Integrating Grade Prediction for Better Student Support in MIT's Introductory Programming Course

by

Alenta Demissew

B.S. Computer Science and Engineering
Massachusetts Institute of Technology, 2020

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

June 2021

Author .....
Department of Electrical Engineering and Computer Science
June 4th, 2021

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

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

Integrating Grade Prediction for Better Student Support in MIT's Introductory Programming Course

by

Alenta Demissew

Submitted to the Department of Electrical Engineering and Computer Science
on June 4th, 2021, in partial fulfillment of the
requirements for the degree of
Master of Engineering in Electrical Engineering and Computer Science

Abstract

We focus on grade prediction in the context of the 6.0001/2 course by utilizing student data – including assignment, assessment, and participation scores/metrics and click data – from past iterations of the course. In doing so, we explore various machine learning algorithms to create expressive, accurate predictive models. We have created and integrated the predictive modeling tool into the current course site to allow course staff to monitor student grade trajectories while guiding and assisting struggling students. Staff are able to interface with this tool which allows them to see this grade prediction along with other useful attributes for any student enrolled in the course. Students, although not directly able to interface with the tool, can be alerted and offered assistance if their grade trajectory is predicted to be a failing grade or below a certain threshold in order to provide them with the necessary resources to succeed in the course. Finally, we analyze the grade predictions and compare them to the final grade outcomes to find trends and patterns with regards to how students with different trajectories through the semester adjust their behaviors in the course based on the marks they receive.

Thesis Supervisor: Ana Bell
Title: EECS Principal Lecturer

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background

Nowadays, socioeconomic advancement and even securing a job requires more education than ever before (see Fig 1.1) [3]. Meanwhile, the shift towards a more digital world has led to the automation of many jobs and caused a surge in demand for computer based skills in the workforce. More and more jobs, projects, and problems require familiarity, experience, and an understanding of computer science fundamentals like programming, computational thinking, optimization, etc regardless of the industry [5].

Table 1.1: Departments Requiring Students to Take 6.0001/2

Department	**6.0001**	**6.0002**
Course 3: Materials Science and Engineering	Optional*	Optional*
Course 6: Electrical Engineering and Computer Science	Required	Required
Course 9: Brain and Cognitive Sciences	Required	Required
Course 15-2: Business Analytics	Required	Required
Course 16: Aeronautics and Astronautics	Required	Required
Course 18-C: Mathematics with Computer Science	Required	Not Required
Course 20: Biological Engineering	Required	Required
Course 22: Nuclear Science and Engineering	Optional*	Optional*

* Students must either take both 6.0001 and 6.0002, or one of three other courses

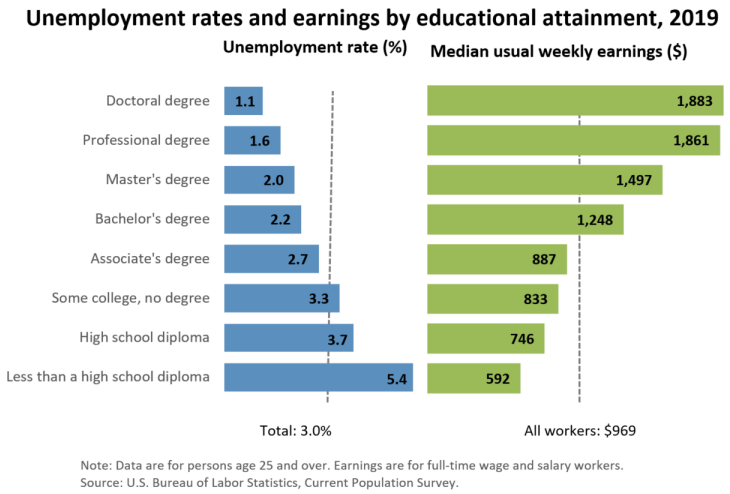


Figure 1-1: Unemployment Rate (blue) and Median Weekly Earnings (green) by Education Attainment based on Labor Force Statistics from the Current Population Survey in 2019. Data is only includes workers over 25 and earnings are for full-time wage and salary workers. [1]

At the Massachusetts Institute of Technology, this has led to many departments, not just the Electrical Engineering and Computer Science (EECS) department, requiring their students to take the introductory programming course(s) 6.0001 Introduction to Computer Science and Programming in Python and 6.0002 Introduction to Computational Thinking and Data Science. Currently, eight departments have 6.0001/2 as a graduation requirement or graduation requirement option for students - See Table 1.1. In addition, during the 2020-2021 academic year 6.0001 and 6.0002 were each listed as a prerequisite or co-requisite for 25 unique courses and there are three courses that list both 6.0001 and 6.0002 as prerequisites or co-requisites [2]. Student enrollment numbers in these two courses is comparable to that of some General Institute Requirements (GIRs) - See Fig 1.1 - with students having a large range of experience levels with the material and includes students at every grade level from freshman who have no programming experience to seniors who've completed programming related courses, projects, and/or internships. The course is not only

popular among MIT undergraduates, but also MIT graduate students from various departments and cross-registered undergraduate students from Harvard University and Wellesley College.

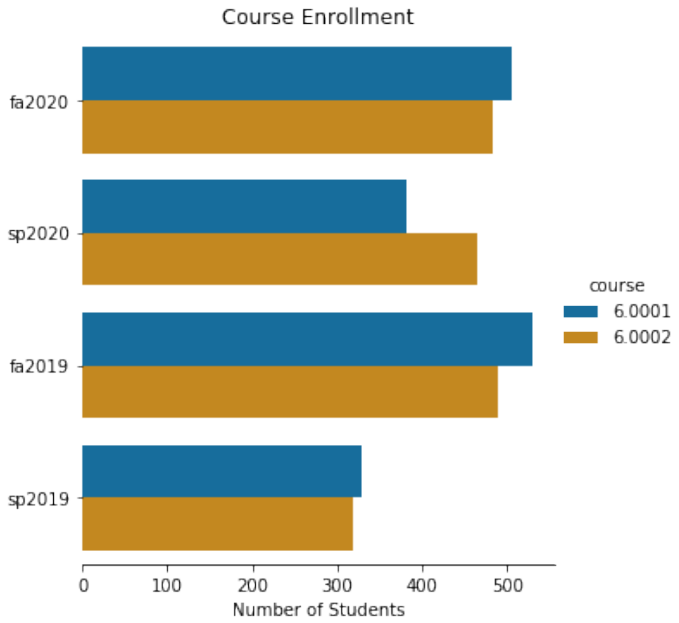


Figure 1-2: Number of students enrolled in past iterations of 6.0001 and 6.0002. The enrollment numbers are based on the number of students who've completed the course and received a grade as to not count those students who drop the course.

1.2 Motivation

The goal of this project is two-fold. First, on a more individualized and course specific basis, we hope to be able to alert staff when a student is not on track to pass the course so that the staff can quickly recognize and offer assistance and resources to students who need it. Especially during this shift to virtual learning sessions, it is difficult to keep track of student progress and notice when students are falling behind, so with this tool, staff are able to more closely monitor and help students struggling in the course. Secondly, we aim to understand and generalize trends and behaviors students with various grade trajectories follow. In doing so, we hope to uncover what habits and factors can contribute to being a successful student and how students can improve their trajectories. In order to do so, we aim to answer the following questions:

1.2.1 Research Questions

- How well can we predict student performance or grades at various points in the semester?
  - What machine learning algorithm(s) performs the best at predicting student grades?
  - Which features are indicative of high student grades?
- How often do students in danger or failing actually fail the course? What percentage of students in the category (danger of failing) actually end up failing?

1.3 Related Work

1.3.1 6.0001/2 MOOC Research

Since its creation, 6.0001/2 dataset has been used to study a multitude of student behaviors. Much of research, however, is focused on the Massive Open Online Course (MOOC) version of the course that is offered on the ed-tech platform edX, which allows anyone to access the course in either a free or paid version, where the paid version offers access to more materials .

One such work classifies students based on their learner type which is based on characteristics like their number of times the student has attempted the course, course version (either free or paid), course pace (either self- or instructor- paced), etc. then studies what factors affect the behaviors and outcomes of students with different learner types [7]. In another project, the 6.0001/2 MOOC data was used to understand student learning trajectories or behaviors over different time frames . These time frames studies were both over the course of solving a single problem and over multiple problems in order to understand which behaviors best support learning [4]. Another paper explored the effect that receiving grade feedback has on the behavior of students and ultimately concluded that students did not change their learning behavior after receiving a specific grade [10]. In this way, the grade prediction feature

we work on should not affect the way that students in the course learn the material nor affect their outcomes in the course.

The importance of research into this version of the course is invaluable as MOOCs make education more accessible for a wider range of learners. However, we intend to study the behavior of another important class of students, in-person learners, in order to understand and improve the way that versions of the courses are taught and managed. Similarly, in his paper, Vostatek analyzes factors that affect and strongly correlate with student performance in both the MOOC and in-person setting versions of 6.0001/2 and goes on to compare and contrast student performance in the two versions of the course, the MOOC version and the on-campus version [9].

1.3.2 Grade Prediction

There have also been many research efforts in predicting student grades. In one such work, Iqbal et. al. employ various machine learning models to predict grades from Information Technology University (ITU), Lahore, Pakistan and ultimately find that Restricted Boltzmann Machines were the most successful at correctly predicting student grades [8]. However, their dataset features focus more on the prior experience of the student (i.e. courses they've taken before, grades they've received in other classes, etc.) which differs greatly from the features in the 6.0001/2 dataset which is mostly comprised of data generated from interactions at the time of taking the course.

In Vostastek's aforementioned paper, he also studies grade prediction for in-person 6.0001/2 students and found some very promising prediction algorithms that have shown the ability to learn which features relate to a student's final grade [9]. Our work builds on this project in that we integrate the grade prediction feature into the course site, further study models that can more accurately predict grades, and investigate the best way to interpret and utilize the grade predictions to improve student performance.

1.4 Outline

The general outline of work that has been completed for this thesis project is as follows:

- Gather the relevant data from previous iterations of the courses.
- Train a predictive model to determine student grades based on their assignment scores, assessment scores, participation, and relevant click data from site.
- Design and implement interface for grade predictions.
- Continuously monitor the tool's performance at weekly intervals throughout the semester while the course is running.
- Study the trajectory of students whose grade trajectory ever is a failing grade or below a certain threshold throughout the entirety of their time in the course.
- Once grades are finalized, analyze the performance and usefulness of the tool with regards to improving student grade outcomes when compared to looking at individual student trajectories and comparing the course to previous iterations as a whole.

Chapter 2

Course Information

2.1 Course Overview

Initially, the material in 6.0001/2 were combined into a single semester course named 6.00 which was first taught in 2008. Now, 6.0001 and 6.0002 are offered as separate half-semester courses each with 6 units of credit[1] as of Spring 2019 with 6.0001 and 6.0002 being held in the first and second halves of each semester.

Officially titled *Introduction to Computer Science and Programming in Python*, 6.0001 focuses on the role computation can have in problem solving while teaching students how to write small programs and best practices in Python. The course assumes little to no prior knowledge or experience with the topics or programming and covers fundamental computer science topics from iteration to python data structures to algorithm analysis. 6.0002, or *Introduction to Computational Thinking and Data Science*, is the continuation of 6.0001 and thus assumes more prior knowledge and experience because 6.0001 is also listed as a formal prerequisite for the course. In 6.0002, students apply what they've learned in 6.0001 to real-world problems and phenomena with emphasis on relevant algorithms and topics like probability and statistics, optimized search, and machine learning.

Generally, a lot of freshman students enroll in one or both of the course(s) and during their first semester at MIT. During the first semester, freshman do not receive

[1] A normal full semester course is 12 units which roughly equates to 4 semester hours or credits.

letter grades on their transcripts; rather than showing the grade they received, they either get a *P* for a pass or *NR* for no record which signifies internally that they didn't pass the course but does not appear on their official transcripts. This tends to occur for most freshman in the Fall semester (i.e. unless they defer their admission by a semester) and can affect the way that students behave and perform in the class.

Although the course was initially only available to students at or cross-registered at MIT, as of 2012 it is now available online through MIT's massive open online course (MOOC) provider, edX. In this version of the course, students can get a certification for a small fee or participate for free with more limited access to materials. In this thesis, we will not include student data from the MOOC version of the course because of variations in grading breakdowns, incentives, course pace, etc.

2.2 Grading Scheme

The final grade in each course is determined by three assignment types and the breakdown is the same for both courses and can be seen in Table 2.1:

1. **Finger Exercises**: Short Python programming exercises that are due before every lecture. The purpose of these assignments are to confirm that students understand the topics covered in lecture and are answered and automatically graded on the course site.

2. **Problem Sets**: Longer Python programming problems that typically take 1 week. When a students submit a problem set, it is graded automatically with a testing suite hosted on the site. Also, students must also complete a checkoff for each problem set where the student meets with a Teaching Assistant (TA) or Lab Assistant (LA) to answer questions and discuss their solutions and concepts covered in the assignment. Students are allowed 3 late days which are extensions that can be applied to any problem set deadline.

3. **Microquizzes**: Timed 30-45 minutes quizzes in which students are given roughly 1-2 programming questions and 2-3 true/false or multiple choice questions.

These assessments are open-note however students may not collaborate nor use the internet during the quiz time window.

Table 2.1: Grading Breakdown in 6.0001/2

Assignment Type	**# Assignments**	**% of Final Grade**
Finger Exercises	6-7	10%
Problem Sets	5-6	45%
Microquizzes	best 3 of 4	45%

2.2.1 Final Grades

The final grade in the course is the weighted average of their scores on all the assignments. Students are guaranteed an A for a score >90%, a B for a score >80%, a C for a score >70%, and so on. For students that fall right around a boundary, the course staff - which is comprised of the lecturers and TAs - meet to discuss whether or not students should be bumped to the higher grade or not based on their progress in the class; interactions in lecture, office hours, or recitation; and any information the student provides them with (i.e. why they missed a deadline).

2.3 Resources Available to Students

Both courses have two 80 minute lectures led by one of three MIT faculty and an optional one hour recitation led by an undergraduate or graduate TA each week. Students also have access to office hours from 11AM - 9PM every Monday through Thursday and 11AM - 5PM every Fridays except during the three hours of scheduled lecture time each week. During each of these hours, there is one TA and three to four LAs that students can utilize to get help understanding concepts, debugging problem sets, and receiving problem set checkoffs. There is also an online Piazza forum where students can anonymously ask and answer each others questions about course content, logistics, clarifications, etc. Course TAs are also responsible for responding to Piazza

questions. Finally, students have access to video summaries and practice problems on MITx, an which offers massive open online courses through edX.

2.4 Buddy System

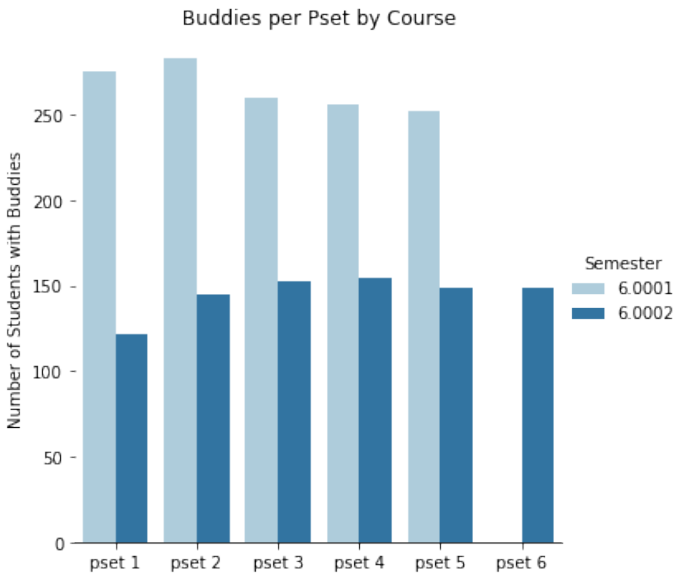


Figure 2-1: Number of students signed up for buddies for each problem set in 6.0001 and 6.0002 during the Fall 2020 semester when it was first introduced.

Usually, all assignments in the course are meant to be completed individually. Students are allowed to work together to understand topics but they are never to share or look at other student's code or assignments. However, due to the COVID-19 pandemic in the Fall 2020 semester, the course introduced the buddy system where students could opt to work with another student whom they could share code with to encourage the types of collaboration seen in dorms or office hours and help students interact and meet others. Any student opting into the system received a random buddy with a similar experience level and in a similar timezone. Initially, opting into the buddy system meant that for every problem set students would get a new buddy they were required to work with. However, midway through 6.0001 this requirement was relaxed so that students could opt in to the system on a per-problem-set basis, which has remained the course policy on buddies since then.

Chapter 3

Data and Models

3.1 Dataset

In order to predict student grades throughout the semester, we utilize a number of sources that the students interact with and aggregate them into one dataset to perform these predictions. From these sources, a dataset of approximately 80 features is created, and utilizing the Fall19 and Fall20 semesters' data, we obtain about 750 data points to use in our testing and training datasets. We specifically chose these two semesters, excluding Spring 2021, as the Spring 2021 was the semester in which students were forced to leave the campus mid semester due to the COVID-19 pandemic, leaving many abnormalities and inconsistencies in the data. Here we will discuss the three sources of data: the course site, the piazza forum, and MITx.

3.1.1 Course Site

The course site contains logs of all the site interactions (i.e. clicks, activities, etc), as well as student's assignment submissions, grades, feedback, etc. For the purposes of this thesis, we focus on the data collected in the student survey, the grades that students received on assignments (psets, checkoffs, finger exercises, and quizzes), student interactions with the queue, and submission times.

Student Survey:

At the start of the semester, students are asked to complete a background survey as an ungraded assignment that also serves to help students set up their programming environment before the course ramps up. The data collected in this survey is useful as it gives insight into which students might struggle more, making it easier to cluster them with students in prior semesters with similar experiences. The survey asks about the following topics:

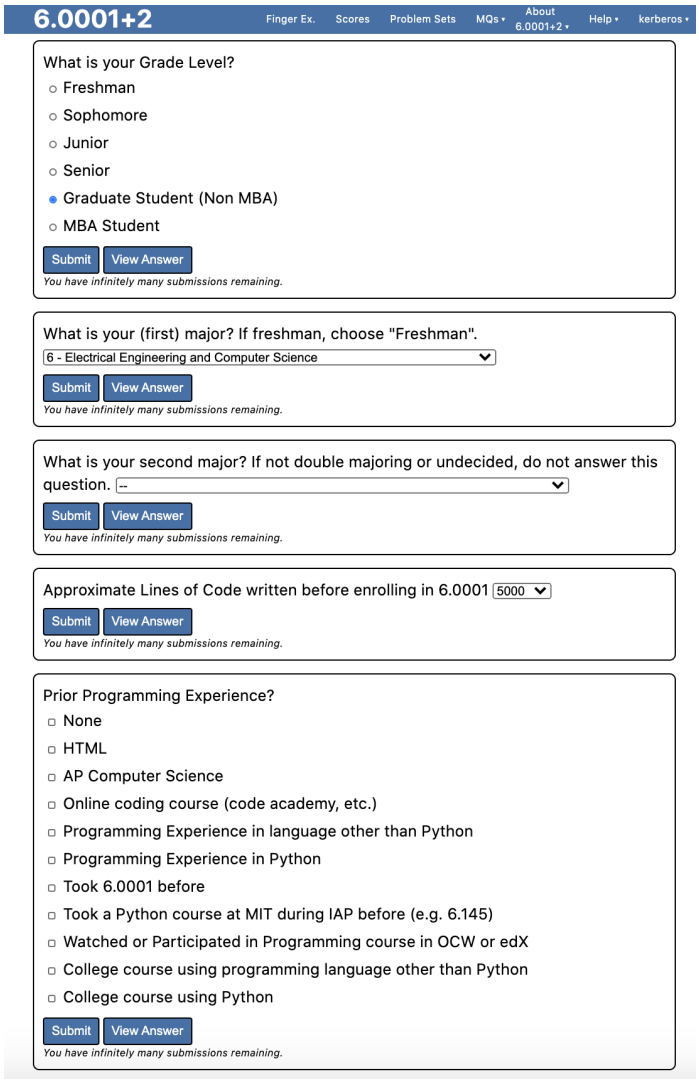


Figure 3-1: Survey that students are asked to complete at the start of 6.0001 or 6.0002 (if student didn't enroll in 6.0001 the semester in which they took 6.0002).

- **Grade Level**: Options include Freshman, Sophomore, Junior, Senior, Gradu-

ate Student (Non MBA), and MBA Student

- **Primary Major**: Options include all MIT majors plus a *Freshman* option for undeclared freshman
- **Secondary Major (if applicable)**: Options include all MIT majors
- **Approximate # of lines of code written prior to enrollment**: Options include 0, 50, 100, 200, 300, 500, 1000, 5000, and 10,000
- **Programming languages and/courses taken prior to enrollment**: Options include None, HTML, AP Computer Science, Online Coding Course (i.e. code academy), Programming Experience in Language Other than Python, Programming Experience in Python, Took 6.0001 Before, Took a Python Course at MIT During IAP Before (e.g. 6.145), Watched or Participated in OCW or edX, College Course using Programming Language Other Than Python, and College Course Using Python (multiple options may be selected)
- **Reason for enrolling in course**: Options include To learn How to Program, To Fulfill a Course Requirement, To Get a Good Grade, and Other (multiple options may be selected)
- **Resources that they are aware of/have used when learning to code**: Options include Google, Online Coding Courses, Stack Overflow, and Friends Who Know How to Program (multiple options may be selected)

Assignment Grades:

Another very useful indicator as to how a student is progressing through the course is their assignment grades, as these are what ultimately determine a student's final grade in the course. Particularly for 6.0001/2, that includes their scores on the psets and their respective checkoffs, as well as their quizzes and finger exercises. Because of the way in which they're formatted, we also have access to the number of attempted questions, and the number of times each question was attempted for finger exercises.

Queue Interactions:

We also utilize click data obtained from the course queue. The queue is a site feature used during office hours, where students can add themselves when they have questions about anything in the course (usually psets or to obtain a checkoff) and be claimed by an LA or TA who will answer, help debug, or perform the checkoff. The queue data is useful because it can help identify when a student is struggling or has a firm grasp of the content depending on the frequency of them adding themself to the queue.

Submission Interactions:

Finally, we also use the student submission times on psets in particular as part of our dataset. Submission history can give a lot of insight into a student's trajectory in a course as it can show how early or late the student submits in relation to the due date. If a student is struggling they might utilize all of the days up until the due date, whereas a student who finishes early might have a better understanding of the material.

3.1.2 Piazza

The course Piazza, as mentioned before, is a forum in which students can ask questions anonymously about anything related to the course. Students and staff can answer the questions and post related follow-up questions and tips. The forum has proven useful in helping students understand topics without needing to have a one on one interaction in office hours and giving them anonymity if they need or want it. We can extrapolate information about how a student is doing in the course based on how much they interact with this forum; like if they ask a lot of questions they might be struggling in contrast to answering a lot of questions which can indicate mastery of the material. From the Piazza site, we are able to export data on the number of days a student was active on the forum and how many questions they've answered, followed up on, or viewed.

3.1.3 MITx

The final source that student data is pulled from is MITx. Although interacting with the site is optional, many students utilize the available problems to study for quizzes and get more experience with the topics covered. From MITx, we can obtain records of each day that a student is active on the platform. For each of those days, we have information like the number of videos and/or chapters they've gone through, number of times they've paused videos, the amount of time between each session on the site for that day, number of problems they've attempted, etc. We suspect that these features can be used to help determine how a student will progress, as these habits give insight into the proactivity and study habits of various students.

3.2 Modeling

In order to model student grades over the course of 6.0001/2, we utilize a couple of different models to study which fits the problem best. First, there is the actual overall grade which acts as a benchmark for how well or poorly a student actually is doing throughout the course. Then, we looked at two different regressors to model the problem; the first of which, SVR, was used by Vostatek to also study predicting grades in 6.0001/2 and performed well in comparison to other models. We compare with an ensemble-based machine learning regression model in an effort to find a model the better suits the data [9].

3.2.1 Actual Overall Grade

The student's actual current overall grade at any point in the semester gives us an idea of how well the student is doing at a certain point in the course. To generate this value at a given point in the course, each student's overall grade is taken as the weighted average (using the respective assignment type weights from Table 2.1) of all of the assignments due by that point in the course.

3.2.2 Machine Learning Regression Models

Because we are predicting a student's grade as a number from 0-100, we looked at a number of machine learning regression models, and ultimately decided on the following two models because of how well they performed on the data as well as their ability to describe the importance of the features in terms of model parameters - Support Vector Regression (SVR) and Extra Trees Regressor (ETR). The latter is particularly useful for our application as we want to know what features are most relevant for predicting grades in order to advise students ways in which they can improve their grade trajectories.

Support Vector Regression:
As discussed in his paper, Vostatek utilized SVRs for this problem because of the of the dataset's high dimensionality and because of the SVR's ability to separate data based on these dimensions [9]. Although the 3 different kernels he used - Linear, Poly, and RBF - performed similarly well, we ultimately decided to utilize a Linear SVR with the same hyperparameters because of the ability to extrapolate the weights assigned to the features.

Extra Trees Regressor:
After testing many different regression models, we that found the Extra Trees Regression performed best on this dataset. Extra Trees is an ensemble-based method that works by averaging a number of randomized decision trees that are created on sub-samples of the data [6].

3.3 Data Prediction Pipeline

In order to interact with the grade prediction and monitoring tools we've created, we have also integrated some new features into the course site. In this section, we will discuss the data prediction pipeline set up to automate the prediction process throughout the various points in the semester. Because we are aggregating data from

a variety of sources, it can be difficult to organize, manage, and join all of the data. To account for this, we created a semi-automated system on the server hosting the course site which generates the datasets at given points in the semester. Because they are not on generated on the server, the MITx and Piazza data is manually pulled and updated on the server each week. Subsequently, scripts are run on the server that generates relevant datasets and makes predictions using all of the data sources.

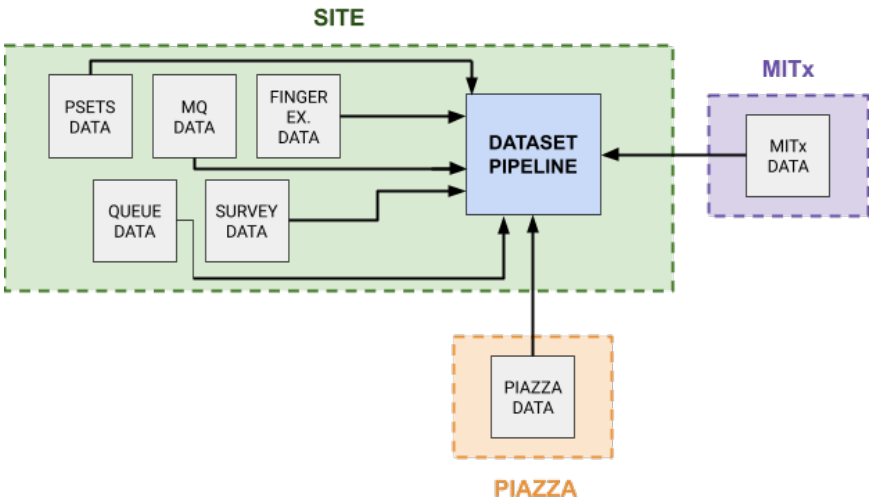


Figure 3-2: Overview of data sources from *Dataset* Section and how they are related to data pipeline for all semesters after Fall 2019.

First, the two non-site data sources are organized and formatted. Then, two datasets are created for the current week; one for the training/testing data used to generate the model and the other for the current semester data for which predictions will be made. In order to create these datasets, the current date is used to find how many days have elapsed since the start of the current iteration of the course. Then, for all of semesters, the datasets are generated using only assignments which have been due by that point in that semester in order to account for variation in due dates as well as the number of assignments duen from semester to semester. Around 80 features are collected and generated for each of the datasets which are described in Table A.1. Finally, a script is run to calculate the overall score and generate the ETR and SVR predictions for each student at that point in the semester.

Chapter 4

Experimental Setup

We evaluate our predictive models on a weekly basis for both the 6.0001 and 6.0002 courses which each last around 7 weeks. For each week's model, we evaluate the performance using three regression metrics to find the most effective model; we compare our predictive models to a baseline model using these metrics; and we analyze the relevance and importance of each feature for each model to understand what factors most contribute to more successful outcomes, all of which we discuss in this section.

4.1 Metrics

We utilize a few different regression metrics to measure and compare the performance of our models.

4.1.1 Mean Absolute Error

The mean absolute error (MAE) is the average in absolute errors over the entire dataset and is calculated as

$$\text{MAE} = \sum_{i=1}^{n} \frac{|y_i - \hat{y}_i|}{n}$$

where n is the number of samples in the dataset and y_i and $\hat{y}_i$ are the expected and

predicted values respectively. A nice property of MAE values is that they increase proportionally with error, rather than disproportionately penalizing larger errors more than smaller ones, which makes it more intuitive when compared to other metrics.

4.1.2 Mean Squared Error

The mean squared error (MSE) is the expected value of the squared error and is calculated as

$$\text{MSE} = \frac{1}{n}\sum_{i=0}^{n}\frac{(y_i - \hat{y}_i)^2}{n}$$

Squaring the difference between expected and predicted effectively magnifies any large errors, penalizing models more for larger errors.

4.1.3 Root Mean Squared Error

The root mean squared error (RMSE) is the expected value of the square root of the error and is calculated as

$$\text{MSE} = \sqrt{\frac{1}{n}\sum_{i=0}^{n}\frac{(y_i - \hat{y}_i)^2}{n}}$$

Because the root of the error is used, the units of the value returned by this error function match that of the expected value, making this metric more easy to understand when evaluating model performance when compared to MSE.

4.2 Baseline Model

We utilize a baseline model as a basis for which we compare the performance of the machine learning regression models used to predict grades. As our baseline model, we employ a very basic regressor which always predicts the mean of the training data.

Because of this, the metrics for this model are consistently the same for each week because the model is independent of all the features which fluctuate from week to week and is only dependent on the final outcomes.

4.3 Quantifying Feature Importance

In an effort to understand the relevance of different features in student outcomes, we aim to quantify the importance of the features for each of our predictive models. In doing so, we hope to use this information to advise struggling students what they can do to improve their grade trajectory. However, because of differences in the ways the models make predictions, we must look at different methods of quantified feature importance for each of the two predictive models we use.

4.3.1 SVR Feature Weights

For the SVR model, the feature importance are represented as the weights assigned to the features by the model, which describes the correlation of the features and the outcome. The sign indicates whether the feature is positively or negatively correlated with the outcome, and having a larger magnitude signifies a stronger correlation relative to the other features. For example, on the feature table in Figure 5-4, in Week 1, we see that `workahead_time` = 11.8 means that the feature `workahead_time` has a feature weight of 11.8, which is the highest of all the features studied. This tells us that the time that students work ahead of the problem set deadline is most highly indicative of student success in this week so, for this a value of 4 days is better than a value of 1 day. However, for a feature like `nprob_attempt` = -5.19, having a lower number of problems is more strongly correlated with a better outcome.

4.3.2 Extra Trees Regression Gini Importance

For the Extra Trees Regression model, the feature importances are impurity-based, also called Gini importance or Mean Decrease in Impurity (MDI). A higher, the

more important the feature. Essentially, Gini importance is a measure of the average decrease in impurity in our decision trees for a given feature. Unlike the SVR model however, this measure of importance does not correlate our features and outcome, but rather is measures how useful the feature was in reducing the variance of our model.

Chapter 5

Results

In this section, we discuss the end result of our integrated system onto the course website including the visualization pages to view and interact with these predictions and student progress in 6.0001/2. We will analyze the performance of the predictive models with respect to the metrics described in Section 4.1. We will also analyze and discuss the features which were most indicative of successful outcomes in the course in terms of the model specific importance metrics described in Section 4.3. Finally, we will look at trends in student trajectories to understand and generalize student behavior in the context of 6.0001/2.

5.1 Website Integration

Equally as important as our models' predictive capabilities is an effective, concise method for interacting with these predictions. We have created multiple pages on the course website that display information about students in the course as a whole, and also about students on an individual level. In this way, staff can quickly parse at information pertaining to the entire class and make generalizations about things like the difficulty of the assignments, but also they can see details about each student and intervene when students struggle.

5.1.1 Overall Progress Page

Course Grades by Student

Filter By:

Grade Level | Department | Lines of Code | Experience | Motivation | Resources

Number of Students: 79 Filter

	Scores															
	Overall	Psets					Exams				Finger Ex					
	-	PS 1	PS 2	PS 3	PS 4	PS 5	MQ 1	MQ 2	MQ 3	MQ 4	FE 1	FE 3	FE 5	FE 7	FE 8	FE 10
student21	1.0	0 0 | 0	0 0 | 0	0 0 | 0	0 0 | 0	- -	0.0	0.0	0.0	0.0	60.0	0.0	0.0	0.0	0.0	0.0
student36	41.87	0 0 | 0	15.5 0 | 0	0 0 | 0	0 0 | 0	- -	7.98	8.85	6.21	5.9	85.7	100.0	0.0	100.0	100.0	100.0
student205	56.79	100 0 | 0	98.7 0 | 0	0 0 | 0	100 0 | 0	97.2 -	6.48	9.27	0.0	5.6	71.4	100.0	100.0	83.3	100.0	100.0
student115	66.41	90.6 0 | 0	100 0 | 0	69 9 | 10	94.4 10 | 10	94.3 -	7.63	5.53	6.4	0.0	100.0	100.0	100.0	100.0	100.0	100.0
student130	67.0	99.1 10 | 10	0 10 | 10	100 10 | 10	100 8 | 10	0 -	5.24	5.18	3.91	3.8	100.0	100.0	100.0	100.0	100.0	100.0
⋮																
student275	93.07	100 10 | 10	98.7 10 | 10	97.6 10 | 10	100 10 | 10	- -	7.98	8.92	7.81	9.6	85.7	100.0	100.0	100.0	100.0	100.0
student263	93.82	100 10 | 10	100 10 | 9.5	71.4 10 | 10	100 10 | 10	100 -	10.0	9.25	10.0	9.0	85.7	100.0	100.0	100.0	100.0	0.0
student203	93.83	92.5 10 | 10	98.7 10 | 10	85.7 8.5 | 10	100 10 | 10	0 -	7.98	8.85	10.0	9.6	100.0	100.0	100.0	100.0	100.0	100.0
student34	94.7	94.3 10 | 10	98.7 10 | 10	100 10 | 10	100 10 | 10	0 -	7.63	8.85	10.0	9.3	100.0	100.0	100.0	100.0	100.0	100.0
student25	95.07	91.5 10 | 10	98.7 10 | 10	100 10 | 10	100 10 | 10	100 10 | 10	9.6	8.1	10.0	8.6	100.0	100.0	100.0	100.0	100.0	100.0
student198	95.55	100 10 | 10	100 10 | 10	71.4 10 | 10	100 10 | 10	97.2 10 | 10	9.2	9.25	10.0	9.6	100.0	100.0	100.0	100.0	100.0	100.0

Figure 5-1: Course progress overview as a collective in terms of current overall grade and assignment scores.

In order to visualize student progress relative to other students and keep track of students in a more general sense, we implemented an overall progress page. Here, staff can view all of the students grades on any assignment as well as the students' current overall grade at that moment in the semester in a way that is easy to understand and navigate. On the page is a filterable table with columns for kerberos, overall grade, each pset (checkoffs are shown within the corresponding pset column), each microquiz, and each finger exercise. The rows in the table are sorted in ascending order of current overall grade. The first column holds the student kerberos (which

have been replaced with random student numbers in Figure 5.1.1 to maintain student anonymity) each which are hyperlinked to the student's Individual Student Progress Page (discussed in Section 5.1.2).

Grade values in the table are color coded such that dark green indicates an A (above 90% on the assignment) and dark red indicates an F (below 50% on the assignment) as shown in Figure 5.1.1. The current overall grade is calculated using only the assignments that are due on or before the current date, however students can turn in assignments early if they've already been released. In order to handle this, boxes in the scores table are shaded light grey to signify that that assignment has already been due and is being used in the overall grade calculation; any other grades are shown in boxes shaded white but are not used in the overall grade. Also, grades for items not yet turned in are signified by a '-' in the table.

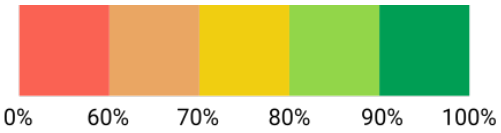


Figure 5-2: Color scale for displayed grades used in both the Overall Progress Page and the Individual Student Progress Pages

There are also options to filter the rows shown by experience and background. Staff is able to filter by any of the questions asked in the student survey (see Section 3). This way, it is easy to group students based on their experience and background to find trends among these groups while easily monitoring students with little to no familiarity with the course material.

5.1.2 Individual Student Progress Page

To visually monitor each student's progress throughout the course, we integrated an Individual Student Progress page for each student on the course site. This page has two main components; the student overview and the grade trajectory information as shown in Figure 5-3 and Figure 5-4 respectively. Arranging the prediction information in this way makes it convenient for staff to view a student's progress in the course

Introduction to Computational Thinking and Data Science (Spring 2021)

Kerberos: student
Name: Unknown
Department: Freshman (Undeclared)
Class Year: Freshman

Background Info			
Lines of code written	Experiences	Motivations	Familiar Resources
~ 1000	html, ap comp sci, college course w/ python exp	learn programming	google, stack overflow, friends with programming exp

Scores														
Psets					Exams				Finger Ex					
PS 1	PS 2	PS 3	PS 4	PS 5	MQ 1	MQ 2	MQ 3	MQ 4	FE 1	FE 3	FE 5	FE 7	FE 8	FE 10
100.0	98.7	100.0	94.4	-	5.96	5.05	7.01	4.3	85.7	92.9	100.0	100.0	100.0	100.0

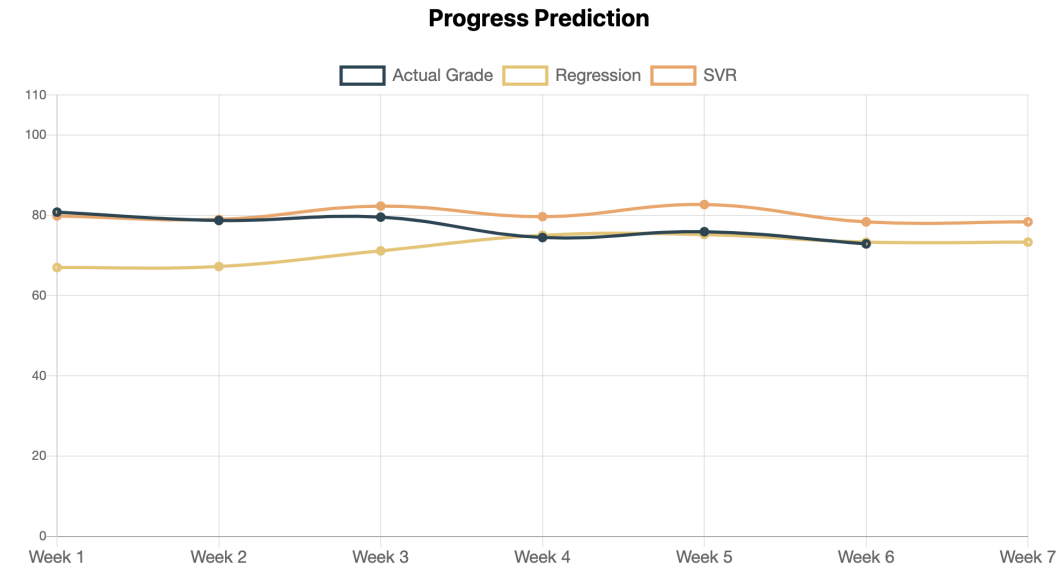
Figure 5-3: Individual student overview comprised of student survey responses and assignment scores which is displayed on each individual student progress page.

as well as easily determine and understand what factors can lead to improving a student's trajectory at each week in the course.

The student overview condenses all of the information available from the student survey discussed in Section 3.1.1 as well as the student's scores on all assignments they've turned in. Much like the overall progress page, a '-' in the scores table signifies that the student has not turned in a given assignment and grade values for these assignment scores are color coded in the same way as in the Overall Progress Page shown in Figure 5.1.1.

The grade trajectory information includes a weekly prediction chart along with a features table as shown in Figure 5-4. The interactive prediction chart displays three lines representing the actual grade, the Extra Trees Regression predictions, and the SVR predictions for that student at that point in the course as described in Section 3.2.

The features table displays each feature along with a measure of it's importance in sorted order of absolute value for each model each week. The information is color



Feature Importances/Coefficients

Week 1		Week 2		Week 3		Week 4		Week 5		Week 6		Week 7	
time_after_submit	0.2	pset_avg	0.28	pset_avg	0.37	pset_avg	0.4	pset_avg	0.4	pset_avg	0.41		
workahead_time	0.06	quiz_avg	0.21	quiz_avg	0.21	quiz_avg	0.18	quiz_avg	0.25	quiz_avg	0.26		
grade_level	0.06	time_after_submit	0.04	course 15	0.03	checkoff_avg	0.05	checkoff_avg	0.07	checkoff_avg	0.06		
nprob_attempt	0.04	course 16	0.04	fex_correct	0.02	course 15	0.03	fex_correct	0.02	fex_correct	0.02		
course 16	0.04	avg_submits	0.03	checkoff_avg	0.02	fex_correct	0.02	nfex_attempt	0.01	course 15	0.02		
course 2	0.04	course 15	0.03	time_after_submit	0.02	grade_level	0.02	nfex_answered	0.01	nfex_attempt	0.01		
workahead_time	11.8	pset_avg	28.72	pset_avg	28.2	pset_avg	30.42	pset_avg	29.1	pset_avg	28.83		
ndays_act	9.44	quiz_avg	19.01	quiz_avg	17.8	quiz_avg	18.58	quiz_avg	27.25	quiz_avg	26.94		
course 5	8.22	course 21H	8.23	fex_correct	6.54	checkoff_avg	10.41	checkoff_avg	12.47	checkoff_avg	12.63		
lines_of_code	7.6	fex_correct	6.98	ndays_act	6.01	fex_correct	6.6	nprob_attempt	-6.5	ndays_act	4.98		
nprob_attempt	-6.19	workahead_time	6.34	nfex_attempt	-5.89	nprob_attempt	-5.95	fex_correct	4.78	fex_correct	4.62		
nvideos_view	-5.7	ndays_act	5.44	checkoff_avg	5.55	nfex_attempt	-5.93	grade_level	-4.51	grade_level	-4.52		

Figure 5-4: Grade trajectory visualization comprised of interactive chart of student grades prediction over the weeks of the course along with table of feature importances for SVR and coefficients for Extra Trees Regression model which are displayed on each of the individual progress pages.

coded in the same way as the progress prediction plots such that the colors correlate to the model's feature importances — yellow for Extra Trees Regression and orange for SVR as discussed in Section 4.3.

5.2 Performance Analysis

In order to understand and compare the performance of each model relative to one another, for each week, we look at the metric values for the baseline model, the Extra Trees Regression (ETR), and the Support Vector Regression (SVR).

5.2.1 MAE

Week	**6.0001**			**6.0002**		
	Baseline	**ETR**	**SVR**	**Baseline**	**ETR**	**SVR**
1	12.363	30.782	**11.835**	**10.097**	21.165	16.792
2	**12.363**	26.319	21.689	10.097	**8.815**	9.153
3	12.363	13.038	**11.5**	10.097	**5.461**	7.042
4	12.363	7.037	**6.869**	10.097	**5.843**	6.397
5	12.363	**4.884**	5.641	10.097	**3.83**	5.413
6	12.363	**3.241**	7.195	10.097	**4.033**	5.232
7	12.363	**2.879**	4.914	10.097	**3.355**	5.608

Table 5.1: Weekly breakdown of **MAE** values for each model separated by course. Smaller MAE values signify smaller error in the predictions and are therefore better. For each week and course, the smallest value among three models is bolded.

In Table 5.1, we see the weekly mean average error values for our baseline model along with our two machine learning models separated by course. For 6.0001, in the first week, the SVR model slightly outperforms the baseline model and greatly outperforms the ETR model. For 6.0002, the baseline outperforms both other models. This is likely due to grades being greatly inflated the first week because nothing is due, however the first deadline occurs in week two or three (depending whether the course is 6.0001 or 6.0002) which would cause a lot of noise in the data.

However, by the next couple of weeks for each course, we see the ETR and SVR models' MAE values decrease by a factor of about four and begin to outperform baseline model. As more data is made available to the models each week, we see that their MAE values significantly decrease as well. The two models perform similarly up until the middle of the courses (weeks 2-4) when we see the ETR begin to significantly

outperform the other two models.

5.2.2 MSE

Week	6.0001			6.0002		
	Baseline	**ETR**	**SVR**	**Baseline**	**ETR**	**SVR**
1	365.491	1073.282	**288.263**	**364.962**	799.424	490.527
2	**365.491**	776.881	537.022	364.962	**102.582**	152.979
3	365.491	221.302	**199.344**	364.962	**51.34**	101.09
4	365.491	**66.029**	73.639	364.962	**55.79**	78.582
5	365.491	**36.767**	47.365	364.962	**27.979**	52.77
6	365.491	**21.515**	66.474	364.962	**28.013**	50.02
7	365.491	**17.046**	34.313	364.962	**22.916**	53.651

Table 5.2: Weekly breakdown of **MSE** values for each model separated by course. Smaller MSE values signify smaller error in the predictions and are therefore better. For each week and course, the smallest value among three models is bolded.

Table 5.2 displays the weekly mean squared error values for all three of the models by course. Here, we see similar trends as the MAE values discussed in 5.2.1, the main difference being that the ETR starts to outperforms the SVR model earlier on in the weeks for 6.0001.

5.2.3 RMSE

Finally, the weekly root mean squared error values for each model by course are shown in Table 5.3 and follow the almost the exact same trends as the MSE values, as these values are simply the square root of the MSE values.

5.2.4 Summary of Performance

In general, for each metric, we see that the ETR model initially underperforms when compared to the baseline and SVR models, which is attributed to the fact that there is little information available to the models during the beginning of each course. However, the ETR significantly outperforms both the baseline and SVR model very

Week	6.0001			6.0002		
	Baseline	**ETR**	**SVR**	**Baseline**	**ETR**	**SVR**
1	19.118	30.34	**16.971**	**19.104**	27.199	22.151
2	**19.118**	28.053	23.139	19.104	**10.15**	12.386
3	19.118	**13.731**	14.168	19.104	**8.775**	10.043
4	19.118	**7.641**	8.579	19.104	**7.671**	8.892
5	19.118	**5.917**	6.911	19.104	**5.51**	7.262
6	19.118	**4.762**	8.167	19.104	**5.286**	7.07
7	19.118	**4.147**	5.873	19.104	**4.82**	7.334

Table 5.3: Weekly breakdown of **RMSE** values for each model separated by course. Smaller RMSE values signify smaller error in the predictions and are therefore better. For each week and course, the smallest value among three models is bolded.

quickly. After the first couple of weeks, the ETR model begins making very accurate final grade predictions in terms of each metric and compared to both other models.

These trends highlight the correlation between our features and student grade outcomes, especially as the semester progresses and more data is available to the models. However, it also highlights the difficulty in predicting grades in the absence of assignment scores and student behavior related features — like time spent on psets, number of submissions to assignments, etc. — , which is made evident by the poor performance in the first couple of weeks of both courses.

We also see that the model ETR model seems to better fit the 6.0002 data as compared to 6.0001 because it takes longer for the ETR to outperform the other models in 6.0001. These courses, although similar, vary significantly in content and prior experience, as 6.0002 is a more even playing field since it is required that students complete 6.0001 before taking it. Also, the content in 6.0001 builds upon itself much more so than 6.0002, whereas 6.0002 is explores a wider variety of computational topics that don't necessarily depend on each other. This could explain this discrepancy in model performance between the courses, and suggests that using a different model for each course could prove more successful in predicting grades.

5.3 Feature Importance Analysis

To understand which features are most important in determining a student's outcome in the course, we will look at the feature importance of each of our predictive models in terms of the quantifying metrics described in Section 4.3.

5.3.1 SVR Analysis

For the SVR model, we look at the feature weights assigned by the model, which is shown in Figure 5-5 for each week and course. The top seven largest positively and negatively valued features for the model are displayed in order.

The first week's values are particularly interesting, as there is no assignment score data available yet since no assignments are due by that point. We see that in both semesters, the `workahead_time`, which is the amount of time on average that a student starts the pset relative to the due date, has the highest valued weight. This implies that students with a larger `workahead_time` are predicted to do better, which makes intuitive sense because students who start earlier on assignments would have more time to fix mistakes and get better scores. We also see some other features with high feature weights that one would expect to correlate with doing well in a course; like `n_days_act`, `avg_submits`, and `lines_of code`, which are the number of days active on MITx, the average number of pset submissions, and the amount of lines of code written prior to enrollment respectively. However, again since there is very little data available at the start of the course, there are some noisy features that don't intuitively correlate with our outcome; like `course 5` and `course 20`, which signify that a student is a chemistry or biological engineering major respectively.

We also see many large negatively correlated features which one might expect that appear in the first week and also persist throughout the semester; like `nprob_attempt` and `nprob_checked` (the total number of MITx problems attempted and checked respectively) and the features pertaining to MITx videos — like `n_videos_view`,`nvideo`, and `nseek_video` (the total number of MITx videos watched, videos interacted with, and the average number of seeks in videos respectively). The problem attempts fea-

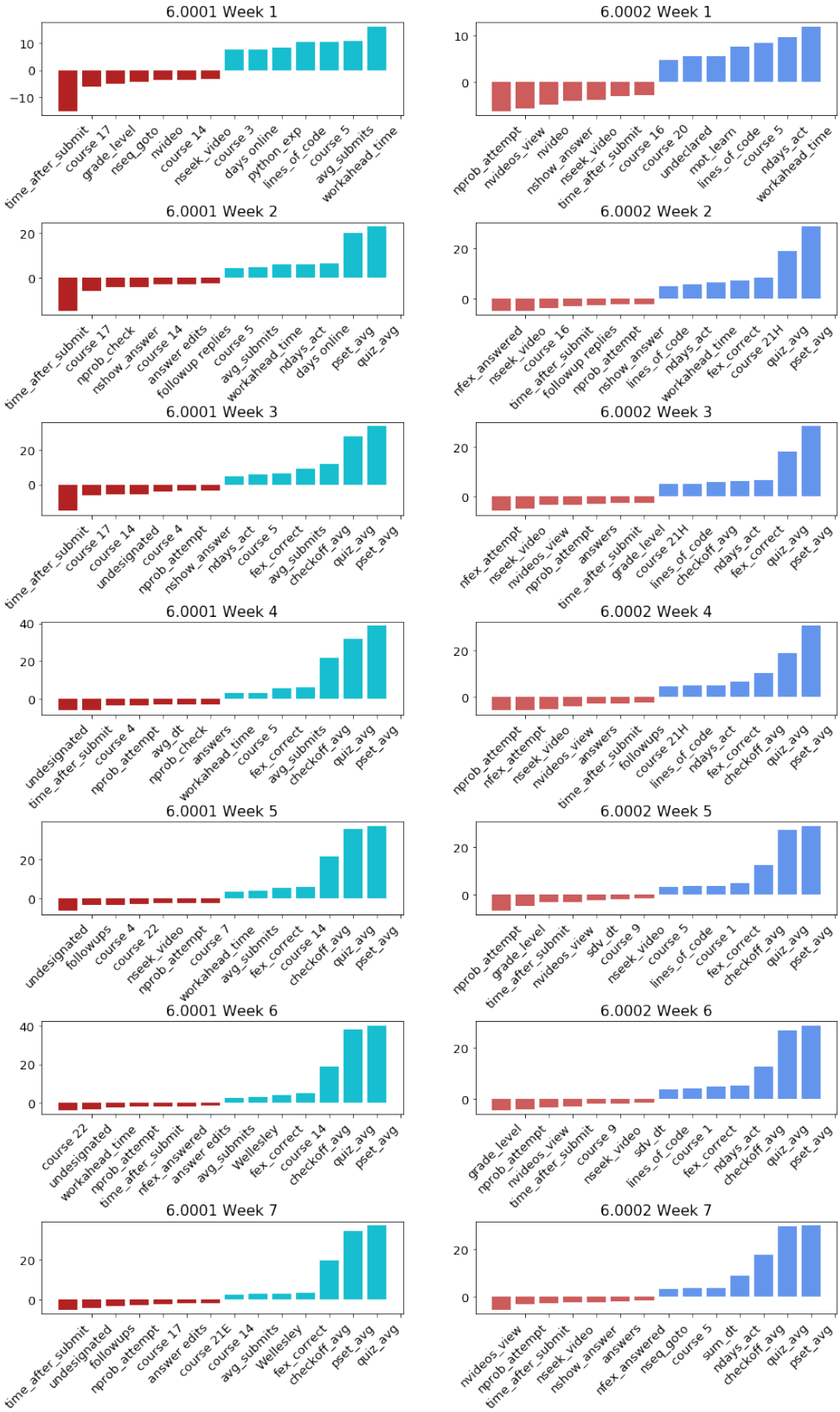


Figure 5-5: Top 14 highest feature weights by absolute value for SVR model per week separated by course.

ture being negatively correlated implies that more attempts indicates more guessing rather than actually learning and understanding concepts. Similarly, the MITx video features being negatively correlated indicates that repeatedly watching/re-watching and repeatedly interacting with videos (i.e. seeking) implies difficulty understanding of concepts.

As the weeks progress, we see that for both courses, the largest absolute valued features are the `pset_avg` and `quiz_avg`. Each week, the other features' magnitude decreases and by the last week, it is clear that these two features are becoming the most strongly correlated with the outcome. After these two features, the `checkoff_avg` and `fex_correct`, are the largest. These features represent the only aspects of the course that count towards the grade, so as the weeks go by and more data on these features are collected, they become the most important to determining a student's success. The other features cannot directly impact the student's outcome like the scores on these assignments, so they are much more relevant at the start of the course when assignment scores fluctuate greatly.

One consistently interesting feature for each week's models is the `time_after_submit` which is the average amount of time between each final pset submission and it's due date. One could expect that this feature would be positively correlated because students who submit psets early, likely would do better than ones who turns things in closer to the deadline. However, the feature consistently has a large negative feature weight. This implies that a smaller `time_after_submit` leads to better outcomes. This is likely to be attributed to having more time to catch and fix issues in psets when turning them in later. In certain weeks, like the first one, `workahead_time` has large positive correlation in contrast to `time_after_submit` having a large negative correlation to the outcome. Although this may seem contradictory, this simply implies that starting the psets early and turning them in near the due date, or in other words spending the full week on the pset, results in better outcomes. This makes sense since utilizing more time to work on the pset allows more time for students to find bugs and get exposure to the concepts and topics covered on the assignments.

Another interesting comparison is that of 6.0001 versus 6.0002 with respect to

which features are most indicative of dictating student success. Most of the features are generally consistent between the two courses, as those features pertaining to the same sources (like MITx) have similar correlations in terms of magnitude and sign. However, we also see that the `lines_of_code` feature is consistently a highly positive valued feature in 6.0002 but the same is not true of 6.0001. This validates that prior coding experience is not as necessary in 6.0001, where students learn the basics of programming, as compared to 6.0002 which assumes experience with programming and is more focused on computational thinking.

5.3.2 ETR Analysis

The ETR model utilizes Gini Importance to represent how important each feature to the model, which can be seen in Figure 5-6. The top ten highest valued features are displayed in order of magnitude for each week and course.

Looking at the first week's most important features we see that some of the top features are also features that had large values for the SVR model like `avg_submits` and `workahead_time`. However, we also see a lot more importance given to the survey related features, like `grade_level`, `exp_None`, and `exp_python`, which represent the student's grade level and experience level. These features would intuitively be most useful at the start of the course as it should represent how comfortable a student may or may not be with the course material.

Quickly, in the subsequent weeks, the ETR model gives more and more importance to the quiz and pset scores. Much like the SVR model, we see that the importances are converging to values that mimic the grading breakdown of the course shown in Table 2.1. This is clear as the `quiz_avg`, `pset_avg`, `checkoff_avg`, and `fex_correct` importance values grow, while all other feature importances approach zero throughout the progression of weeks.

We also see that experience is more important in 6.0002 when compared to 6.0001, which is consistent with the SVR feature weight results. This is evident through features like `exp_python` and `awr_stack_ovf` which signifies that the student has prior experience with python and is aware of Stack Overflow as a resource respectively.

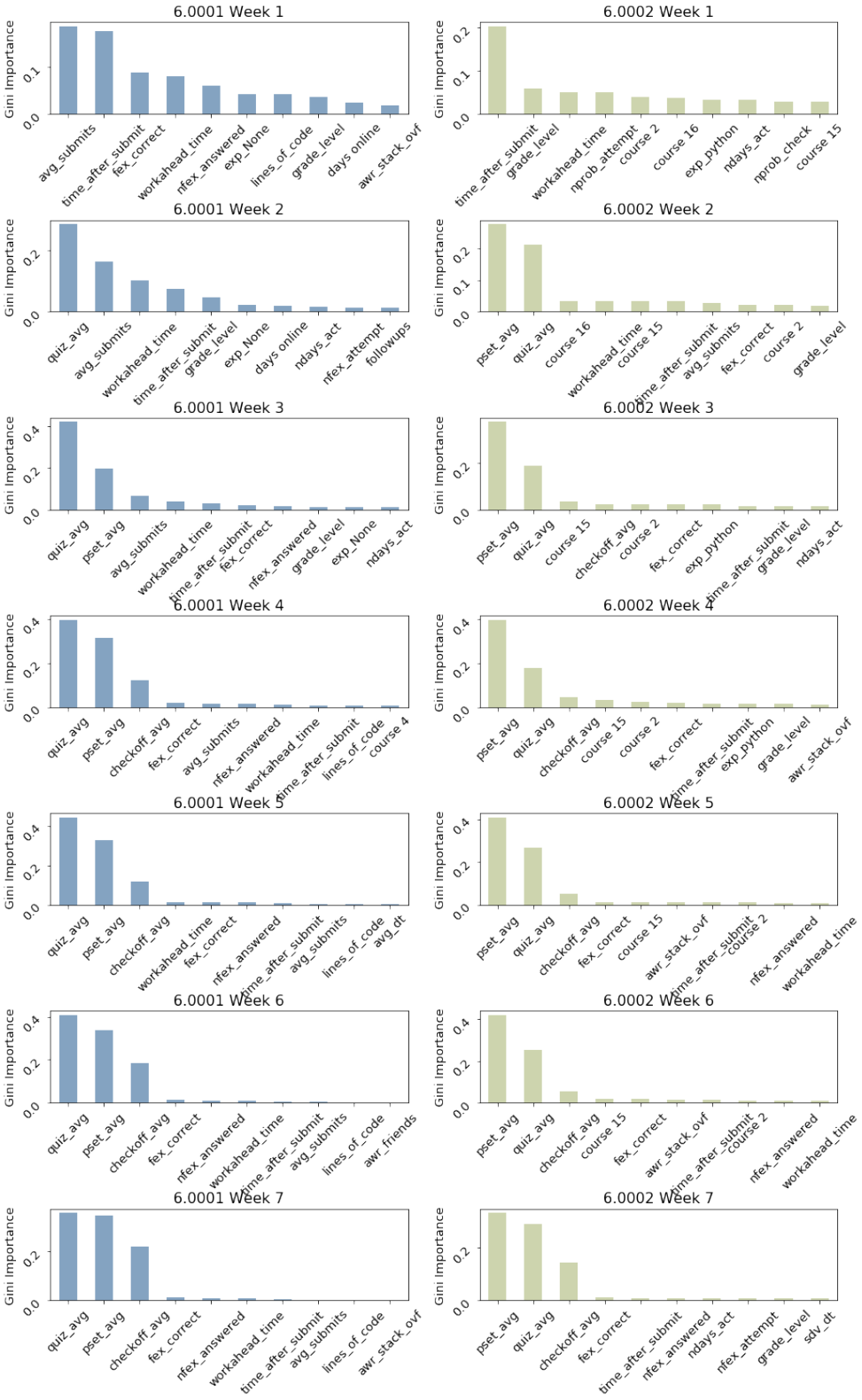


Figure 5-6: Top 10 feature with highest Gini Importance values for Extra Trees Regressor by week separated by course.

These features have relatively high importance values in 6.0002, and although in 6.0001 the `lines_of_code` feature is important in the later weeks, it's importance value is much smaller than the other features.

5.3.3 Comparison of ML Models

As we've discussed, there are a lot of similarities between the trends found in the feature importance values for each model. Specifically, we see similarities in the way that the models utilize the features in the absence of score data at the beginning of the course and how they converge to similar importance levels throughout the course of the semester. Because of this, the most useful weeks for utilizing these grade predictions are those in the middle of the semester.

Although the ETR model generally outperforms the SVR model, the SVR model is powerful in that it's feature weights can be more expressive in explaining the importance of the features. This is because the SVR feature weights correlate the feature values with the outcomes. By looking at the signs of the weights we can determine the relationship between the feature and the outcome, whereas the Gini importances simply tells us how useful a feature was for the model.

The two models' feature importances, however, cannot be directly compared as they utilize different methods and calculate different feature importance values, although both provide useful information about which features are most indicative of successful student outcomes in these courses.

5.4 Student Trajectory Analysis

Now we will analyze and discuss general patterns in trends using the student trajectory data obtained from our machine learning model predictions.

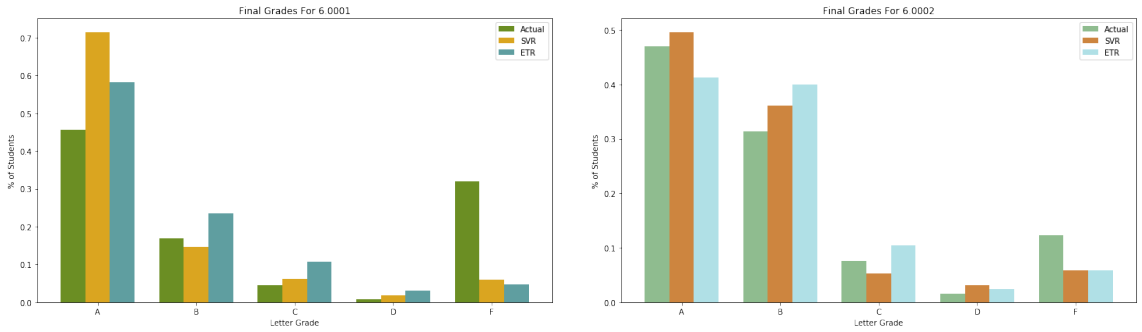


Figure 5-7: Final grades breakdown (by percentage) for the Spring 2021 semester split by course. Shows both the actual final breakdown as well as the predicted breakdowns for the ML models.

5.4.1 Final Grades

First, we will look at the final grades breakdown for this semester of 6.0001/2 to contextualize the predictions made by our machine learning models. In Figure 5-7, we can see a side by side comparison of the predicted and actual grade breakdowns. The predictions made by the models do not factor in any intervention measures by staff.

As shown, most students in the course get an A or B and very few get D's or F's. We also see that both of the models' grade breakdowns are fairly close to the actual outcomes. One outlier is the number of A's predicted by the SVR model, however this matches up with our results as the SVR model does not perform as well as the ETR model — with respect to our metrics as discussed in Section 5.2 — for predicting grades especially at the end of the semester.

One thing to note is that the number of actual failures are significantly higher than the predicted, especially in 6.0001. This fluctuation can be attributed to the many students who forget to drop the course early on in the semester and thus turn in little to no assignments. This is a common occurrence in these courses especially with students who put the course on Pass/No Record grading.

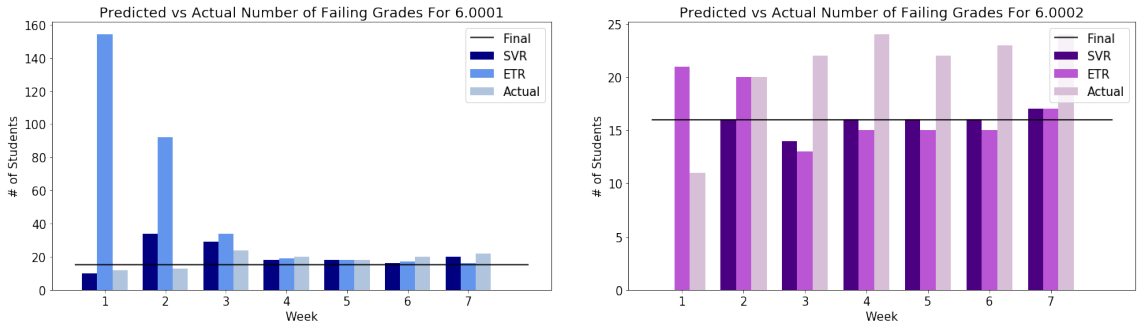


Figure 5-8: Number of students projected to fail — predicted to get a score between 15% and 60% — by each of the ML models alongside the number with failing grades each week over the course of semester for each course. The 15-60% range is used to account for students who meant to drop the course but have not — a common occurrence in these courses. The horizontal lines above the bars in the chart represent the final number of students whose actual grade was failing at the end of the semester.

5.4.2 Students in Danger of Failing

As discussed, we are most interested in the behavior of students who end up failing, as students who follow such a trajectory should be reached out to in an effort to help them improve and understand course content. Figure 5-8 displays the number of students predicted/actually failing at any given week in the course along with a line signifying the final number of students who've failed the course. The final value is calculated after all grade adjustments including regrades and the dropping of the lowest microquiz.

As we have seen throughout, the first week or two have very ambiguous predicted values that vary wildly with the actual values because of the lack of data. However, one interesting thing we do see from these numbers is that the models tend to get very close to the final value by the middle to end of the course. This is very important as we can accurately reach out to struggling students and intervene when necessary to help students get back on track.

5.4.3 General Patterns

In an effort to understand when it is best for staff to intervene, we will discuss the general projections made by our models as compared to the actual values each week.

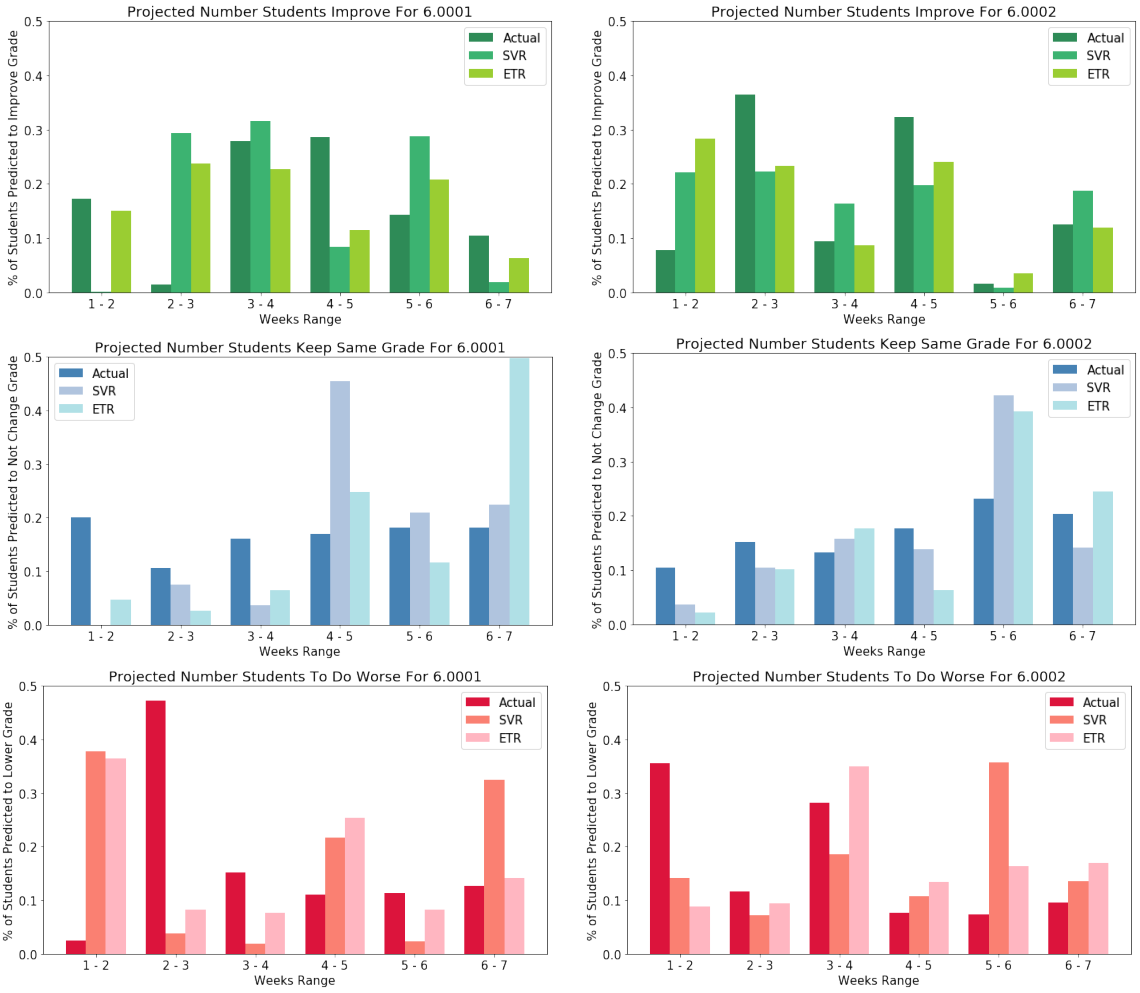


Figure 5-9: Student Grade Change Projections for each model and each semester. Each row of charts shows the percentage students are predicted to improve, stay the same, or worsen their grade respectively. We define a grade as staying the same if it is within a ± 1 percentage points threshold.

Figure 5-9 shows the percentage of students projected to improve, keep, or worsen their grades from the beginning of one week to the end of the same week. We show these grade projection percentages for the seven weeks of each course. These percentages are calculated as the number of students per week in a given category — improving, keeping same, or worsening their grade — out of the total number of students to be in that category over the course of each class. For example, in weeks 1-2, the dark green bar with a value of $\sim$17% represents the percent of `actual` number students who improve in weeks 1-2 out of all students who improve over all weeks.

We see that the models both follow the same general trends as the actual outcomes

with some slight variation. As expected, the values for the first week are not very conclusive nor informative because of the lack of data. What we do see, however, is that students tend to have higher improvement at the beginning to middle of the semester around weeks 2-5. We also see that generally, the percentage of students that keep the same grade stays at around 10-25% throughout the entire course, although the models predict higher values during the latter end of the course. Also, students tend to start to do worse at the beginning/middle of the course — at around weeks 1-4. Then we see these numbers stay relatively low after these weeks.

The trends seem fairly consistent between the two courses, however we do see some discrepancies especially around the ends of each course. The beginnings of each course also tend to differ, however this is likely due to the lack of data available to the models during these weeks. For example, looking at the improvement values, we see that weeks 5-6 are very different when comparing courses – the values are very high in 6.0001 and very low in 6.0002. However, there were also many differences in assignment due dates between the courses, especially near the end of each course. For example, in 6.0001, the final pset has no checkoff wheres the 6.0002 counterpart does have a checkoff and the final microquiz in 6.0001 was in the 6^{th} week of the course and in the 5^{th} for 6.0002. Also, the final pset in 6.0002 was assigned a lot earlier in the course this semester since it was extended to include more topics. Therefore these differences in the trends between the courses, could be due to these variations in assignment due dates between the courses.

We care about reaching out to students that are falling behind, therefore, staff should reach out to students when they are expected to do worse. For this reason, these weeks at the beginning to middle of the course (weeks 1-4) are optimal for intervention.

Chapter 6

Conclusion

6.1 Future Work

Now that the prediction capability is integrated into the 6.0001/2 course site, there is an opportunity to utilize it to help struggling students. Thresholds can easily be set such that if a student's predicted grade is lower it, staff will be alerted. In this scenario, staff can utilize the feature importance values of the predictive models to direct students to methods and behaviors that can improve their outcomes. With this in mind, a couple of open questions include:

- How much does the tool contribute to decreasing the number of students that end up failing the course when compared to past semesters?
- How many students do staff reach out to due to the grade they're predicted to receive?
- How many students respond and/or utilize more resources following interactions with the staff regarding their predicted grades?

Another natural extension of this project is to apply these predictive models to other formats of the course, like the edX MOOC that is available to the public. In doing so, we can study the differences in grade trajectories and important features

within these different contexts to better understand learner behaviors as students are motivated and incentivised in very different ways.

6.2 Final Thoughts

Education is unequivocally important for advancement in many aspects of society, including financial security, job security, food security, etc.. Furthermore, as more and more jobs are beginning to be automated, the need for technical skills in fields like computer science are becoming more and more necessary. Therefore, understanding how students learn and what behaviors can contribute to success in courses like 6.0001/2 is key to improving educational processes in the burgeoning tech sector.

As we have discussed in this work, our predictive models provide accurate and useful predictions as well as useful insight into what behaviors can be correlated and relate to successful outcomes in the context of 6.0001/2. More specifically, these predictions are least useful at the beginning of a course, when there is lack of information available to the model, and at the end of the course when grades are practically already determined. Therefore, as discussed, the most insightful predictions, therefore, are in the middle of the course, and thus these weeks are the most crucial for intervention. As these predictive models are more integrated into the course staff workflow, it is imperative that these predictions are monitored and used to reach out to struggling students during these critical middle weeks of the course.

Finally, we have shown how helpful and effective these grade predictions can be and believe that they can be extended to help improve and understand the learning experience in many contexts. Through this research and these tools developed for predicting and monitoring student grades, we hope to provide a basis for which future courses can improve their understanding of course difficulty, students' retained knowledge of material, and learner behavior.

Appendix A

Tables

Feature	**Brief Description**
kerberos	student institute wide unique identifier
fex_correct	# of correct finger exercise questions
fex_num_attempts	# of attempted finger exercise questions
fex_num_answered	# of finger exercise questions answered
ndays_act	# of days active on MITx
sum_dt	total amount of time away from MITx window
n_dt	# of times away from MITx window in session
nevents	# of total MITx events logged
npause_video	# of times videos were paused on MITx
nproblem_check	# of problems checked on MITx
nseek_video	# of times seeked through MITx videos
nseq_goto	# of times went to a specific point in MITx videos
nshow_answer	# of times answer to an MITx question was shown
nvideo	# of MITx videos interacted with
nvideos_viewed	# of MITx videos watched
nproblems_attempted	# of MITx problems attempted
avg_dt	avg time away from an MITx window in session
sdv_dt	std time away from an MITx window in session

max_dt	max time away from an MITx window in session
pset_avg	average on psets
checkoff_avg	average on checkoffs
quiz_avg	average on quizzes
pset_queue_help	# of times gone onto queue for help
Grade	A-F grade in course
Overall	numerical grade value
awr_friends	student uses friends as a resource
awr_google	student uses Google as a resource
awr_online	student uses internet as a resource
awr_stack_overflow	student uses stack overflow as a resource
exp_None	student has no prior experience
exp_ap_comp_sci	student has taken AP Computer Science
exp_html	student has experience with HTML
exp_non_python	student has experience language with non Python
exp_non_python_college	student has taken a non python coding course
exp_ocw	student has taken OCW version of course
exp_online_class	student has taken an online coding course
exp_python	student has experience with Python
exp_python_college	tudent has taken a python coding course
grade_level	student grade level
lines_of_code	number of lines of code written prior to course
mot_gpa	motivated to take course to boost GPA
mot_learn	motivated to take course to learn
mot_other	motivated to take course by something else
mot_requirement	taking course because it's a requirement
semester	current semester
dep_Employee	student is an employee of the Institute

dep_MaterialsScienceAndEng	major is MaterialsScienceAndEng
dep_ElectricalEngComputerSci	major is ElectricalEngComputerSci
dep_PoliticalScience	major is PoliticalScience
dep_Architecture	major is Architecture
dep_AeroAndAstro	major is AeronauticsAndAstronautics
dep_Biology	major is Biology
dep_Undesignated	major is Undesignated
dep_Unknown	major is Unknown
dep_Economics	major is Economics
dep_Humanities	major is
dep_EarthAtmosPlanetarySci	major is Humanities
dep_ChemicalEngineering	major is ChemicalEngineering
dep_Chemistry	major is Chemistry
dep_Undeclared	major is Undeclared
dep_CivilAndEnvironmentalEng	major is CivilAndEnvironmentalEng
dep_Management	major is Management
dep_BiologicalEngineering	major is BiologicalEngineering
dep_NuclearScienceEngineering	major is NuclearScienceEngineering
dep_Mathematics	major is Mathematics
dep_HealthSciencesTechnology	major is HealthSciencesTechnology
dep_Physics	major is Physics
Wellesley	student is a cross-registered Wellesley student
Harvard	student is a cross-registered Harvard student
dep_UrbanStudiesAndPlanning	major is UrbanStudiesAndPlanning
dep_HumanitiesEngineering	major is UrbanStudiesAndPlanning
dep_MechanicalEngineering	major is MechanicalEngineering
dep_BrainAndCognitiveSciences	major is BrainAndCognitiveSciences
days online	number of days active on piazza

posts	number of piazza posts made
answers	number of answers to piazza questions posted
edits to answers	number of edits to piazza answers posted
followups	number of follow ups made on piazza
replies to followups	number of replies to follow ups made on piazza
avg_time_after_last_submit	avg time of last submit before deadline
avg_workahead_time	avg time of first submit after pset release
avg_submits	avg # times student submits psets

Table A.1: Dataset features with descriptions

Appendix B

Progress Page Documentation

B.1 Website File Structure

Within the website source code, we've added a progress directory which holds all of the code used for both the frontend and backend of the grade prediction pages. This section discusses the relevant files/directories created for this work with the hopes of to helping with set up of the grade prediction tool in future semesters of 6.0001/2. The directory tree (rooted at the `progress` directory) of relevant files and directories is shown in Figure B.1, and the files/directories contain the following:

- **_files**: This directory holds various data files used for the predictions.
  - **gradedata**: Holds csvs of grades assigned at the end of courses used in the datasets. Files are named using the following convention:
    `grades-{semester}-{course#}.csv`
  - **profiles**: Holds profiles of students for previous semesters of the courses. The data for the files are pulled from the website logs. Profiles are as described in Vostatek's paper [9]. Files are named using the following convention:
    `profiles-{semester}.csv`
  - **registrationdata**: Holds registration information for the semesters used in the datasets. Data is pulled from LMOD. Files are named using the

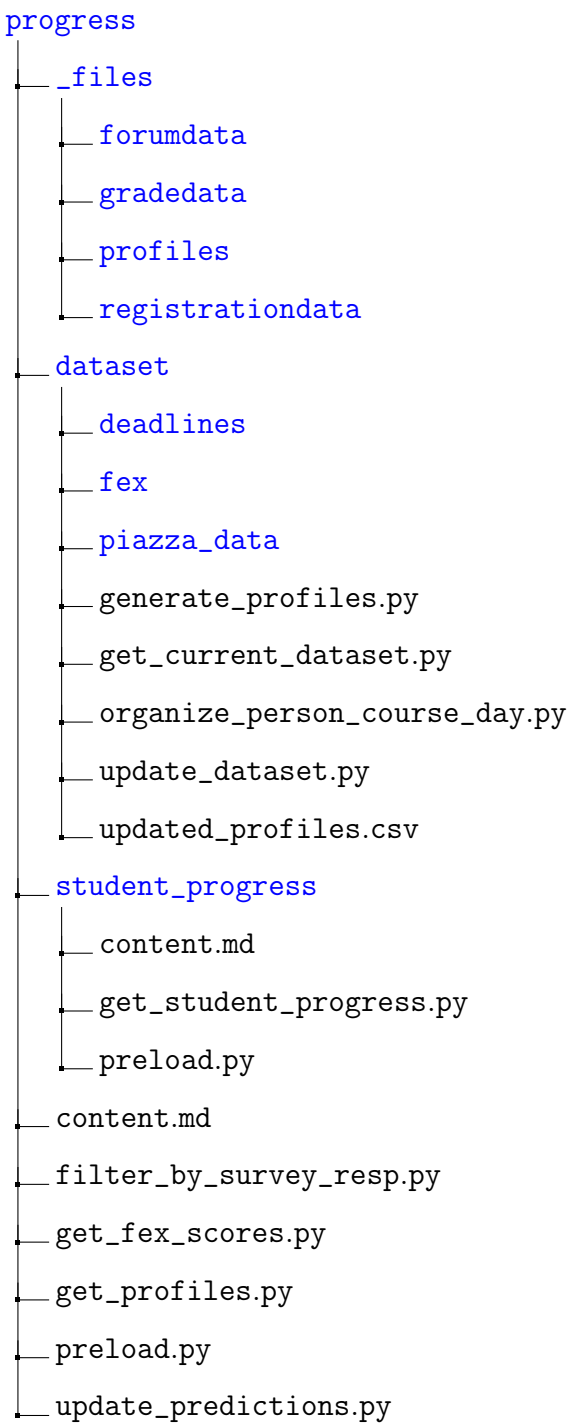


Figure B-1: Condensed file structure of directories and files used to create and visualize grade predictions. Blue indicates a directory and black indicates a file. Only relevant files/directories are shown in figure, however there are other files/directories that are not essential or are superfluous for understanding the grade prediction pipeline.

following convention:

`{semester}_reg_{course#}.csv`

- **dataset**: This directory is for all files pertaining to generating the datasets used in the predictions.
  - **deadlines**: This is a directory to hold the deadline information used when pulling information for each semester (i.e. finding out what files were due by a certain date in a semester). Each semester used in the predictions (including the current semester) must have a file (named `{semester}.py`) that holds all of the deadlines for the semester. All of the deadline data can be pulled from the site source code and must be formatted in the same way. There is also a `generate_deadlines_file.py` file, which is used to generate a json of all deadlines over all semesters which is used by the prediction pipeline.
  - **fex**: This directory contains information pertaining to finger exercises when they were on MITx (before they were moved to the course site) as well other MITx data used by the prediction pipeline. There is a `6.0001` and `6.0002` directory located in this directory. Each holds a file pulled from MITx weekly for that course. They are named with the following convention:

    `6.000{#}r_{#}_person_course_day.csv` where # is 10 for the fa19 semester, 11 for the sp19 semester, 12 for the fa20 semester, and so on.
  - **piazza_data**: Holds data pulled weekly from piazza that is used for predictions. There is also a `6.0001` and `6.0002` directory located in this directory. For each week (1-7) of each course, there is a file for this data in the respective directory based on course. It is named with the following convention:

    `{semester}week{#}.csv`
  - **generate_profiles.py**: Used to generate profiles at the end of the semester for use in predictions in future semesters (especially useful for when the log

for that semester gets cleared). This file uses the `unformatted_profiles.csv` that is also found in the **colorblue dataset** directory.

  - **get_current_dataset.py**: Uses all of the data described above to generate a dataset for the current semester used in grade predictions.
  - **organize_person_course_day.py**: Organizes the `person_course_day` csvs described above so that they information in these csvs can be easily accessed when generating the datasets.
  - **unformatted_profiles.csv**: Holds the unformatted profiles that are pulled directly from course logs. After using `generate_profiles.py`, data available in this file is formatted and stored in **updated_profiles.csv**
  - **update_dataset.py**: Organizes the dataset of all previous semesters (excluding the current semester's data) to be used as the train/test datasets.
  - **updated_profiles.csv**: Holds the formatted profiles used in the datasets discussed above.

- student_progress: Holds the files relevant for the individual student progress pages

  - **content.md**: A markdown file that renders the individual student progress page. Holds all HTML and formatting for the page as well.
  - **get_student_progress.py**: File used to pull all student progress information – i.e. the numbers used to populate the grade trajectory chart.
  - **preload.py**: File for helpers and relevant variables used in the `content.md` file.

- **content.md**: A markdown file that renders the overall student progress page.

- **filter_by_survey_resp.py**: File used to filter the data shown of the overall student progress page based on the dropdowns available on the page.

- **get_profiles.py**: File used to pull profiles for previous iterations of the courses from course site logs.

- **preload.py**: File for helpers and relevant variables used in the `content.md` file.
- **update_predictions.py**: Used to generate the predictions of a given week in a course. Performs the training/testing of ML models and generates the predictions from the datasets created by `get_current_dataset.py` and `update_dataset.py`.

B.2 Updating Weekly Predictions

B.2.1 Beginning of Course Setup

There are a couple of files that need to be created/updated at the start of each course. Once they are finalized, they are used for the rest of the semester and do not need to be updated.

Deadlines File

The first is the deadlines file, which is used to calculate which assignments were due at various points in the semesters used to create our datasets. In order to update this file, once due dates for the courses are finalized, create a file called `{semester}.py` and put it in the `progress/dataset/deadlines` directory. The files must be formatted in the same way in order for the dates to be extrapolated – see `fa20.py` in this directory as an example of how to format and store data in these files. Then run the `progress/dataset/deadlines/generate_deadlines_file.py` locally. Then just commit and push these changes to git. **Note**: if any changes to due dates made throughout the course of either class, update the `{semester}.py`, re-run `progress/dataset/deadlines/generate_deadlines_file.py`, and commit/push these changes so that they're reflected.

Profiles Files

The other files that needs to be generated at the start of each semester are the `unformatted_profiles.py` and the `updated_profiles.py` files. To create them, first add/update the relevant variables in `progress/get_profiles.py` – see docstrings/comments in this file for more information regarding what varibles to add/update

–, then run it on the server by going to the url `https://sicp-s1.mit.edu/{semester}/progress/get_profiles.py`. Then save the response to the `progress/unformatted_profiles.py` file. Finally, locally run the `progress/dataset/update_profiles.py`, which will update the `updated_profiles.py` file, and commit/push these changes.

B.2.2 Weekly Prediction Generation

Updating Piazza Data

To update Piazza data weekly, complete the following steps each Sunday:

1. Go to Statistics → Export Statistics (on the bar at the top of the page)
2. Change the max date to the Sunday of that week and export
3. Save file and rename it `{semester}week{#}.csv`
4. Store file at `progress/dataset/piazza_data/{semester}`
5. Commit and push changes

Updating MITx Data

To update MITx data weekly, complete the following steps:

1. Pull data on BigQuery
2. Rename to `6.000{#}r_{semester #}_person_course_day.csv`
3. Store file at `progress/dataset/fex/6.000{#}`
4. Commit and spush changes

Generating Grade Predictions

After updating the weekly data, generating the grade predictions is as easy as clicking the ***Generate Predictions*** button at the top, left-hand side of the overall progress page. To monitor the status of the predictions (successes/failures of API calls), use your browser's developer tools to view the console after the ***Generate Predictions*** button is pushed. There you will see what calls have been made, any errors, and a log of when the prediction generation process is finished.

Bibliography

[1] Labor Force Statistics from the Current Population Survey. https://www.bls.gov/emp/chart-unemployment-earnings-education.htm. Accessed: 2021-03-31.

[2] MIT Subject Listing & Schedule Advanced Search. http://student.mit.edu/catalog/extsearch.cgi. Accessed: 2020-12-27.

[3] Learn more, earn more: Education leads to higher wages, lower unemployment : Career Outlook. https://www.bls.gov/careeroutlook/2020/data-on-display/education-pays.htm, May 2020. Accessed: 2020-12-27.

[4] Ayesha R. Bajwa. *Analyzing student learning trajectories in an introductory programming MOOC.* 2019.

[5] World Economic Forum. The future of jobs report 2020. World Economic Forum, Geneva, Switzerland, 2020.

[6] Pierre Geurts, Damien Ernst, and Louis Wehenkel. Extremely randomized trees. *Machine learning*, 63(1):3–42, 2006.

[7] Christine Vonder Haar and Ana Bell. Analysis of repeat learners in computer science moocs. In *2020 IEEE Learning With MOOCS (LWMOOCS)*, pages 4–7. IEEE.

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

[9] Vincent Charles Vostatek. *Predicting factors that affect student performance in MOOC and on-campus computer science education.* PhD thesis, Massachusetts Institute of Technology, 2020.

[10] M. Eng Wang, Li. *The influence of grades on learning behavior of students in MOOCs.* 2019.