep_Biology
dep_BrainAndCognitiveSciences	dep_ChemicalEngineering
dep_Chemistry	dep_CivilAndEnvironmentalEng
dep_EarthAtmosPlanetarySci	dep_Economics
dep_ElectricalEngComputerSci	dep_Employee
dep_HealthSciencesTechnology	dep_HumanitiesEngineering
dep_MIT	dep_Management
dep_MaterialsScienceAndEng.	dep_Mathematics
dep_MechanicalEngineering	dep_NuclearScienceEngineering
dep_Physics	dep_PoliticalScience
dep_Undeclared	dep_Undesignated
dep_Unknown	dep_UrbanStudiesAndPlanning
dep_department	Harvard
MIT	Wellesley
f	m

Table 2.2: List of all of the features that have been gathered on the student **background** for analysis and prediction

2.2.2 Dataset Breakdown

When performing analysis on the data from programming classes, we are trying to discover causal relations between behaviors and performance. In order to make sure that we are discovering a causal link, and not simply correlation, we need to try our best to prevent confounders from corrupting our results. That is why breaking down the datasets is very important.

Therefore, we will examine the datasets categorized based upon several axes in order to corroborate past findings and develop new insights: gender, grade level, and prior experience.

Gender

At MIT, approximately 48% of the student body is female, and 52% of the student body is male [1]. However, the choices of majors still tend to differ between genders. In the computer science department, for example, there are approximately 1353 total undergraduate students. However, only about 554 (40%) of the undergraduate computer science majors are female.

Surprisingly, however, the gender breakdown for both 6.0001 and 6.0002 seem to be the reverse of overall Institute demographics. For all 3 of the residential courses being examined, the majority of the students enrolled are female. The exact distributions are shown in figure 4-2.

Grade Level

One factor that we suspected may impact performance would be grade level. There are 6 different grade levels that we examined for the residential data: Freshman, Sophomore, Junior, Senior, Graduate Student, MBA student. Each course had a majority freshmen, with sophomores being the second most common. There were approximately an equal number of juniors and seniors who completed the course.

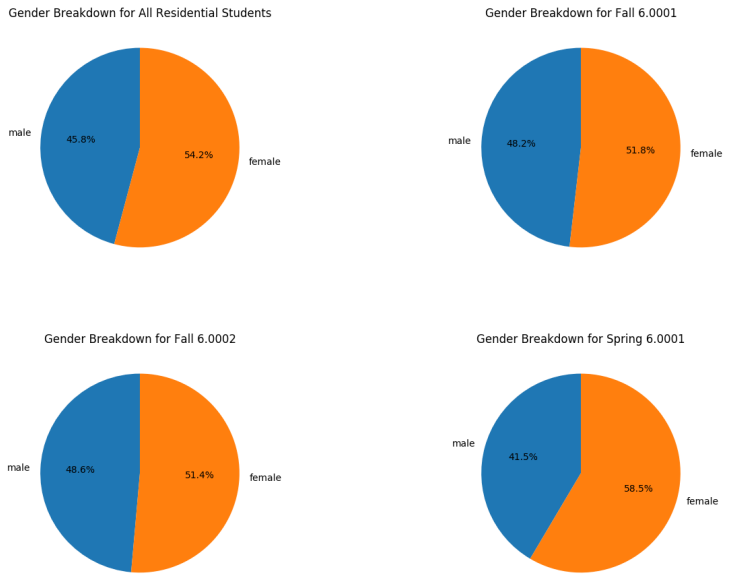

Figure 2-4: Gender breakdown for all three residential courses, and overall. We found that each residential course was majority female.

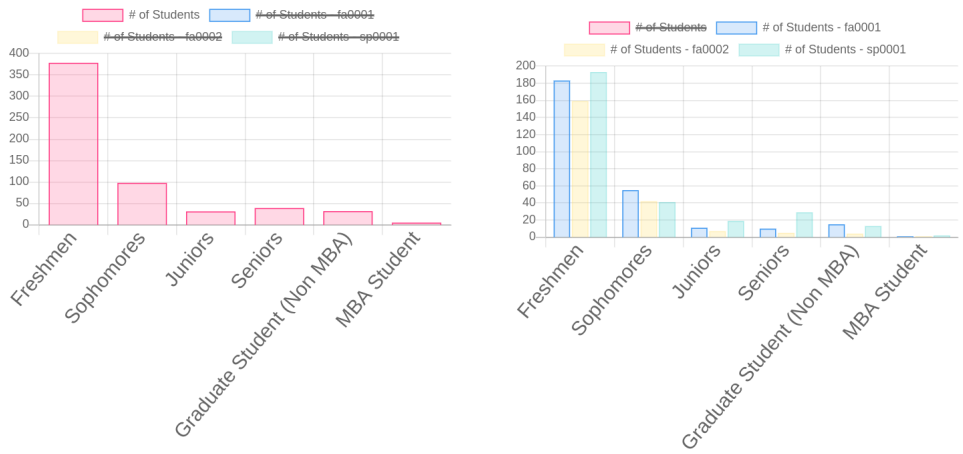

Figure 2-5: Grade Level breakdown. The graph on the left shows the grade level breakdown for the combination of all three courses. The graph on the right displays grade level breakdown for each course individually.

Course

As mentioned before, we are analyzing data from 3 separate residential courses. The 3 courses are fall 6.0001, fall 6.0002, and spring 6.0001. The course material differs between 6.0001 and 6.0002, but stays the same between different runs of 6.0001. However, due to the evacuation of campus because of covid-19, 6.0001 in the spring differed in final grading than 6.0001 in the fall. Therefore, when looking at the data, we have also examined the data from each course in isolation in order to try to normalize any noise presented due to the differences of the courses.

Prior Experience

Another very important distinction to make between students is their level of experience programming prior to taking the course. We asked students to self report previous experience along two axes: amount of code written (quantitative) and specific experiences they had and courses had taken (qualitative).

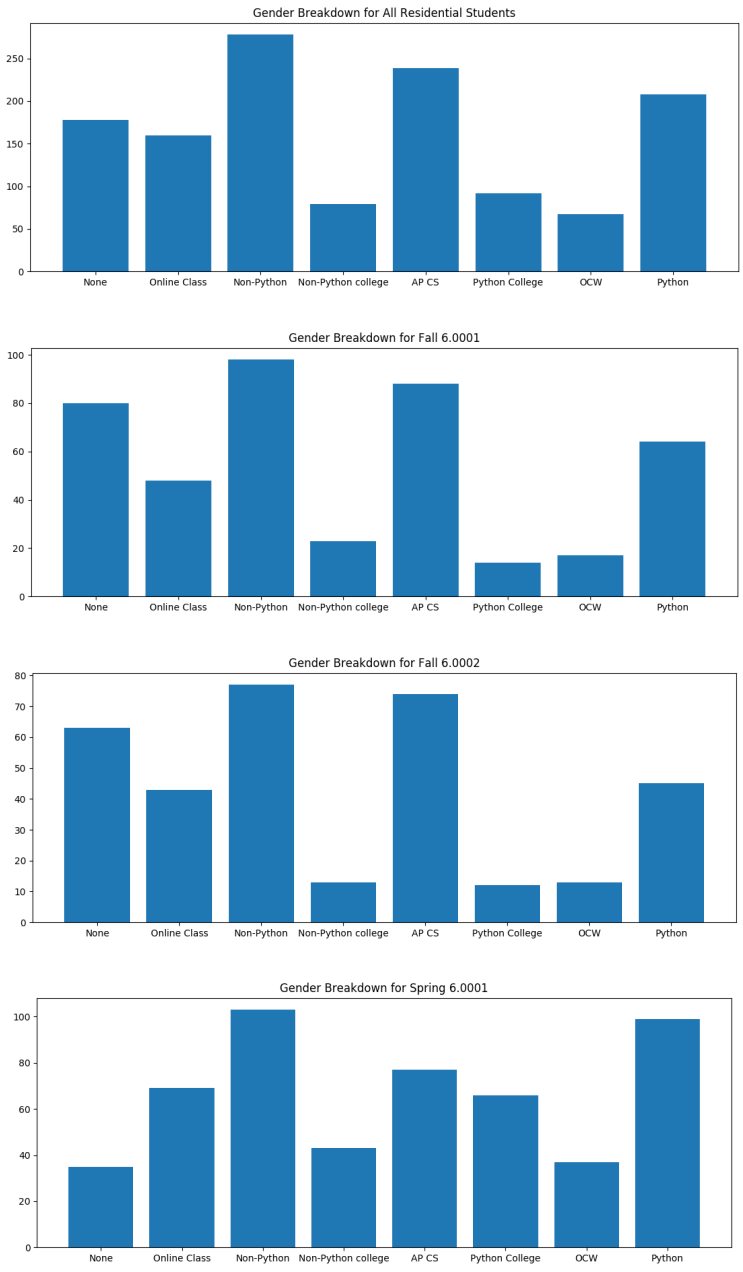

Figure 2-6: The different qualitative prior experiences students taking the residential versions of the course have.

Chapter 3

MIT Residential Website

3.1 Overview

Prior to this school year, the residential 6.0001 and 6.0002 had a very decentralized infrastructure for their course website. There was a separate website for submitting problem sets, seeing grades, the help queue, taking exams, and accessing lecture notes and course material.

None of the backends were well organized or secure, and the different sites were not connected with one another. For the data to be updated on one page, it would have to be manually transferred over from one to another. In addition, there was no automatic grading of assignments. All submissions needed to be manually gathered and ran in an autograder for the results to be generated.

Seeing this as a major problem for both the efficacy of the course and the ability to mine educational data on the students, the first major contribution of our work was in the construction of a course website for 6.0001 and 6.0002. The goal of this new website was to build a centralized platform on which students could perform all of their class related tasks, as well as an information hub where they could get all lecture material and course information. We wanted to eliminate the need for many of the various websites that were previously being used, as well as create a new website with a much more user friendly interface for the sake of the students, and a much more robust and capable backend that would automate so many of the tasks that

staff members otherwise would have had to tediously perform.

The unique position we were in enabled us to tailor the data that we are gathering and analyzing. So, not only was this a contribution to one of the largest course in the Electrical Engineering and Computer Science department, but this was also an essential aspect to our research into student behaviors' causality in regards to performance in programming courses.

3.2 CAT-SOOP

The course website was built using an already existing course website creation software called CAT-SOOP [6]. CAT-SOOP is a course website management software that takes care of many functionalities. Functionalities that it provides include:

- Content Presentation
- Assignment grading
- Queue system
- Control of release and due dates for problems
- On disk database management

The base functionality of the course website was built using the CAT-SOOP software. In addition, the thorough logs of student activity that CAT-SOOP keeps track of assisted in great ways for the data mining process.

3.3 Design

Some of the essential features that we included on the new course website that we built for 6.0001 and 6.0002 are:

- Access to all of the problem sets.

- Problem set submission and autograding. This takes away the need for staff to go in and manually run each student's code through a tester.
- Course calendar. Including automated content updates and lecture material releases.
- Lecture notes and other material for the students
- Course syllabus and style guide
- Student's ability to see all of their grades thus far in the course.
- A help queue for the students to use while in office hours
  - This includes checkoff capabilities, so staff can administer checkoffs to the students when they come in for office hours, and this grade can be automatically propagated to the grade book.
- Grade book for the staff members to be able to see all grades
  - Staff can search for a specific student and see all of their grades on each of the assignments, as well as a history of their submissions and view their code and score for each submission
  - Staff can also see checkoff grade and other information associated with the checkoff such as length of checkoff meeting.
  - Staff can also search for all submissions for assignments to see the progress being made for the class a while
  - Staff can look for students' code that did not run properly and download submissions that have been flagged by the system for manual grading.
  - Staff can generate a .zip file of all problem set and micro quiz submissions for local examination
  - Data vizualization page so that the staff can examine trends in course data and student performance and behavior. (more on that in section 3.4.1)

- Exam administering software. Initially, we were able to leverage the built in CAT-SOOP infrastructure for the in class exams, however, when the semester became remote, we designed a new system to administer exams on the website. This included allowing for a 12 hour window for taking the exam, but only allowing a student 30 minutes to complete the exam relative to their start time.

For further information on the design of the course website, including how to maintain and operate it, please see the Appendix at the end of the paper. In it we include a highly detailed explanation of the many parts of the website and how they all interact.

3.4 Data Collection

All of these features, and many more that we have added throughout the first year of the new course website's usage, allowed for a much better interface for the students, a much easier experience for the staff member, and it also allowed for a much greater degree of data analytics.

With the course website being completely customizable, we were able to add several data mining features to the website. For example, we were able to collect all of the queue activity, all of the page loads, assignment and notes downloads, clicks, submissions, and many other features that have turned out to be of great use in analyzing student performance based upon their behaviors.

3.4.1 Data Visualization

In addition to all of the features of the course website, there is also a page on which staff can access visualizations of all of the data being collected in the course. This resource is useful in analyzing and detecting trends in data, and in seeing how student performance is on each assignment.

The page is connected directly to the database of the current run of the website, and is also easily adaptable to be able to take in data from previous semesters.

Currently, it is able to analyze data from fall 6.0001 and 6.0002, as well as spring 6.0001.

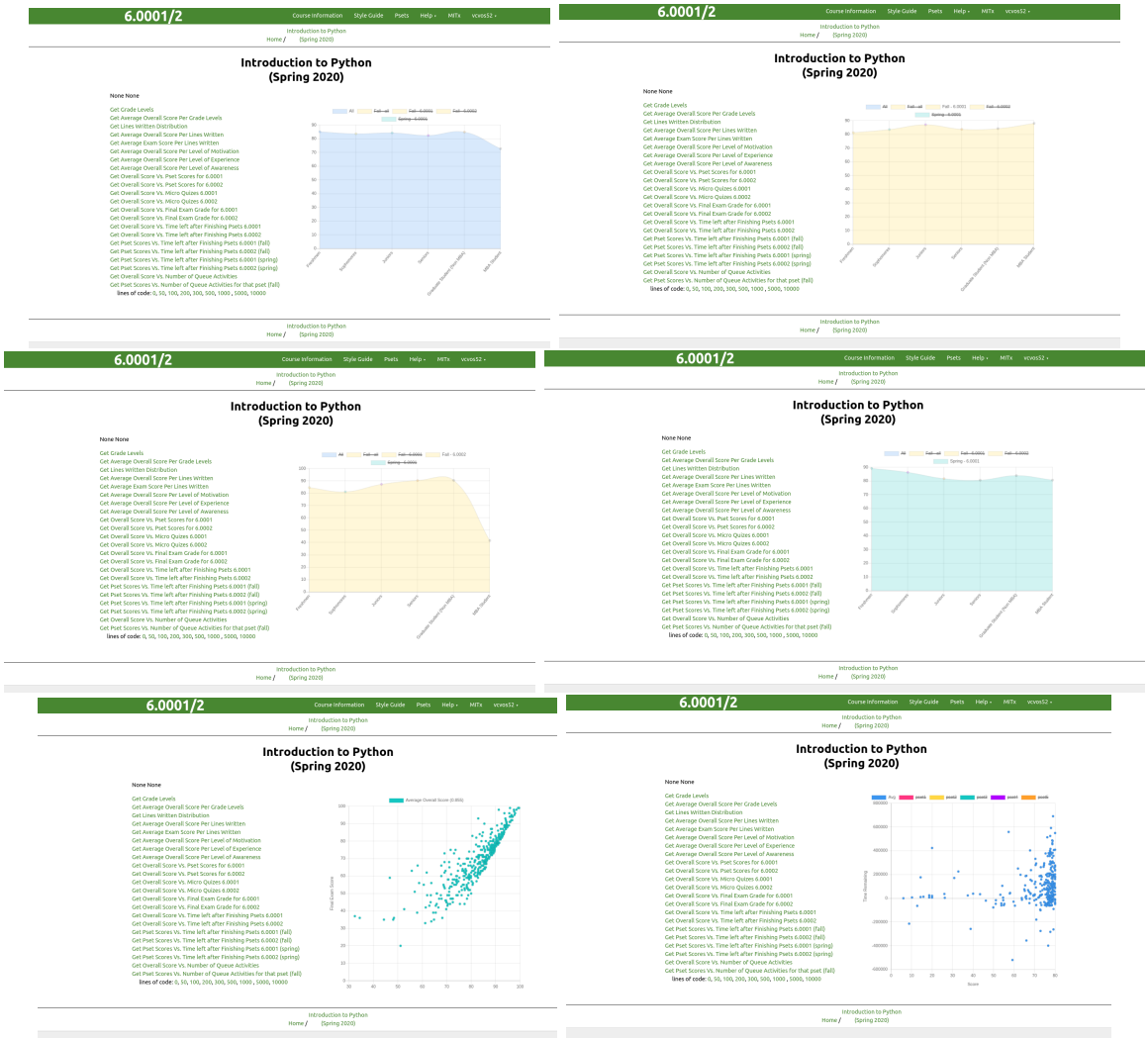

Figure 3-1: Taken from visualization page of course website. Example charts of getting performance based on grade level. Can be used to analyze trends.

Chapter 4

Analysis of Behaviors

In this chapter, we will be looking at the specific behaviors of students who take both the residential and MOOC programming courses. We will be looking for factors that correlate with performance in the courses, both positively correlated and negatively correlated. Then we will be examining the causality between variables in order to make a determination as to whether or not there exist features that can predict how a student will perform in the course.

We will start off with discussing my results in analyzing the data that we gathered from the residential courses. We will examine specific features that were strongly correlated with performance in the course, as well as features that were surprisingly not correlated with performance. From this information, we will reason about the causality of select features in determining performance, and present findings on which behaviors lead to better results for which profile of student. Each feature will be examined for 3 dataset segmentations: grade level, experience level, and gender.

Subsequently, we will examine the data from 6.00.1x and 6.00.2x. we will generally be looking over the data in similar ways to how we analyzed it for the residential version of the course. We will be looking at the MOOC data through the lens of some of the research that has already been completed in this field.

Finally, we will dedicate a section to comparing and contrasting our findings from the MOOC courses and the residential courses. We will discuss the possible reasons for differences, and will discuss the differences in learning between in person education

and online education, and what we found to be lacking in free online education as compared to in person university education.

4.1 Residential Analysis

The goal of this part of the research was to identify features of the students that led to better performance in the course. The first step in doing so was to identify features that were correlated with performance in the course. To do so, we collected all of the aforementioned data, and analyzed it from several different angles. See table 4.1 for the 80 most highly correlated features with overall final grade in the course. We used both Spearman and Pearson correlation. Spearman correlation measures the monotonic relationship between variables. In other words, it measures how well two variables move up and down together. Pearson correlation measures the linear relationship between variables, meaning that the degree to which variables move together is also measured.

The first surprising finding that we can see in table 4.1 is that above all other background features and behaviors that a student can exhibit in the course, the amount of time in which they "work ahead" (in other words, the difference in time between the due date and their final submission) is the most strongly correlated factor with success in the course. This is true for all grade levels and experience levels. Therefore, this confirms the idea that starting (avg_workahead_time) and finishing (avg_pset_time_after_last_submit) your work early is much more conducive to more learning and better performance when compared to procrastination.

4.1.1 Experience

As we can see from table 4.1, many of the features that we collected through the course website correlated with overall final grade in the course. This is indicative that performance can be predicted to some degree based upon knowledge of background and behavior in the course.

However, one thing that we can see is that some of the most positively corre-

Feature	Pearson Coeff	Spearman Coeff	Description
max_dt	-0.308	-0.363	Time spent on mitx
sdv_dt	-0.306	-0.350	SD of time on mitx
exp_None	-0.296	-0.303	No prior experience
nchapters	-0.295	-0.347	Chapters accessed
ndays_act	-0.292	-0.374	days active
sum_dt	-0.281	-0.395	sum of time active
nproblem_check	-0.279	-0.405	# of practice problems
nevents	-0.273	-0.393	# of events
n_dt	-0.270	-0.393	
avg_dt	-0.265	-0.329	avg time on mitx
avg_attempts_per_problem	-0.252	-0.333	
dep_MIT	-0.242	-0.123	MIT faculty
language_nevents	-0.234	-0.373	
nvideos_unique_viewed	-0.212	-0.310	
queue_help	-0.186	-0.263	# times OH help
nplay_video	-0.185	-0.3034	
nvideo	-0.185	-0.304	
l1_total_attempts	-0.184	-0.239	lec 1 optional problems
npause_video	-0.181	-0.311	
nseq_goto	-0.178	-0.347	
l2_total_attempts	-0.178	-0.206	
l1_unique_problems_attempted	-0.176	-0.236	
l2_unique_problems_attempted	-0.175	-0.203	
nseek_video	-0.175	-0.282	
nshow_answer	-0.171	-0.307	optional problem
l1_number_of_correct_problems	-0.163	-0.229	
nvideos_total_watched	-0.162	-0.304	
total_attempts	-0.157	-0.306	optional problems
unique_problems_attempted	-0.156	-0.298	optional problems
queue_all	-0.139	-0.260	Office hours
l2_number_of_correct_problems	-0.134	-0.177	
l6_unique_problems_attempted	-0.134	-0.213	
queue_pset2	-0.131	-0.238	OH for pset 2
l6_total_attempts	-0.128	-0.224	
cheat	-0.126	-0.130	Times caught cheating
l5_unique_problems_attempted	-0.125	-0.179	
l5_total_attempts	-0.123	-0.183	
l4_total_attempts	-0.120	-0.185	
l7_total_attempts	-0.116	-0.205	
queue_pset4	-0.115	-0.186	
queue_pset1	-0.114	-0.232	
l7_unique_problems_attempted	-0.111	-0.195	
l4_unique_problems_attempted	-0.111	-0.179	
number_of_correct_problems	-0.102	-0.258	optional problems
l6_number_of_correct_problems	-0.089	-0.188	
queue_pset5	-0.085	-0.182	
l910_unique_problems_attempted	-0.080	-0.153	
dep_Unknown	-0.077	N/A	Unkown department
Wellesley	-0.075	N/A	Wellesley cross reg.
mot_learn	-0.073	N/A	reason in class-to learn
dep_UrbanStudiesAndPlanning	0.070	N/A	Urban studies major

Feature	Pearson Coeff	Spearman Coeff	Description
dep_ChemicalEngineering	0.071	N/A	
MIT	0.074	N/A	MIT student
mot_other	0.074	N/A	other motivation
dep_ElectricalEngComputerSci	0.081	N/A	
click_count	0.084	N/A	# clicks on website
exp_ocw	0.103	0.127	prev exp - ocw
forum_views	0.108	0.153	forum activity
exp_online_class	0.111	0.114	prev experience
f	0.114	N/A	Gender - female
awr_online	0.118	0.134	awareness of resources
mot_gpa	0.121	0.131	motivation - GPA
awr_friends	0.131	0.128	Help from friends
exp_html	0.133	0.140	experience HTML
exp_non_python_college	0.134	0.158	exp in other language
exp_python	0.136	0.152	
exp_python_college	0.151	0.177	took python course
mot_requirement	0.155	0.133	motivation: course req
exp_ap_comp_sci	0.166	0.148	
awr_stack_overflow	0.183	0.187	
exp_non_python	0.188	0.195	
activity_count	0.240	0.312	page loads on website
FE	0.295	0.183	finger exercises
semester	0.312	0.409	spring/fall semester
lines_of_code	0.353	0.372	prev experience
PS1	0.355	0.356	
avg_workahead_time	0.461	0.524	first submit after release
avg_pset_time_after_last_submit	0.479	0.525	last submit before deadline

Table 4.1: The 80 most highly correlated features with Overall final grade in the course.

lated and negatively correlated factors associated with success in the course are prior experiences that the students had. These are features with prefix "exp_". While these are extremely valuable in predicting performance, they could certainly serve as confounders to other features. We will keep this in mind when analyzing the other behaviors.

As we can see, having no experience has a -0.296 Pearson correlation and a -0.303 Spearman correlation with overall grade in the course. In addition, all forms of previous experience is positively correlated with overall grade in the course. Knowing other programming languages (0.188 and 0.195), having taken AP computer science in highschool (0.166 and 0.148) and having previous experience in programming in python (0.151 and 0.177) all correlated with better performance in the course. Relative to the other features, these are some of the more strongly correlated features.

In addition, we also ask students to approximate how many lines of code they have written before entering the course. We see a similar trend. The number of lines of code that the students write have correlation coefficients of 0.353 and 0.372, the fourth highest coefficients measured.

One thing that did stand out as we were analyzing the data is that for the fall courses (6.0001 and 6.0002) the students with the most experience (lines of code in this case) performed the best in the course. However, in the spring run of 6.0001, the people with the most experience performed significantly worse than students with less experience. People who had written more than 10,000 lines of code before enrolling in 6.0001 in the spring performed as well as those who had written 200 lines of code. We found that the reason for this is because in all runs, more experience was strongly correlated with a higher Exam average, however problem set average is much less correlated with experience. This is because experienced students are able to perform very well on the quizzes because they already have the knowledge to complete the general problems on the quizzes before they enter the course. However, problems set performance is more of a function of the time taken to complete them. Experienced students are likely taking less time on assignments while inexperienced students are taking more time. This is illustrated in Figure 4-2. This adversely affected more

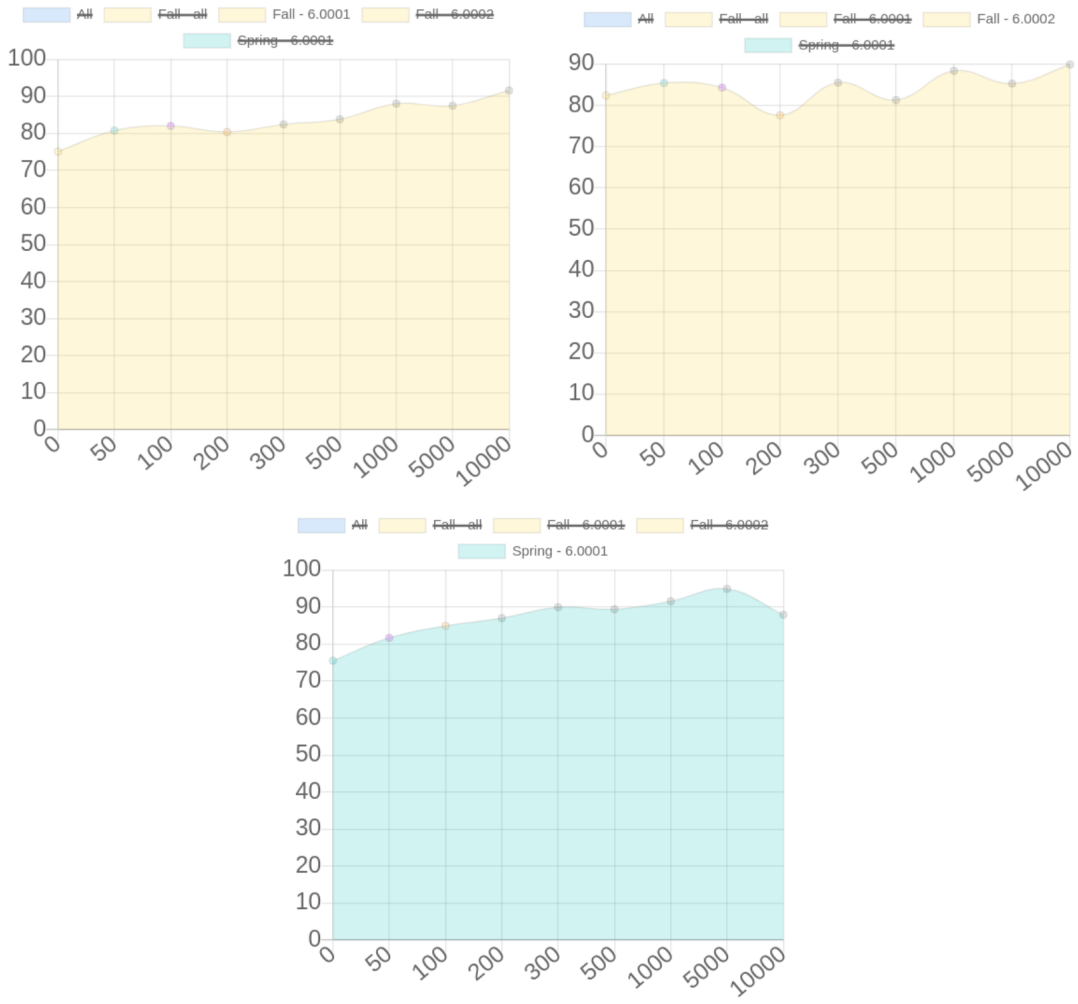

Figure 4-1: Plotting overall grade in the course vs. lines of code previously written. This is another, more quantitative way to measure past programming experience.

experienced students in 6.0001 in the spring because the grading rubric for the course was altered due to covid-19. It was altered in a way such that problem sets were much more heavily weighted than they otherwise would have been.

The fact that experience is correlated with performance is not surprising, however the extent to which it is correlated with performance in the class is. This is an indicator that, while the courses are tailored to be fair to all students, performance and learning are certainly determined in large part by past experience. This can be seen by the grade distribution for the 6.0001 final exam in figure 4-3. However, another interesting finding that we made was that prior experience becomes significantly less

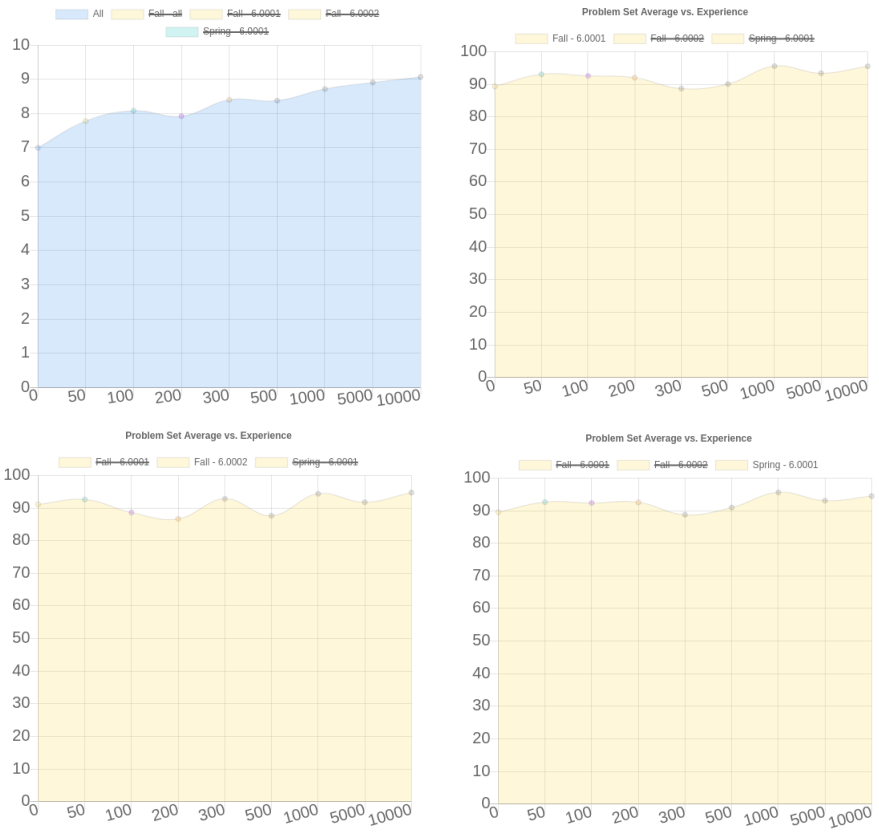

Figure 4-2: Plot of average overall exam score vs. lines of code written, and plots of average problem set score vs. lines of code. We can see that it monotonically increases as experience increases for the exam score, but it is very weakly correlated for problem set score. This confirms our theory that more experienced students performed poorly in the spring course because of the grade reweighting.

important in 6.0002. The effects of experience in each individual course are illustrated in Table 4.2. The conclusion drawn from this, and used in our predictive modelling, is that background should be weighted more when predicting student performance in 6.0001, and behavior should be more weighted when predicting student performance in 6.0002.

4.1.2 Grade Level

Surprisingly, the grade that the student is in was not strongly correlated with overall grade in the course. For example, in the fall run of 6.0001, the grade with the lowest overall average score in the course were the freshmen with an average of 81%, and

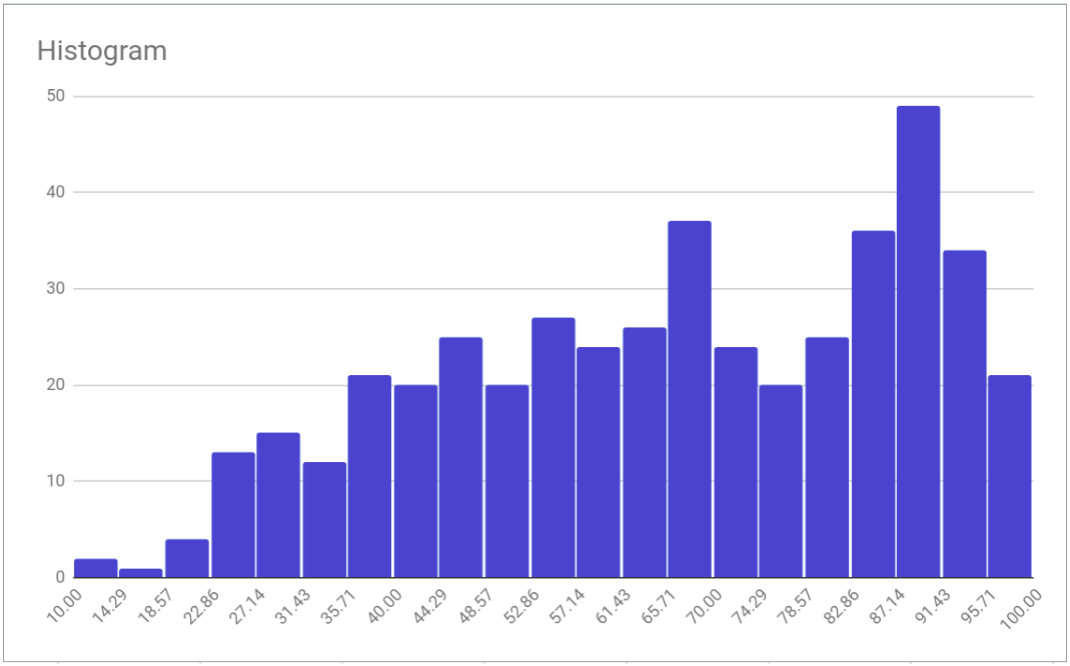

Figure 4-3: 6.0001 Final Exam grade distribution. We can see that there are essentially two normal distributions being formed. This is due to the fact that people with more experience perform significantly better.

Course	Most Correlated	Second Most Corr	Most Negatively Correlated
Overall	Non-Python language (0.19)	AP CS (0.17)	None (-0.30)
Fall 6.0001	Non-Python language (0.3)	AP CS (0.26)	None (-0.39)
Fall 6.0002	HTML (0.16)	College course (0.14)	None (-0.11)
Spring 6.0001	AP CS (0.25)	Non-Python language (0.14)	None (-0.24)

Table 4.2: How correlated experience is with overall grade in each course. We see that experience has much stronger correlation with overall grade in 6.0001 than in 6.0002. This points towards the fact that, while there is a learning curve in programming, students are able to climb up it fairly quickly.

the highest average overall grade was the junior class with 86% average overall grade. All of the other grade levels were within that rather small range. This points to the fact that grade level may not have an impact on performance in the course.

However, as stated before, prior experience is an impactful confounder, and must be normalized for when looking at any other subset of the data. Looking at table 4.3, we see that the level of inexperience is fairly evenly distributed across all grade levels. Interestingly, if we take a subset of just students with no experience from the entire dataset, we find that being a freshman is the most negatively correlated factor with overall grade. Therefore, when normalized for experience, younger students do perform worse.

Another possible confounder for this would be the fact that MIT freshmen are on pass/no record their first semester at MIT. To test if this is the reason for freshmen doing worse than upperclassmen when accounting for experience differences, we compared how freshmen as a whole did in the fall run of 6.0001 (pass/no record) to how they did in the spring (when they were assigned grades). Surprisingly, we found that freshmen performance was not impacted by whether they received grades or not. Their average grades on each problem set and on each micro quiz was about the same when comparing across semesters. In addition, how much they procrastinated on each assignment/how much time they spent on assignments (measured by when they submitted relative to the due date) was also not significantly affected. All of this indicates that PNR does not impact student performance, and therefore does not explain away the fact that when normalizing for experience, freshmen do significantly worse than other grade levels.

Freshmen	Sophomore	Juniors	Senior	Graduate	MBA
24%	23%	12%	18%	23%	33%

Table 4.3: Percentage of students with no prior experience for each grade level.

4.1.3 Gender

As mentioned before, there has been a significant amount of research into how gender affects learning and performance in courses. We wanted to take up this question and investigate it in our residential course. As was mentioned before, all of the runs of the residential courses had a majority of females in them, which is greatly disproportionate compared to the percent of females at MIT as a whole, and even more so when compared to the percentage of females in the computer science department at MIT.

In addition to females being disproportionately represented in 6.0001 and 6.0002, they also tend to outperform their male counterparts. Being female is moderately positively correlated with overall performance in the course (0.13). Also, if we look at each of the 3 courses individually, being female is positively correlated in each run.

In order to find the causal reason for this, we look at the breakdown of who these female students are. Firstly, we already established that most computer science majors are males. Also, being as how these are introductory courses and are prerequisites for other computer science courses in the curriculum at MIT, most of the freshmen that take the class are aspiring computer science majors. Therefore, it would follow that a disproportionate amount of the freshmen taking the class are males. This is confirmed when looking at the gender breakdown of the freshmen. In the fall, for 6.0001, 54% of freshman students in the course were males. For 6.0002, 56% percent of the freshman students in the course were males. As was established in the previous section, being a freshmen seems to have a negative causal relationship with performance in the class, therefore the fact that females outperform males should not be attributed to gender, but to the fact that a higher percentage of females are older students.

4.1.4 Forum Participation

The residential course also utilizes an online forum. As explained before, this is a platform where students can ask questions and get answers from either staff members

or other students.

One thing to be noted is that the course switched forum platforms between the fall and the spring. This switch has a shocking result on student participation on the forum. For example, the average number of days that the students visited the forums dropped from about 41 in the fall to about 5 in the spring, and the average number of post views that a student had dropped from 62 in the fall to 38 in the spring.

Therefore, in order to find the effect that forum participation had on student performance, we once again looked at the data in subsets in order to isolate forum participation from other confounders.

Firstly, when we examine correlation of forum participation with performance for each run individually, we see the marked effects that the platform switch had. The summary of the degree of correlation is in table 4.4. Here we see that participation was much less strongly correlated with performance once the switch to the new forum was made, however forum participation still helped students in each course.

	Fall 6.0001	Fall 6.0002	Spring 6.0001
Forum Days	0.28	0.27	0.15
Forum Views	0.20	0.17	0.10

Table 4.4: How forum participation is related to performance for each course. Values here are Pearson correlation coefficients. This shows that the correlation between forum participation and performance was halved after the switch to the new forum in Spring 6.0001.

	Fall 6.0001	Fall 6.0002	Spring 6.0001
Forum Days	0.39	0.38	0.21
Forum Views	0.28	0.29	0.15

Table 4.5: How forum participation is related to performance for each course for students with no prior experience. Values here are Pearson correlation coefficients.

In addition, we hypothesized that forum participation, similarly to any form of extra help, would benefit those with no experience more so than those who are already experienced. The numbers in table 4.5 show that, for students with no prior experience, access to a responsive forum is a vital resource for learning how to program,

and increased participation in the forum leads to better performance.

4.1.5 Office Hours

One of the great advantages of the residential course over the MOOC is the availability of in person tutoring. Both 6.0001 and 6.0002 hold 50 hours of office hours each week, during which students can ask staff members questions about assignments or course material. However, to our surprise, we found that frequency of receiving help in office hours was strongly negatively correlated with overall grade (-0.19).

Once again, when trying to find causal relationships, we examined the feature in isolation. We hypothesized that, once again, students with no experience would benefit greatly from attending office hours. The reason for office hours attendance being strongly negatively correlated with performance is likely due to the fact that students who are struggling with the course material are more likely to be the ones attending office hours, and while they are better off attending office hours than not attending, their final grades are still going to be lower than those who have a mastery of the material and therefore do not need to go to office hours. The causality graph is displayed in figure 4-4.

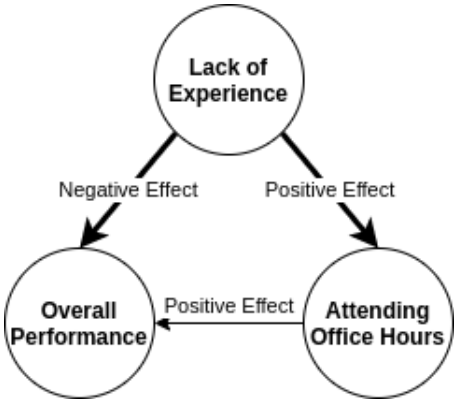

Figure 4-4: Causality Graph of office hours and its effect on overall performance. The graph shows that, while office hours has a positive causal effect on overall performance, lack of experience has a stronger negative effect on overall performance while also having a strong positive effect on office hours attendance.

When we take a subset of the data of just students with no prior experience, this hypothesis starts to emerge. In fall 6.0001, there is very weak correlation between

total amount of office hour visits and overall performance, however, when the course material starts to get more complex in 6.0002, the correlation coefficient of queue visits with performance jumps up to 0.25.

We also have data for which problem sets they were asking for help with whenever they go to office hours. When looking at students with no prior experience, we find that the more students go to office hours for a problem set, the better they tend to do on that problem set. See table 4.8 for the numbers. For comparison, when we look at students with prior experience, there is no (or negative is some cases) correlation between office hours for a problem set and performance on the problem set.

In addition, another factor that could explain away the aforementioned female superior performance is the fact that females attend office hours at a significantly higher rate than males. In general, females went to office hours an average of 23.65 times, while males went on average 15.49 times.

	Fall 6.0001	Fall 6.0002	Spring 6.0001
Problem Set 1	0.00	0.16	0.18
Problem Set 2	0.20	0.06	0.01
Problem Set 3	0.26	0.12	0.31
Problem Set 4	0.19	0.26	0.20
Problem Set 5	0.10	0.19	N/A

Table 4.6: The Pearson Correlation coefficients for each problem set and how many times the student went to office hours to ask for help for that problem set (checkoffs excluded). The dataset is limited to just students with no prior experience.

4.1.6 Optional Additional Practice

Finally, there are also optional exercises that the students can do for extra practice. These exercises are each associated with a lecture and test the material that was covered in that lecture.

Similarly to office hours attendance discussed above, we see that there is a negative correlation between doing these extra practice problems and overall performance in the class for 6.0001. Likely, this negative correlation is due to performance of extra practice problems being confounded by lack of experience. However, the optional ex-

		Fall 6.0001	Fall 6.0002
All	Days Active	-0.11	0.23
	Chapters Visited	0.02	0.17
	Unique problems attempted	-0.08	0.19
No Experience	Days Active	0.15	0.30
	Chapters Visited	0.26	0.31
	Unique problems attempted	0.15	0.29
Experienced	Days Active	-0.17	0.24
	Chapters Visited	-0.04	0.14
	Unique problems attempted	-0.10	0.16

Table 4.7: We see that extra practice was beneficial for inexperienced students in both classes, and it was beneficial to experienced students in the advanced course but not in the introductory course. The numbers here are the Pearson correlation between each feature and overall performance.

ercises data differs from the office hours data in the fact that it is positively correlated with overall performance for 6.0002. This interesting difference is likely explained by another principle that we discussed earlier in table 4.2. Prior experience (and lack thereof) is significantly less impactful in 6.0002 than in 6.0001. Therefore, there is less of a confounding effect causing the negative correlation that we see in 6.0001. Table 4.7 shows just how significant performing these extra exercises is for performance.

4.2 MOOC Analysis

There has already been significant research done on the MOOC courses, so we will not go into too much detail discussing all of the findings. However, we see that there also strong predictors of student performance in the MOOC versions. For example, as found in previous research [5] the number of days active on the website and the amount of course videos that the students watch (equivalent to lectures in the residential version) are strongly correlated with overall performance in the course. This trend can be seen in figure 4-5.

In addition to activity on the website, the second most important factor was participation in the online forum. While the correlation was not as strong as was found in the residential course, it is still a good indicator as to whether or not a

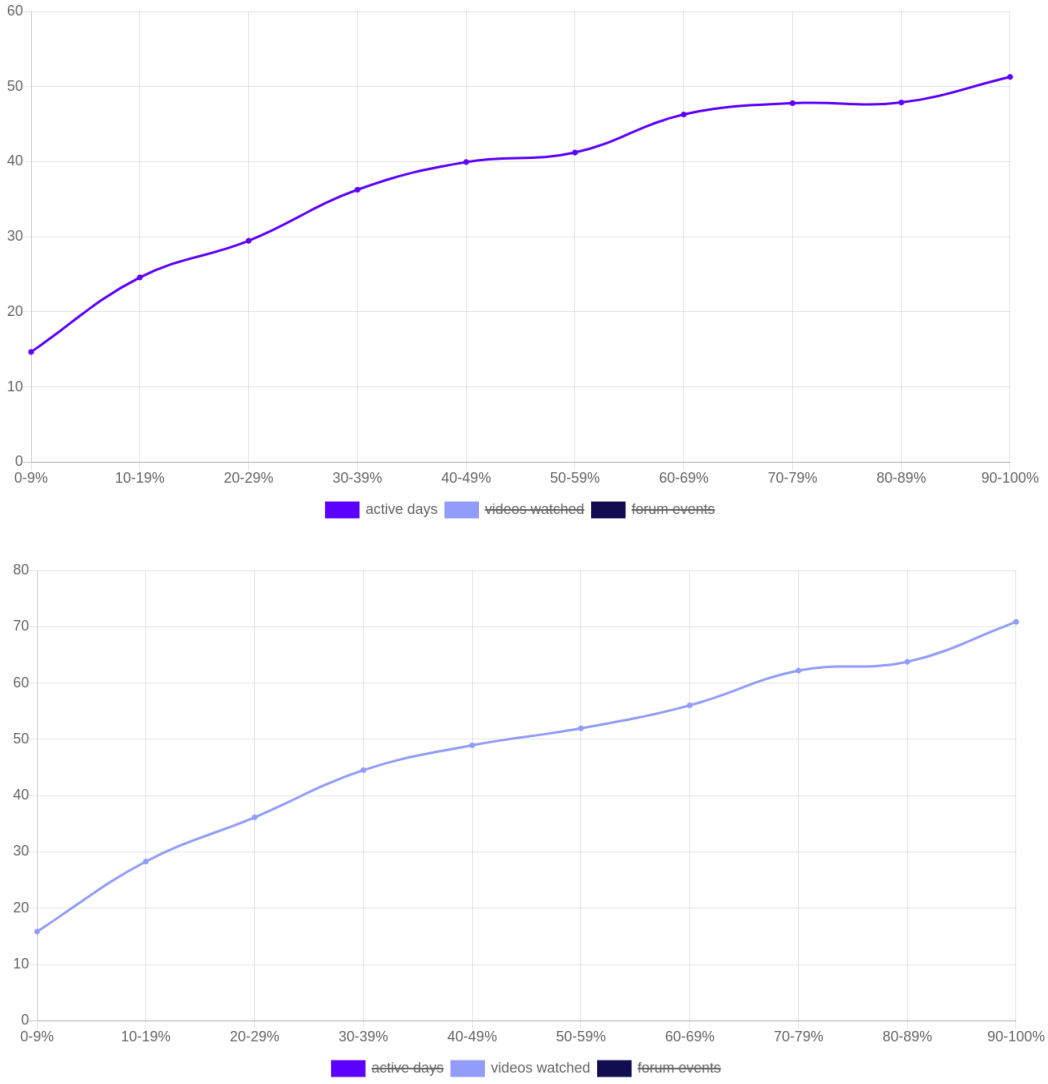

Figure 4-5: Plots showing the relation between how active students are on the website and how successful they are in the course. The graphic was generated using a tool built by [5].

student will complete the course.

One novel question that we asked about the MOOC data was whether or not the geographical region that a student came from had any impact on their performance. The number from the several thousands of students that took 6.00.1x indicate that, in general, the country that a student came from had little relationship with the overall grade that they earned in the class. However, we did find a relationship between students coming from economically developed regions, and those who came from undeveloped or developing regions. While we cannot make a firm conclusion about that, it is interesting and would merit further research.

4.2.1 Comparison

Now that we have looked at both the residential course and the MOOC version, we can compare our findings. The first finding was expected: students in the residential course do significantly better than in the MOOC course. Both overall grade and completion rate indicate this. However, we are more interested in the identification of shared features that can be used to predict student performance across platforms.

One of the first observations made by [5] was that problem set performance is linked to performance on exams. Surely the two features are confounded by other factors, however, there is good reason to believe that more effort put into problem sets results in better mastery of the material for the exam. Figure 4-6 shows the comparison between the relationship between problem sets and exams for the MOOC version and the residential version of both 6.0001 and 6.0002.

In addition to this, the number of videos watched are positively correlated in both courses. This shows that taking advantage of additional materials and practice is greatly beneficial to learning across platforms.

The forum also served as a very useful tool on the MOOC version of the class. It was more highly correlated with success than was the forum used for the residential version of the course. However, it was likely given more importance because of the absence of any other means of receiving help. Students that take the residential version not only have access to a forum, but they also have access to in person

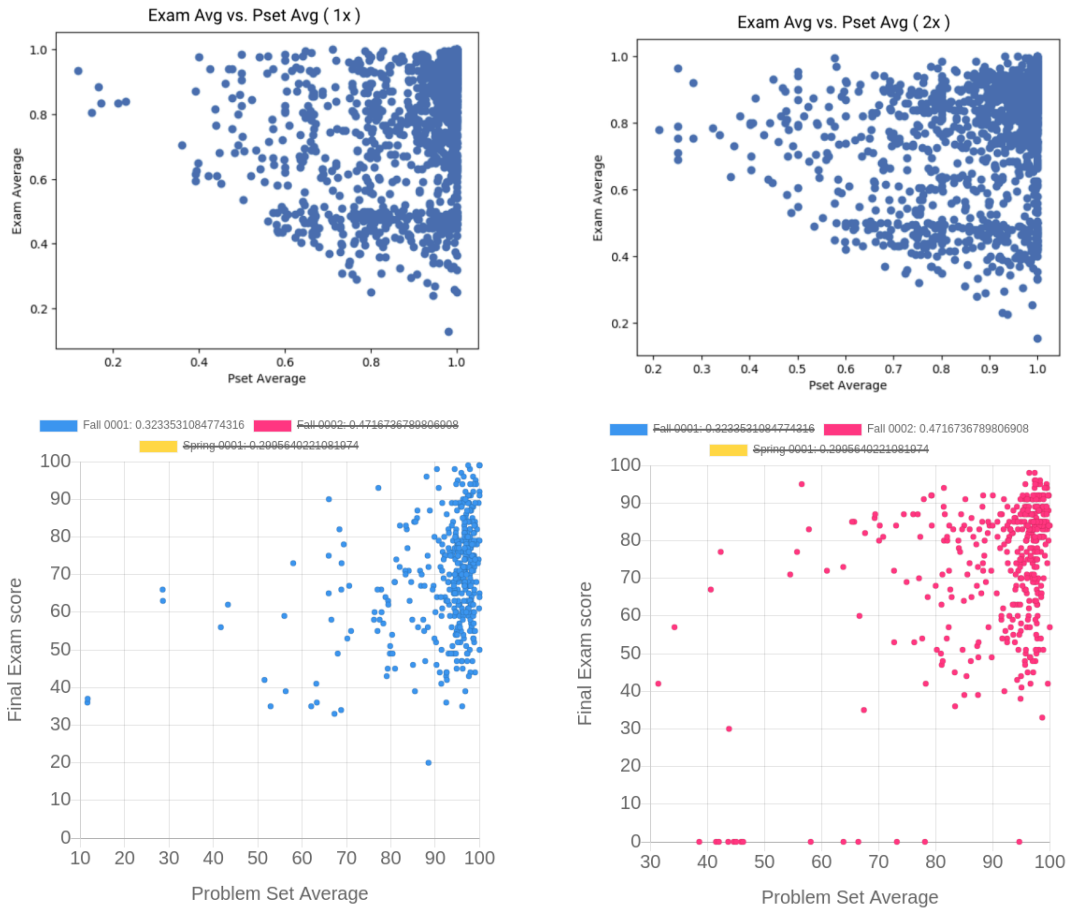

Figure 4-6: Plots showing the relation between how students perform on problem sets and how they perform on exams. Top Images taken from [5].

tutoring in office hours as well as peers that they take the course with. Therefore, the avenues of receiving help are more diversified in the residential versions while they are centralized in the forum for the MOOC version. This goes to show how important interpersonal assistance is in introductory programming courses.

4.2.2 Dropout Rate

Finally, the last thing to look at is dropout/completion rate. This is the biggest obstacle to overcome for online education to reach its potential. Students that intend to take online courses rarely ever follow through. For example, in 6.00.1x, of the 57,418 students that enrolled, only 35,874 ever viewed the course, and of that, only

1190 completed the course. 6.00.2x had a slightly higher attrition rate, with 3,656 of the original 7,692 students viewing the course, and 309 completing it.

	Fall 6.0001	Fall 6.0002	6.00.1x	6.00.2x
Enrolled	550	460	57,418	7,692
Viewed	N/A	N/A	35,874 (62%)	3,656 (48%)
Explored	N/A	N/A	4,286 (7.5%)	582 (7.6%)
Completed	407 (75%)	383 (83%)	1190 (2%)	309 (4%)

Table 4.8: The retention rate for students in 6.0001, 6.0002, 6.00.1x and 6.00.2x. Viewed and Explored metrics are not available for the residential courses. We see that the rate of completion is significantly higher in the residential courses as opposed to the online courses. This illustrates one of the main problems with online education.

Chapter 5

Predicting Performance

The final major contribution of this paper is exploring the predictability of student performance in a course given both their background information, and their behaviors in the course. In order to do this, we leverage the data that were previously discussed in order to train machine learning algorithms to be able to accurately predict how unseen students will perform in a course.

5.1 Motivation

The end goal for this aspect of the research is to create a feature on the 6.0001 and 6.0002 course website that will provide students with predictions for how well they will perform, and if they are in danger of failing the course. The predictive product on the website will be able to gather all of the data relevant to their profile, including their activity on the website, their grades thus far in the course, their background information, and all of the other features that were discussed in the previous chapter. Then, it will use machine learning algorithms to make predictions on how that student will perform in the course. In addition, simple gradients can be taken from the predictive algorithm to provide suggestions on how the student can increase their likelihood of success. This feature will be a possible solution to address many of the previously mentioned problems such as the problem with inexperienced students lagging behind their moderately experienced counterparts. By being able to

see how they are predicted to perform in the course and what behaviors will lead to better performance for people with their profile, students can make informed choices in how to tailor their study habits in order to maximize performance.

5.2 Algorithms

Given the low volume and high dimensionality of our dataset (about 750 data points each with 128 features), we decided to try out a few standard machine learning algorithms, as well as one multi layer perceptron in order to attempt to capture any non-linear relationships among the data.

We used three metrics to measure the performance of our models. Mean Squared Error takes the average error squared over all predictions: $\frac{1}{N}\sum l^2$. This metric disproportionately penalizes errors as they get worse. Root of Mean Squared Error provides a more interpretable way to understand model performance, while still increasing penalty as the prediction gets worse. Finally, Mean Absolute Error is the average of how far off each prediction is.

The baseline prediction that we are comparing against is simply predicting each student to earn the mean overall score in the course. This baseline's performance is:

- Mean Squared Error (MSE): 103.03
- Root of Mean Squared Error (RMSE): 9.70
- Mean Absolute Error (MAE): 8.03

5.2.1 SVM

Support Vector Machines (SVM) are a good choice for this problem due to the high dimensionality of the data, and the SVM's characteristic ability to separate data based on their dimensions. We tried 3 different kernels and ran a regression algorithm on the data with each.

Linear Kernel

SVR(kernel='linear', C=3, gamma='auto')

We first experimented with a linear kernel and different values for the regularizer. The regularizer determines how much the model is penalized for errors, with high regularization parameters penalizing the model more than lower ones. After testing out different parameters, we found a regularization parameter of 3 to work best. This makes sense because, due to the complexity of the data, in order to have an algorithm that will generalize to unseen data, there needs to be a relatively high leniency for error. A relatively low value for the regularization parameter permits a higher tolerance to error in margins.

- MSE: 42.04
- RMSE: 6.48
- MAE: 4.94

Poly Kernel

SVR(kernel='poly', C=100, gamma='auto', degree=3, epsilon=.1, coef0=2)

Secondly, we tried a polynomial kernel. The polynomial kernel adds a higher degree of complexity into the boundary decision, so therefore we were able to take advantage of a higher regularization parameter. The optimal C that we found was 100 with a polynomial degree of 3. This makes the penalty for margin violations higher, which allows for increased accuracy without overfitting because the polynomial nature of the kernel allows for more precise boundaries to be drawn.

- MSE: 38.13
- RMSE: 6.18
- MAE: 4.82

RBF Kernel

SVR(kernel='rbf', C=10, gamma=.1, epsilon=.1)

The RBF kernel performed slightly worse than the other two kernels. It still achieved a fairly high MSE and MAE compared to the other algorithms used, but among the SVM variants, it performed the worst.

- MSE: 44.21
- RMSE: 6.65
- MAE: 5.17

5.2.2 Regression

Regression algorithms also worked very well on this data. Particularly, they were effective at reducing the mean absolute error. However, given the large amount of noise inherent in human behavioral data, the mean squared error was slightly disproportionately high.

We found that regression models with L1 losses worked particularly well on the dataset on hand. LASSO and Ridge regression encourage finding simple models with L1 loss, and enable multicollinearity through creating sparse models. Since our data is comprised of many features all of which are either weakly or moderately correlated with its label, these forms of regression tended to work well.

	MSE	RMSE	MAE
Ridge Regression	40.13	6.33	4.96
LASSO Regression	40.15	6.33	4.82
Bayes Regression	41.13	6.41	5.00

Table 5.1: Here we see that each of the regression models that we used perform comparably, with Bayes Regression scoring a slightly higher mean squared error. However, this difference is within the margin of error.

5.2.3 Multi Layer Perceptron

MLPRegressor(hidden_layer_sizes=(20,10,5), activation='relu', solver='adam',
alpha=0.001, batch_size='auto', learning_rate='constant', learning_rate_init=0.05,
power_t=0.5, max_iter=5000, shuffle=True, random_state=0, tol=0.01,
verbose=False, warm_start=False, momentum=0.9, nesterovs_momentum=True,
early_stopping=False, validation_fraction=0.1, beta_1=0.9, beta_2=0.999,
epsilon=1e-08)

Finally, we also tried to fit a multi layer perceptron to the data. While there was not a large amount of data such that neural networks or deep learning would be an obvious choice, we did wish to attempt to find any sort of nonlinear relations among the data. Multi layer perceptrons allow for just that. With their hidden layers and connectedness, they allow for the approximation of any nonlinear function. The following are the results that we got using the MLP.

- MSE: 46.62
- RMSE: 6.83
- MAE: 5.27

5.3 Summary of Predictions

As we can see in figure 5-1, training machine learning algorithms from the data collected through the course website that we built leads to very accurate predictions as to how a student will perform. The errors of the machine learning algorithms are approximately half the mean absolute error of the baseline. This shows that the models are capable of learning features that have causal relationships to the overall grade that a student earns in a class. In addition, the mean squared errors are generally less than half of the mean squared error of the baseline model. This indicates that our models will rarely incur large errors, and therefore it is less likely

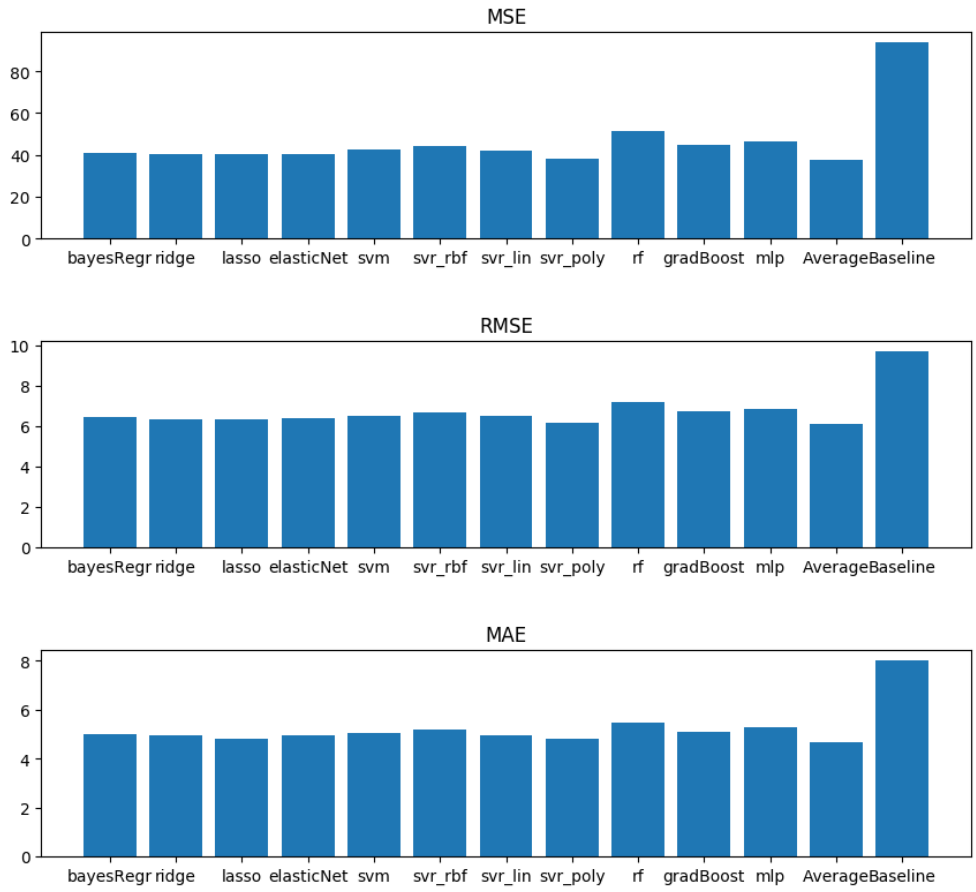

Figure 5-1: These bar graphs show the accuracy of each of the classifiers on the dataset. We are measuring mean squared error, root of mean squared error, and mean absolute error. As we can see, each of the classifiers performed significantly better than the baseline classifier. When we take the predictions of each of the separate algorithms and perform an equal weighted voting to calculate the average prediction, we get the highest accuracy in all three categories.

that any prediction given to a student will be incorrect by more than the mean average error of approximately 5 percentage points.

Naturally, as more assignments are completed and are fed into the models, and as more runs of the course are performed, these models will become more accurate, and thus more useful. The results of our training and testing show that student behavior in a course is predictive of their final results.

Chapter 6

Conclusion

6.1 Further Work

There are two key areas where this research could be expanded upon in further work. The first area is completing the performance prediction product on the 6.0001 and 6.0002 residential course website. There is still some work that would need to be completed in order to pipe all of the data into the necessary location so that student profiles can be constructed, and then run through the predictive algorithm in order to estimate how they will perform. In addition, more research could be conducted on how best to interpret the machine learning algorithms' predictions in order to provide tailored suggestions to the students as to how best increase their predicted performance in the course. This would be a greatly helpful product that would easily follow from this work. Given my involvement and progress already made on completing such a product for 6.0001 and 6.0002 at MIT, I intend to continue my work on this and see it to completion.

Additionally, another interesting field is optimizing models in order to predict performance. With all of the analysis that we have performed throughout this project, we now know the features that are most indicative of student performance. Further work could be focused on expanding and refining the datasets, and possibly incorporating the 6.00.1x and 6.00.2x data into the models.

6.2 Final Thoughts

With the enormous value of education in the world today, it is important to be able to better understand what factors are leading students to succeed, and what factors are causing others to fail. In our research, we were able to contribute towards answering that question. We were able to collect rich data on student behavior and performance in two university courses, and use it to compare to an online version of those courses. We found that the experience students have when entering a class is extremely influential on the types of behaviors that lead to success.

Another major contribution of our work was the finding that student performance is predictable. Not only that, but we also took that a step further to implement predictive algorithms that can be used to analyze the data that we collect, and garner insights from it in order to assist students in their education.

We believe that the most powerful technologies should be leveraged to help solve the most important problems. With education being of great importance, this research is a step towards applying the newest methods in order to assist in solving novel problems.

Appendix A

Tables

Features	Fall 6.0001	Fall 6.0002	Spring 6.0001
exp_None	-0.382452	-0.133227	-0.221909
nvideos_total_watched	-0.272425	0.079718	-0.101661
nvideos_unique_viewed	-0.272425	0.079718	-0.101661
sum_dt	-0.256963	0.150411	-0.132827
nseek_video	-0.256699	0.071850	-0.047127
nvideo	-0.255770	0.075118	-0.107758
nplay_video	-0.255770	0.075118	-0.107758
language_nevents	-0.253657	0.093534	-0.066734
npause_video	-0.252047	0.056163	-0.125422
nevents	-0.237254	0.161427	-0.079022
n_dt	-0.235992	0.162318	-0.080248
nproblem_check	-0.205337	0.147690	-0.067317
cheat	-0.178394	-0.142978	-0.088969
l2_total_attempts	-0.151036	0.043481	-0.153077
queue_all	-0.147974	-0.150230	-0.033940
queue_help	-0.140940	-0.101377	-0.236627
queue_count	-0.135409	-0.050297	-0.001299
avg_attempts_per_problem	-0.129287	-0.049696	-0.142106
dep_Undeclared	-0.128292	-0.059254	0.212509
avg_submits	-0.126835	0.213466	0.152906
l1_total_attempts	-0.126135	0.068845	-0.112030
queue_add	-0.125543	-0.042717	-0.000160

l2_unique_problems_attempted	-0.125542	0.094836	-0.124172
nshow_answer	-0.121271	0.136256	-0.057754
queue_pset3	-0.116199	-0.086334	-0.008420
ndays_act	-0.113006	0.231363	0.131170
queue_pset4	-0.112583	-0.136244	-0.127089
queue_pset1	-0.108921	-0.063172	-0.110673
mot_learn	-0.107853	0.006898	-0.111061
nseq_goto	-0.104721	0.207525	-0.075668
l4_total_attempts	-0.101146	0.192420	-0.101385
l5_unique_problems_attempted	-0.095255	0.158921	-0.105897
l1_unique_problems_attempted	-0.094723	0.086639	-0.083018
total_attempts	-0.094485	0.155038	-0.090218
dep_Employee	-0.093725	N/A	N/A
Harvard	-0.084919	N/A	-0.150183
unique_problems_attempted	-0.083227	0.186772	-0.085378
l1_number_of_correct_problems	-0.083136	0.116795	-0.087116
l5_total_attempts	-0.082845	0.155237	-0.086394
dep_Economics	-0.080466	-0.059986	0.030995
l7_unique_problems_attempted	-0.076714	0.162012	-0.024665
l4_unique_problems_attempted	-0.075943	0.224179	-0.107976
l6_unique_problems_attempted	-0.074362	0.117969	0.018518
l6_total_attempts	-0.070656	0.096395	0.002844
dep_BrainAndCognitiveSciences	-0.062671	0.009136	-0.039410
l7_total_attempts	-0.062135	0.135391	-0.008977
dep_HealthSciencesTechnology	-0.060745	N/A	0.029883
queue_pset5	-0.060710	-0.134425	-0.030379
l2_number_of_correct_problems	-0.058707	0.074400	-0.106677
queue_pset2	-0.057865	-0.019426	-0.272775
dep_Unknown	-0.054517	-0.037252	-0.220187
dep_Biology	-0.051778	0.033339	-0.019572
activity_count	-0.039769	0.088502	0.046033
l3_total_attempts	-0.038894	0.077677	-0.152362
l3_unique_problems_attempted	-0.038474	0.097618	-0.120791
dep_Architecture	-0.033336	N/A	0.047119

dep_AeronauticsAndAstronautics	-0.022248	-0.104248	0.033260
max_dt	-0.019779	0.168617	0.004448
l6_number_of_correct_problems	-0.019745	0.124820	0.047936
dep_Undesignated	-0.014941	N/A	-0.081724
dep_MaterialsScienceAndEng.	-0.014694	N/A	-0.034463
dep_Management	-0.014055	-0.018061	-0.055160
click_count	-0.011863	-0.027697	0.013079
forum_notes	-0.007648	-0.134233	N/A
l7_number_of_correct_problems	-0.007357	0.216833	0.025799
number_of_correct_problems	-0.007101	0.214031	-0.053576
Wellesley	0.007961	-0.037252	-0.183741
l8_unique_problems_attempted	0.009372	0.144373	-0.026206
dep_EarthAtmosPlanetarySci	0.013051	-0.031806	0.017652
l8_total_attempts	0.013633	0.134336	-0.024958
avg_dt	0.017450	0.134227	-0.102459
MIT	0.020812	0.037252	0.220187
dep_ChemicalEngineering	0.021213	0.069680	0.065490
l3_number_of_correct_problems	0.022198	0.075431	-0.113244
nchapters	0.027954	0.166115	0.230152
l910_unique_problems_attempted	0.033752	0.072392	0.017347
dep_BiologicalEngineering	0.037053	0.022340	0.034495
dep_department	0.041042	N/A	N/A
dep_HumanitiesEngineering	0.056310	-0.190171	N/A
sdv_dt	0.061437	0.027289	-0.057327
l910_total_attempts	0.062014	0.058502	0.012288
forum_questions	0.063351	0.020771	0.071496
forum_endorsements	0.063418	0.040312	0.005709
dep_CivilAndEnvironmentalEng	0.066867	N/A	0.025435
dep_PoliticalScience	0.070306	0.051094	N/A
forum_answers	0.070586	0.067109	0.088886
l11_total_attempts	0.074848	N/A	0.011495
dep_ElectricalEngComputerSci	0.075758	0.039516	0.106183
l8_number_of_correct_problems	0.076499	0.157730	-0.024965
l910_number_of_correct_problems	0.076761	0.096208	0.044120

dep_Physics	0.077304	N/A	-0.034379
exp_ocw	0.077405	0.025714	0.085034
l11_unique_problems_attempted	0.084750	N/A	0.040624
forum_contributions	0.086489	0.038398	0.056794
dep_Mathematics	0.088344	0.117216	-0.119021
awr_google	0.090036	0.050357	0.072221
exp_online_class	0.098468	0.054456	0.109369
mot_gpa	0.100887	0.049492	0.057279
dep_Chemistry	0.101410	0.020717	-0.026015
l11_number_of_correct_problems	0.103703	N/A	0.033157
awr_online	0.113078	0.039954	0.089415
grade_level	0.115263	0.136729	-0.097794
dep_MechanicalEngineering	0.116072	0.096296	0.028376
f	0.126301	0.092399	0.081262
dep_UrbanStudiesAndPlanning	0.127383	0.110228	0.011676
exp_python_college	0.138716	-0.038057	0.081132
exp_html	0.150942	0.158530	0.096540
awr_friends	0.173711	0.077060	0.004936
exp_python	0.185047	-0.013425	0.091198
forum_views	0.201824	0.165548	0.112755
exp_non_python_college	0.203478	0.142133	0.023643
mot_requirement	0.213747	0.019025	0.192639
exp_ap_comp_sci	0.249212	0.028552	0.247589
forum_days	0.273414	0.271149	0.158982
exp_non_python	0.302410	0.150321	0.135224
FE	0.319524	0.419502	0.436453
awr_stack_overflow	0.352749	0.122050	0.110001
PS1	0.359603	0.484581	0.352341
avg_workahead_time	0.417022	0.446394	0.394479
avg_pset_time_remaining_after_last_submit	0.444426	0.431610	0.418251
lines_of_code	0.481576	0.210433	0.292615

Table A.1: Displaying the correlation of each feature with overall grade for each residential run separately.

Appendix B

Course Website Documentation

B.1 Running Locally

Installation of the underlying catsoop software can be found at
https://catsoop.org/website/docs/installing.
Use the git clone option for now. Once you have catsoop installed and set up, clone the 6.0001/2 website into your computer at the following location:
‘ /.local/share/catsoop/courses/‘
The command to clone is: ‘git clone catsoop@sicp-s1.mit.edu:plgrm/repos/<course name>‘ Now you can run the course locally! Just run the command: ‘catsoop start‘ And navigate to localhost:6010 in your browser.

B.2 Install python packages on server

The goal is to get the package into /python/pycs/lib/python3.7/site-packages
$ *source catsoop/python/sandbox/bin/activate*
$ *pipinstall < package >*
$ *deactivate*
Then cd into ‘ /python/sandbox/lib/python3.7/site-packages‘ and cp the packages into pycs

B.3 General CAT-SOOP information

CAT-SOOP is a course website building software that maintains databases, security, server side functionalities, and compilation of the code. It is a hierarchical structure where each layer contains at least two files: content.md and preload.py

- content.md is a markdown file that renders the page for the student. In addition to markdown, python can be used. You can either include and run python code in between <python> tags (analogous to javascript tags in regular web development) or outside of tags by surrounding with @{}. In addition, anything printed in python is displayed on the web page. Another important feature are API calls. That is how I call various python files on the server side to perform more involved functions.

- preload.py is run before the page is loaded. All variables and functions can be leveraged in the content.md file. All variables defined in a preload.py file are available for use in any file at the same level as or below it in the hierarchy.

There are various special variables. The following is a non-exhaustive list:

- cs_username: returns the name of the user.

- cs_user_info: accesses all of the variables in the user's profile in __USERS__.

- cs_view_without_auth: enables viewing of a page without being authenticated.

- cs_login_box: the code for the login box display

- cs_top_menu: the menu on the top of the page. Look for it in the top preload.py file for more details.

- cs_base_color: the color theme for the website.

- cs_content_header: the header on the top of that page.

- cs_post_load: function that is ran after the page is loaded. Every page runs it and it is inherited from higher up preload.py files unless it is redefined in the current preload.py file. It is in this file that I am recording the activity log data.
- cs_pre_load: function that is ran before the page is loaded
- cs_content: the content that is displayed on the web page. One of my strategies was to check a user's role once a page was loaded, and if they did not meet the necessary permissions to view that page, simply set cs_content to be "You are not allowed to view this page", this hiding it from the unauthorized user.

B.4 Website File Structure

Below is the file structure for the website repository. Assume that all elements without an extension are directories. Assume that each directory, except for the directories surrounded by underscores, contains an implicit preload.py and a content.md.

Disclaimer: this is a lot of information to digest, but I include this because it contains important information for future TA's who will be taking over the course website from me. It is a large code repository, so some guidance will be needed.

```
<Course Name>
├── __PLUGINS__
├── __QTYPES__
│   ├── _pythoncode_600
│   ├── 600zip
│   ├── 60001_early_lab
│   ├── lab2
│   ├── lab5
│   ├── lab009_weighted
│   ├── multi_file
│   └── pythoncode_600
├── __STATIC__
├── __USERS__
├── additional_resources
└── big-brother
    ├── __STATIC__
    └── _files
```

```
│
│   ├── get_activity_log_data.py
│   ├── get_all_checkoffs.py
│   ├── get_all_final.py
│   ├── get_queue_info_student.py
│   ├── get_sus_checkoffs.py
│   ├── give_extension.py
│   ├── give_MQ_extension.py
│   ├── MOSS_MQ_files.py
│   ├── MOSS.py
│   └── restart_MQ.py
├── exams
│   ├── _files .2 get_image
│   └── content.py
├── grade_checkoff
│   └── submit.py
├── grades
│   ├── generate_MQ_sheet.py
│   ├── get_all.py
│   ├── get_user_checkoff.py
│   ├── get_user.py
│   ├── log.py
│   ├── manual_quiz.py
│   └── manual.py
├── information
├── lab_downloader
│   └── content.py .2 lab_viewer
├── MQ0
├── MQ1
├── MQ1_2
│   └── startQuiz.py
├── MQ2
├── MQ2_2
│   └── startQuiz.py
├── MQ3
├── MQ3_2
│   └── startQuiz.py
└── profile .2 psets
    ├── 1_pset0
    │   └── _files
    │       └── test.py
    ├── 1_pset1
    │   └── _files
    │       └── test.py
    ├── 1_pset2
    │   └── _files
```

```
        └── test.py
    ├── 1_pset3
    │   └── _files
    │       └── test.py
    ├── 1_pset4
    │   └── _files
    │       └── test.py
    ├── 1_pset5
    │   └── _files
    │       └── test.py
    ├── 2_pset1
    │   └── _files
    │       └── test.py
    ├── 2_pset2
    │   └── _files
    │       └── test.py
    ├── 2_pset3
    │   └── _files
    │       └── test.py
    ├── 2_pset4
    │   └── _files
    │       └── test.py
    └── 2_pset5
        └── _files
            └── test.py
├── queue
├── student_picture
├── styleguide
├── user_scripts
├── viz
│   ├── _files
│   ├── compile_activity_log_data.py
│   ├── compile_click_log_data.py
│   └── compile_queue_log_data.py
├── contend.md
├── preload.py
└── log_click.py
```

Okay, now lets walk through what these pieces do...

__PLUGINS__:

This directory contains much of the backend required for the queue. For instance, you may be able to customize some of what is passed to the server and the socket via

the handlers that are contained in this directory. In general, however, I spent very little time messing around in here. Adam Hartz would know this part of CAT-SOOP better than I would.

__QTYPES__:

Here is where the workhorses of all of the on website questions are located. Each directory contains a python file (with the same name as the directory) that handles submissions, checks, and all other actions that a student can take with a question. In addition, it determines what data is recorder in the database for each submission, including the score.

- _pythoncode_600: This question type was custom made for exams to be administered on the course website. It incorporates much more flexibility and reliability than the question type that came with the website. This is all code that I wrote myself, combining the good parts of several other question types. It grades code from that student that is inputted in a code box. Special variables to remember:
  - csq_name: The name of the problem. Important for locating score for that problem in the database.
  - csq_allow_viewanswer: Boolean. Determines if the students have the option to reveal the answer to the question or not. Recommended to be set to False.
  - csq_allow_check: Same as above.
  - csq_allow_save: Same as above.
  - csq_prompt: String. the test prompt that the student is presented with. Instuctions for the specific question should be here.
  - csq_initial: String. The initial code that populates the text box that the student is given to fill in their code. Any skeleton code goes here.

- csq_check_function: a function. Takes in student submission and solution from staff code. Compares them, and returns Boolean.
- csq_soln: String. The solution code that is called to generate the correct answer to check student code against in csq_check_function.
- csq_score_message: Function. Parameter is the student's score. Returns whatever will be displayed to the student after they submit. Can be nothing (as we do on psets) or can be their score, or can be a message if they score below a certain point or nothing otherwise...
- csq_tests: List of Dictionaries. Each element of the list is a dict that represents a single test case. Keys of the dictionary include:
  - code: A string of python code that will be run to generate the answers.
  - hidden: Boolean. indicates to the qtype whether this test case is hidden or public (i.e. viewable to the student after they submit.)
- csq_timeout: timeout for the test cases, in seconds.

- 600zip: Question type that was built for problem set 2 in 6.0001 where the students had to submit multiple files. Allows for submit of .zip file, unpacks .zip file, looks for a file called 'ps2' and another file called 'graph' in the .zip file, and auto grades these two files. I ended up not using this anywhere, because I opted to use multi_file instead, but it is pretty cool still.

- 60001_early_lab: This is a specially built question type that is made just for the first problem set of 6.0001. The first problem set does not use functions, so, in order to auto grade the student code, I made this special qtype to search through the text that the student submits, and to cut and paste their code into functions and procede to test their code in the newly built functions.

- lab2: This is a specially built question type for problem set 2 in 6.0001. This problem set tests the student code based upon the console output of the student code, not a function's return value. This qtype is generally the same as the other

problem set qtypes, however, the TEST_CODE that drives the testing in the sandbox environment is tailored to record and check the stdout.

- lab5: this is a personal favorite of mine. It does not do anything fancy or complicated. It runs a simple unittest suite to check the student functions briefly. This problem set does not have an autograder component, so it does not matter much. However, the neat part is that this qtype produces an image with the student's code, saves it in the database, retrieves the image from the database, and presents it to the student as their feedback for submitting the code so that they can see what their code produces! This was a special request from Ana, and I was happy to oblige.

- lab009_weighted: this was a qtype that was created by the course administrators for 6.009. However, there were some bugs in the code that I had to fix in order to use it with our system. In general, I use this infrastructure for many of the other qtypes. This qtype creates a sandbox directory, saves the student code in one file in the directory, and loads a test file that is stored in a subdirectory called "_files" that is in the directory that houses the actual question that the student is submitting (in this case, the problem set directory), and saves that as another file in the directory. Then it makes a proc call to run the test file, which is a unittest suite. The unittest suite will return an output detailing which unittests were passed and which ones were failed. This output, gathered in lab009_weighted, is then used to calculate the score of the student code for that test suite.

- multi_file: Warning - the first 30 lines of 'handle_submission' are hardcoded to work with 6.0002 problem set 2! This qtype handles the auto grading of a problem set with multiple file submissions. This could be fixed if you make it necessary for the programmers of the website to input file names, and other names that are currently hardcoded.

- pythoncode_600: this is the built in pythoncode question type, but slightly

modified to allow for hidden test cases. This, just like pythoncode, should not be used. It is not robust, and does not allow for imports! These failures were the reason that I built _pythoncode_600.

__STATIC__:

This directory is where all static files are stored and can be accessed easily in the content.md files. See the official CAT-SOOP documentation for more on this.

__USERS__:

This directory contains one python file for each student. The file must be names <kerberos>.py. Only once a student has a file created for them here can they log in and access materials and submit assignments. There are several special variables that these files contain (more can be added and accessed using cs_user_info.get(<variable name>, None) anywhere in a preload.py function or content.md file)

- role: the role of the user. can be: Guest, Student, LA, TA, Instructor, or Admin.
- name: The name of the user.
- email: the email of the student.
- pset<number>_<1 or 2 for the course>_ext: number of days of extension given for that pset for that student. There is an interface on the big-brother page which allows you to write to that variable automatically.
- pset<number>_<1 or 2 for the course>_score: Overwrites the student's score for that particular problem set.

big-brother:

Buckle up... Here is where a lot of stuff happens. This is a special dashboard that the majority of the staff members do not know about nor use. Here, staff can add

students to the class, give extensions on psets, give extensions on micro quizzes, download all the pset files in the MOSS format for collaboration violation checks, download MQ code in MOSS format for collaboration violations, restart MQ timer for certain students, get activity log data for either pages or students or user roles (see if students are trying to access a forbidden page), get suspicious checkoffs to keep an eye on LA's, and generate final overall grades for the grade meeting. Each of these functions is its own python file in the big-brother directory.

exams:

This is the location where I have been posting student's final grades in the course, as well as their micro quiz grades.

get_image:

Contains a python file that retrieves a an image that is saves in the database. Used for pset 5 of 6.0001.

grade_checkoff:

This is the page that LAs and TAs interface with when giving checkoffs to the students. It takes care of what the staff members see and what they input to the databases.

grades:

This is another staff only page. This is where you can see the grade book for the class, as well as the grades for individual students. Not only can you see grades, but you can also see each of their submissions, including the history of submissions for each of their assignments. Once again, the directory contains many python files that perform all of the work.

lab_downloader:

Helper backend code that retrieves a student's code submission from the database and returns it so that it can be downloaded.

lab_viewer:

A page that is used to view a student's code. Instead of downloading their file it redirects you to another page where their code can be viewed.

psets:

this is where all of the problem sets are stored. Each problem set is its own directory. For Any problem set that is using a unittest for the auto grader, all of the files associated with the tester must be places within the "_files" subdirectory for that specific problem set.

queue:

This is part of the queue infrastructure. This was not a part that I made.

viz

Contains the code for all of the visualizations. Used in large part for this research.

B.5 Database

The databases for the website are taken care of, in large part, by the CAT-SOOP software itself. CAT-SOOP takes care of recording student submissions in the appropriate place, as well as maintaining logs for each problem for each student.

The database root for the course website is located on the server at `~/.local/share/catsoop/`. This is a very important location, because this is where much of the course information is stored and retrieved from.

The database directory tree is as follows:

```
cs_data_root
├── courses
│   └── <course name> - Contains copy of all courses in system
├── _logs
│   ├── activitylog-<course-name>.txt
│   ├── clicklog-<course-name>.txt
│   ├── MQ<#>_extensions.csv
│   ├── _courses
│   │   └── <course-name>
│   │       └── <One Directory for each student kerberos>
│   │           └── <File path:  path to page from course home>
│   │               ├── problemactions.log
│   │               └── problemstate.log
│   └── _uploads
│       └── <course-name>
│           └── <File path:  path to page from course home>
│               └── <One dir for each submission:  student association is contained
│                   in student's problemstate.log>
│                   ├── content
│                   └── info
└── MOSS
```

courses

Copies of the courses are located here. When running locally, this is the directory that you will place the course repository in.

_logs

This is the directory containing all of the database logs. This includes information about each student's answer to each problem and other information about their submissions and activities.

- activitylog-<course-name>.txt: This is a text file that contains the info of every page load on the website. Each datapoint includes: the time, the kerberos of the student accessing the page, which page it is, and the student role.

- clicklog-<course-name>.txt: This is a text file that contains the info of every

click on the website. Each datapoint includes: the time, the kerberos of the student, which button it is, and the student role. A `log\_click()` event listener must be places on objects that you wish to listen to.

- MQ<#>_extensions.csv: A file containing the data for extensions on the microquiz. Each line of data is of the format: `student kerberos,minutes,staff kerberos`.
- _courses: This is the directory where all the log data is stored. For example, the directory in which a student's log data about a pset is located at: `cs\_data\_root/\_logs/\_courses/<course-name>/<kerberos>/<psets>/<pset name>`
- problemstate.log: A dictionary containing the grade for the student's submissions, the tester response, the path to the file in _uploads, the time stamp of the submission, and other information. Retrievable using csm_log.py functions that are a part of the CAT-SOOP package. I recommend finding that file in the catsoop directory and checking out those functions.
- problemactions.log: All of the actions taken by the student towards that problem. This includes a history of every submit and details associated with it.

_uploads

This is a directory containing all of the uploaded files that the students submit for all of their problems. Refer to the directory tree above for file paths. Location of a particular student's upload can be found in their problemstate.log or problemactions.log file.

B.6 Micro Quizzes

1. Location: Each quiz is located in its own directory: COURSE/MQ<#>
2. File Structure: Simple file structure: just a content.md file and preload.py file
3. Timing:

(a) The release date of the MQ is stored in a variable in the root preload.py file. It is named ‘cs_MQ<#>_release‘ i.e. ‘cs_MQ1_release‘ for MQ1. This variable is used to set the built in ‘cs_release_date‘ in ‘MQ<#>/preload.py‘

(b) Due time is dependent on release time. In MQ1/preload.py, I take the release time and add 25 minutes to it, to get the due time. Due time is put in built in variable: ‘cs_due_date‘

(c) ‘cs_auto_lock‘ is on - so the questions will lock for students as soon as the due date is reached.

4. Extensions: Extensions were tricky to implement, but I decided on doing it this way:

   (a) All extensions are stores in a comma separated csv file, located at: ‘cs_data_root/_logs/MQ<#>_extensions.csv‘
   Entries are lines in the style of <kerberos>,<minutes>,<TA who gave ext>

Bibliography

[1] MIT admissions department. First-year class profile. 2020.

[2] Efthimia Aivaloglou and Felienne Hermans. Early programming education and career orientation: The effects of gender, self-efficacy, motivation and stereotypes. pages 679–685, 02 2019.

[3] Ayesha R. Bajwa. Analyzing student learning trajectories in an introductory programming mooc. 2019.

[4] POLYZOU A. REN Z. SWEENEY M. KARYPIS G. ELBADRAWY, A. and H. RANGWALA. Predicting student performance using personalized analytics. *Computer*, 2016.

[5] Christine Vonder Haar. Understanding learner engagement and the effect of course structure in massive open online courses. 2020.

[6] Adam Hartz. Cat-soop: A tool for automatic collection and assessment of homework exercises. *MIT*, 2012.

[7] Jitesh Maiyuran. Understanding the doer effect for computational subjects with moocs. 2018.

[8] KALINA YACEF RYAN S.J.D. BAKER. The state of educational data mining in 2009: A review and future visions. *Journal of Educational Data Mining*, 2009.

[9] DRUMOND L. HORVATH T. NANOPOULOS A. THAI-NGHE, N. and L. SCHMIDT-THIEME. Matrix and tensor factorization for predicting student performance. *CSEDU*, 2011.

[10] Li Wang. The influence of grades on learning behavior of students in moocs. 2019.

[11] Adnan Noor Mian Zafar Iqbal, Junaid Qadir and Faisal Kamiran. Machine learning based student grade prediction: A case study. *arXiv:1708.08744v1*, 2017.