# When to Check In?: Identifying Early Signs of Student Struggle at Various Cut-Points in CS1 Course Data

Abigail Liu
apliu@udel.edu
University of Delaware
Newark, DE, USA

Sammy Alashoush
samalash@udel.edu
University of Delaware
Newark, DE, USA

Austin Cory Bart
acbart@udel.edu
University of Delaware
Newark, DE, USA

Teomara Rutherford
teomara@udel.edu
University of Delaware
Newark, DE, USA

Nazim Karaca
nazim@udel.edu
University of Delaware
Newark, DE, USA

John Aromando Jr.
jaro@udel.edu
University of Delaware
Newark, DE, USA

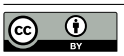
Matthew Louis Mauriello
mlm@udel.edu
University of Delaware
Newark, DE, USA

## Abstract

Researchers have investigated early warning systems through learning analytics (LA) to identify students who need support in computer science (CS) courses. There is debate over which data are best for prediction and how much data are needed. For these warning systems to be useful, it is important to understand at what times LA data features are most effective at identifying students at risk of negative course outcomes. In this paper, we compare 16 common behavioral features found across the LA literature using generalized linear mixed models (GLMMs) to predict students' final course and final exam grades. We aggregate features over 2-week intervals (e.g., at 2 weeks, 4 weeks), examining each feature's ability to predict student performance at different times in the semester. Our findings reveal that a student's accuracy in unit testing problems is a statistically significant predictor across all cut-points when predicting final exam and final course grades. Other predictors are statistically significant at a subset of cut-points. For example, how early a student starts an assignment is statistically significant in earlier weeks of the semester when predicting final exam grades, whereas the number of syntax errors is statistically significant towards the middle of the semester when predicting final course grades. Our work contributes a comprehensive analysis of a large set of LA features while maintaining the temporal context of course progression. We also provide design recommendations for implementing these features in a dashboard to convey course insights to instructors.

## CCS Concepts

• **Applied computing** → *Education*; • **Information systems** → *Data mining*; • **Social and professional topics** → **CS1**.




*ITiCSE 2026, Madrid, Spain*

ACM ISBN 979-8-4007-2634-7/2026/07
https://doi.org/10.1145/3803400.3809327

## Keywords

CS1 course data, CS Education, Early prediction, Educational data mining, Learning analytics, Predicting student performance, Programming course, Struggle identification, Wheel-spinning

**ACM Reference Format:**
Abigail Liu, Sammy Alashoush, Austin Cory Bart, Teomara Rutherford, Nazim Karaca, John Aromando Jr., and Matthew Louis Mauriello. 2026. When to Check In?: Identifying Early Signs of Student Struggle at Various Cut-Points in CS1 Course Data. In *Proceedings of the 31st ACM Conference on Innovation and Technology in Computer Science Education V. 1 (ITiCSE 2026), July 10–15, 2026, Madrid, Spain.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3803400.3809327

## 1 Introduction

According to the 2024 CRA Taulbee Survey [34], introductory computer science (CS) courses have received large amounts of enrollment with a generally increasing trajectory. However, these same courses also face sizable dropout rates of about 25% [38]. These trends have occurred for decades, with some fluctuations over the years [25] and are corroborated by earlier studies [25, 27]. This phenomenon, along with the fact that students do not always ask for help on their own [11, 14, 35], is the motivation for research studying predictors of student struggle and performance (e.g., [4, 17, 24, 28, 32]). This research belongs to the learning analytics (LA) domain, which uses data from students as they complete course content to analyze which behaviors predict performance.

Among these studies, there is a large variability in the data features and predictive methods used, with sets of features measuring across categories such as "Code state contents," "Handling errors," and "Time-on-task." This makes it difficult to compare works. There is also a discrepancy in the amount of data needed to achieve an accurate prediction. Most works tend to choose cut-points, i.e., time boundaries at which cumulative student data is evaluated, based on the percentage of course materials completed or on the course's context. One example is Adnan et al. [1], who used machine learning and deep learning approaches to predict final exam scores. They used cut-points in 20% intervals, predicting scores at 20% and 40%

course completion, and so on. Another example is Llanos et al. [22], who predicted course performance at weeks 3, 5, and 7 because specific lab assignments were scheduled in those weeks. In addition, most works use small datasets and test their methods on only a few course sections (e.g., [4, 5, 32]). This makes it difficult to tell if the data features are generalizable across different student samples.

In this work, we explore how well various data features predict student performance at different times in the semester. Using a dataset containing 30 course sections and 16 data features aligned with prior research (defined in Table 1), we answer the following research questions: *(RQ1) Can we predict student performance using common behavioral data features from LA literature? (RQ2) Which data features are most predictive at different times in the semester?*

Our work provides two main contributions. First, we analyze a large set of data features from the current LA literature and compare them across a single, large dataset. It is rare to have a comprehensive evaluation with data features from multiple works. By evaluating these on the same dataset, we can report results on a single sample of students, enabling a more objective comparison of features. Compared with other datasets, ours contains more information due to its size and granularity, comprising data from over 1,100 students and 2,206,507 student submissions. Second, our aggregation methodology allows us to incorporate the temporal context of course progression in our analysis. By aggregating data features at different cut-points throughout the semester, we observe how some features become more or less predictive over time. These contributions work together to provide a comprehensive and contextual analysis of common LA data features, and our findings can be extended towards developing analytical tools to help CS1 instructors identify early signs of student struggle.

## 2 Related Work

### 2.1 Behavioral Data Features

Educational data mining researchers increasingly use action log data to predict student performance [16]. These logs are a series of actions recorded as a student completes course material content, often in systems such as learning management systems (LMS) or version control systems (VCS). The granularity of the data, as well as the aspects recorded, varies across datasets. For instance, Arakawa et al. [4] collected data from 312 students using GitHub VCS, recording each time a student pushed a version of code to their repository. Other works, such as Pinto et al. [28], used the PrairieLearn LMS to collect data from 733 students, tracking progress on programming homework, weekly quizzes, and cumulative exams.

The proposed data features used in each work vary greatly. Researchers hypothesized student behaviors or patterns that may be related to student struggle and performance, and then designed features that operationalize these behaviors. Based on our literature review, these features can be grouped into 13 categories. Table 1 lists these categories along with their descriptions. Some of these categories may appear to be the same. We highlight the differences between the two pairs of closely related categories below:

In our definitions, Effort focuses on the number of times an action was done. For these features, some action is needed to complete the assignment, but having too many or few could signal giving too much or little effort. This category has features like "Number of submissions," where students must submit a certain number to complete assignments. However, having too many submissions could mean struggle, whereas too few could be a lack of engagement.

In previous works, Struggle focuses on features with author-defined criteria for struggle. These are often binary features indicating whether a student meets the criteria. For example, Alam et al. [2] labeled a student as struggling with a particular coding problem if they attempted it but never got it correct, or if they got it correct with more submissions than 75% of students.

Like Effort/Struggle separation, Time-on-task focuses on data features tracking the amount of time spent working. The difference between each Time-on-task feature is how they defined when a student is spending time on an assignment. Some prior work used only times when the student is actively typing [18], whereas others used the intervals between logged submission timestamps [28].

The Engagement category operationalizes how active or focused students were when working on an assignment. These features may be related to time, but focus on whether the student engaged in an action that indicated losing focus. This includes binary features like "Overly Idle" [37], which labeled the student as idle if they stop coding for more than 5 minutes, and "Interleaving" [28], which checked if the student switches to another assignment before returning to the original one during the same login period.

### 2.2 Prediction Models

Many prior works used accuracy measures, such as unit testing ratios [4, 32] or final course grades [40], as data labels for their prediction model. The analytical methods in LA can be applied to two types of problems: regression and classification. In regression, the model directly predicts the measurement through regression or machine learning models. For instance, Araya et al. [5] used data features focused on student behaviors to predict each student's average assignment grades, average midterm exam grades, and final course grade using random forests, support vector regression, and linear regression. In classification, students are sorted into groups depending on their performance. For example, Llanos et al. [22] used each student's final course grade to label them as low-, medium-, or high-performing, training machine learning models to predict which group each student belonged to based on their performance in lab assignments and exams. In this work, we compared 16 features from our literature review, using generalized linear mixed models to classify students based on final exam and course grade outcomes.

## 3 Methodology

### 3.1 Data Acquisition

The Evidence of Learning (EOL) dataset comprises two distinct courses at a Mid-Atlantic US R1 University (data collected and anonymized under our approved IRB Protocol #1478144-7). Both courses follow the PythonBakery curriculum,[1] a 15-week introductory Python course for undergraduate students. The curriculum was created for the Canvas LMS, and its materials are publicly available. It incorporates open source learning technologies, such as BlockPy [6] and Pedal.[2] BlockPy is a web-based Python environment designed for beginner programmers. It features Canvas

[1] https://python-bakery.github.io/
[2] https://pedal-edu.github.io/pedal/

| Meta-Category | Description |
|---|---|
| Student Info | Course/Demographic information about students [22] |
| Student Prep | Students' prior knowledge and skills [19] |
| Achieving Mastery | Accuracy measures indicating that a student is learning. Includes unit tests, grades, correct/incorrect submissions, if the student eventually got the correct answer [2, 4, 5, 15, 18, 22, 26, 28, 39] |
| Code State Contents | Contents of a particular code state, not looking at its relationship to other code states [2, 5, 21] |
| Course/System Info | Information about the course or learning system [22, 26, 39] |
| Changes in Code States | What/how much a student is changing in their code [4, 15, 21, 37] |
| Difficulty of Task | Related to how challenging the assignment is. Includes student perceptions of how hard a piece of code is to implement [2] |
| Effort | Related to how many attempts a student used to complete an assignment [2, 4, 5, 8, 13, 15, 22, 26, 28, 37, 39] |
| Engagement | Related to how active a student is in completing an assignment, usually about how often/consistent students work [5, 15, 21, 28, 37] |
| Handling Errors | Related to resolving errors in code (e.g., syntax and runtime errors) [4, 5, 15] |
| Procrastinating | Related to when a student works on an assignment [4, 5] |
| Struggle | Indicators of students getting stuck. Includes wheel-spinning [2, 4, 28, 37] |
| Time-on-Task | Related to the amount of time a student works on an assignment [2, 5, 15, 18, 22, 28, 28] |

**Table 1: Meta-categories for data features. The descriptions and works that use features from each are listed.**

integration, allowing students to write, compile, and submit coding assignments directly on the Canvas website. With BlockPy and Canvas, we unobtrusively log each student's actions while they complete assignments. Pedal is a framework for analyzing student code. Instructors can create scripts that automatically provide feedback on student code based on the code's context. Through Pedal, instructors set up unit tests to evaluate a student's code. We used this unit-testing feature in our data. All data used in this study are saved on our BlockPy server.

## 3.2 Dataset Contents

The EOL dataset contains a total of 2,206,507 student submissions with 15,043,100 unique code states. Each time a student compiles code for an assignment, we consider this as a new submission. Courses were held between the Fall 2022 and Spring 2024 semesters across 30 sections. 21 (15 non-Honors) sections were for non-CS engineering majors (CISC-X). 9 (8 non-Honors) sections were for CS majors (CISC-Y). The EOL dataset contains 1,189 students, including 853 (730 non-Honors) and 336 (306 non-Honors) from CISC-X and CISC-Y, respectively. The courses had largely the same content. Both courses are scheduled separately, with their own Honors sections, and the bulk of students are in the non-Honors sections.

The PythonBakery curriculum consists of coding assignments, projects, and exams. For this analysis, we focus on coding assignments. An assignment comprises a set of problems grouped by topic. Some problems have unit testing (test cases set by the instructor to assess student code). For this study, we focus on the 14 main assignments, which include fundamental coding topics, such as If statements, For loops, and data structures. As the semester progresses, course materials build on previously learned concepts and become increasingly more complex. In the context of our data, we focus on coding assignments because they occur regularly throughout the semester, capturing more nuance in how a student progresses through the course. Additionally, student behavior on these assignments may reveal signs of struggle much earlier than on exams or projects scheduled later in the course. Each assignment has an available, due, and lock date on Canvas. Through BlockPy, students edit and submit their code directly on the Canvas webpage. Student actions, such as starting an assignment, editing a file, and submitting a code version, are logged. Metadata, such as the timestamp of when the action occurred, the feedback provided by Pedal, and the assignment in which the action occurred, is also saved. Each week, two assignments (parts A/B) are made available to students, both covering the same topic. For the first 8 weeks of the course, the assignment topics and the number of problems in each are as follows- 1: Intro (25/39); 2: Functions (28/21); 3: If Statements (6/15); 4: Data Structures (21/21); 6: For Loops (14/12); 7: Sequences (12/11); 8: Nesting (11/9). Week 5 had a midterm exam, so no assignments were given.

The dataset follows the ProgSnap2 data format [29], which features tables stored in an SQL database. The main table is an event log that records the events (or actions) each student performs in the BlockPy environment. Each record includes the event's timestamp, the event type, and any Pedal feedback triggered.

## 3.3 Feature Selection

Based on prior research, we derived data features that measure various student behaviors and patterns. We provided meta-categories for each feature, indicating what each is intended to measure. We selected a subset of features identified in prior literature as predictors of student struggle and performance, based on viability with our dataset. Table 2 lists these features with their associated meta-categories and descriptions. Note that some of the descriptions and calculations were modified to fit our dataset.

Some of the data features in our models are not actionable predictors. Instead, they are used to contextualize model predictions. As seen in Table 2, whether a student is a CS major or in an Honors section influences which course section they attend. Further, whether a student took the course in the fall or spring semester has contextual nuance. Students typically take CS1 courses in the fall, because taking it later impacts eligibility to enroll in subsequent courses. So, students in the spring semester are "behind" those taking it in the fall. These three data features are considered student demographics, based on enrollment rather than identity traits such as gender or race. They allow us to control for the different sizes, needs, and capabilities of each course section in our dataset.

### 3.4 Outcomes

We tried two outcomes: the final course grade and the final exam grade, since both were available. Both have been used in several works as outcomes [2, 5, 15, 18, 20, 22, 39, 40]. We mapped actual values to binary labels at a threshold of 80%, based on the consensus of co-authors who were also course instructors for CISC-X/Y: students who get a C (<80) do poorly in subsequent courses.

### 3.5 Data Preprocessing

We excluded students who did not complete work after the third week of the semester and did not take the final exam, because they have very little data and may have dropped the course during the add/drop period. Seven met this criterion, leaving us with 1,182 students. Students who did not complete the final exam but completed work after the third week had their final exam grade mapped to zero. Of the 1,182 students, 881 and 868 were $\geq 80$ for our final exam grade and final course grade outcomes, respectively.

In this study, we represented each student as a set of data feature values. To capture how these feature values change over a semester, we aggregated them across the assignments made available to students so far. Table 2 provides details on how each feature was calculated. Those with an asterisk (*) indicate that a weighted average was used to aggregate. Those without contain the aggregated calculation in their description. We aggregated data in 2-week intervals, resulting in 2-, 4-, 6-, and 8-week views of the semester.

### 3.6 Model Choice

We used generalized linear mixed models (GLMMs), similar to models used in prior research [5, 18, 21, 28, 37, 39]. Because GLMMs are statistical models, we focused on the logistic regression coefficients and $p$-values to assess the relationships between data features and outcomes. Moreover, we focused on deviance scores to assess how well our models fit the data, where lower values indicate a better fit [23]. The 16 data features are the fixed effects, and the course section of each student is the random effect. We fitted a GLMM for 2, 4, 6, and 8 weeks of the semester for each of our two outcomes, yielding a total of 8 GLMMs. Because several GLMMs were used, we may have encountered statistically significant results from random chance. To account for the false discovery rate, we applied the Benjamini-Hochberg procedure [7]. From this, we see the predictive power of each data feature and how this changes over time.

## 4 Results

### 4.1 Predicting Final Exam Grades

Table 3 shows the standardized logistic regression coefficients for each model predicting final exam grades. Values marked with an asterisk (*) indicate $p$-values < .05 after applying the Bejamini-Hochberg procedure. A student's accuracy in unit test problems and the amount of syntax errors they made were statistically significant predictors across all cut-points. Compared to the other features, these had larger coefficient magnitudes. Other data features were statistically significant at a subset of cut-points. How early a student started an assignment and how many missing assignments were statistically significant only after 2 and 4 weeks, not later. Conversely, whether a student experienced wheel-spinning was only a statistically significant predictor after week 8. Additionally, deviance scores decreased as we include more semester data, indicating that more data improves the model's fit.

### 4.2 Predicting Final Course Grades

Table 4 presents the standardized logistic regression coefficients for each of our models predicting final course grades. Values marked with an asterisk (*) indicate that its $p$-value <.05 after applying the Bejamini-Hochberg procedure. A student's accuracy in unit test problems, how early a student started an assignment, and how many missing assignments were statistically significant across all cut-points. Besides the number of syntax errors, these features had the largest coefficient magnitudes among the data features. The amount of syntax errors, although not statistically significant after 2 weeks, was statistically significant after weeks 4, 6, and 8. Whether a student got 100% on an assignment and the number of coding sessions was statistically significant after weeks 2 and 4, but not after 6 and 8. We see that including more semester data decreased the deviance score, allowing the model to better fit the data. Compared to the deviance scores in Table 3, these scores are smaller. This means that the models predicting final course grades fit the data better than those predicting final exam grades.

## 5 Discussion

Using final exam and course grades outcomes, we found a student's accuracy in unit tests was a statistically significant predictor across all cut-points. Other features, such as the number of syntax errors, when they started an assignment, and how many missing assignments were statistically significant predictors at certain parts of the semester. These results indicate we can predict student performance with common behavioral data features from LA literature (RQ1). From our features, we identified which are most predictive at different times in the semester (RQ2).

### 5.1 Temporal Course Context

During the semester, the complexity of course materials increases. The first assignments cover isolated core concepts, whereas the later assignments are more complex and build on previously learned concepts. Likewise, students' coding behaviors may also change across the semester. These changes are reflected in the change in statistical significance for data features. For instance, when predicting final

| Meta-Categories | Feature Name | Description | Range |
|---|---|---|---|
| Student Info | Semester | Which semester a student's course was in (0 for fall, 1 for spring) [22] | [0,1] |
| Student Info | IsHonors | If a student was in a non-Honors (0) or Honors (1) course section [22] | [0,1] |
| Student Info | IsCSMajor | If a student was a non-CS major (0) or CS major (1) [22] | [0,1] |
| Student Prep | PretestScore | Score student received for the pretest to the course (4 indicates 100%) [19] | [0,4] |
| Achieving Mastery | Solved* | If a student got 100% on an Assignment before the lock date. (0 if no, 1 if yes) [2, 28] | [0,1] |
| Achieving Mastery | UnitTestRatio* | [# of passed unit tests]/[total # of unit tests] in an Assignment [4, 39] | [0,1] |
| Effort | NumSubmissions | [# of submissions]/[total # of problems after X weeks] [2, 5, 8, 13, 22, 26, 28, 39] | [0, ∞] |
| Engagement, Effort | NumSessions | [# of coding sessions]/[# of Assignments after X weeks] A coding session ends when no action is done after 15 minutes [28] | [0, ∞] |
| Engagement, Effort | Missing Assignments | [# of unattempted Assignments] /[# of Assignments after X weeks] [15] | [0, 1] |
| Handling Errors, Effort | SyntaxErrorRatio* | [# of syntax error messages]/[total # of feedback messages] in an Assignment [4, 5] | [0, 1] |
| Procrastinating | NormStartTime* | Normalized timestamp of the first submission of an Assignment. 0, 0.5, and 1 are when an Assignment is available, due, and locked, respectively [4] | [0, 1] |
| Struggle | NumRapidGuesses | [# of rapid guesses]/[# of Assignments after X weeks] A rapid guess is when ≥ 3 submissions are made in a row in < 15 seconds [28] | [0, ∞] |
| Struggle | WheelSpinning* | If a student attempted the Assignment but never got 100%, or got 100% with more submissions than 75% of other students. (0 if no, 1 if yes) [2] | [0, 1] |
| Struggle | NumExcessRuns | [# of times code was compiled multiple times without any changes] /[# of Assignments after X weeks] [37] | [0, ∞] |
| Struggle | NumFeedback Unchanged | [# of submissions where the feedback message did not change] /[# of Assignments after X weeks] [4] | [0, ∞] |
| Time-on-Task | TotalTime SpentCoding | [# seconds spent coding]/[# of Assignments after X weeks] from latencies between keystrokes excluding breaks over 15 minutes [18] | [0, ∞] |

**Table 2: Data features used in this work. The description, citations using similar metrics, and range are provided. * indicates that a feature was aggregated using weighted average. Weights were determined by the number of problems in each assignment.**

course grades, the amount of syntax errors is not statistically significant after the first 2 weeks of the semester, but becomes statistically significant after weeks 4, 6, and 8. This can be interpreted in light of students' growing familiarity with the coding language. At the beginning, syntax errors are common among all students because they were only recently introduced to the coding language and its syntax. However, later in the semester, students are expected to be well-versed in the syntax. Students with a high number of syntax errors later in the semester are more concerning than those at the beginning. Likewise, when predicting final exam grades, whether a student experiences wheel-spinning is not a statistically significant predictor after weeks 2, 4, and 6, but becomes statistically significant after week 8. Wheel-spinning may be more common earlier in the semester and is less concerning than later in the course.

Another trend that may have affected our results is students may attempt fewer assignments later in the semester. This could be due to several factors, such as balancing effort between courses or losing motivation [31]. This decrease in attempts affects effort-based data features. For instance, if a student did not attempt an assignment, the number of syntax errors, rapid guesses, wheel-spinning, excess runs, and unchanged feedback values for that assignment was 0. Meanwhile, a student who attempted that assignment perfectly on the first attempt also received a 0. These different scenarios result in the same values for the listed features. Although using the number of missing assignments as a feature could mitigate this by identifying if a student attempted an assignment, these features may have affected the models' fit. We kept their descriptions and calculations as they were operationalized in prior work. In future work, we may redesign the features to better align with our data.

## 5.2 Design Recommendations

The data features used in this study can be easily calculated and implemented into a dashboard for instructors to observe how students complete assignments. By identifying patterns and early warning signs within student behavior, instructors can intervene in a timely manner. By providing insights, instructors can make more informed decisions when designing student-centered course materials [10]. We will explore the demand and feasibility of this in future work.

## 5.3 Limitations and Future Work

We focused our analysis on the 14 assignments covering the main topics of our CS1 courses. We did not include other course materials, such as midterm exams, projects, and assignments covering more advanced topics. We plan to incorporate these in future experiments. We also only tested cut-points at 2-week intervals. Using smaller intervals may provide more details on student behavior, potentially

| | Standardized Logistic Regression Coefficients "After X weeks" | | | |
|---|---|---|---|---|
| **Data Feature** | 2 | 4 | 6 | 8 |
| Semester | -0.27* | -0.24* | -0.24* | -0.23 |
| IsHonors | 1.38* | 1.41* | 1.42* | 1.41* |
| IsCSMajor | 0.32* | 0.30* | 0.30* | 0.29* |
| PretestScore | 0.08 | 0.08 | 0.07 | 0.11 |
| Solved | 0.55 | -0.06 | -0.45 | -0.92 |
| UnitTestRatio | 2.56* | 4.14* | 4.56* | 4.17* |
| NumSubmissions | >-0.01 | 0.03 | 0.05 | 0.06 |
| NumSessions | >-0.001 | >-0.001 | >-0.001 | >-0.001 |
| SyntaxErrorRatio | -23.40* | -23.61* | -33.70* | -46.69* |
| NormStartTime | -2.79* | -2.12* | -1.86 | -1.30 |
| NumRapidGuesses | -0.01 | >-0.01 | -0.01 | -0.02 |
| WheelSpinning | -0.63 | -0.870 | -1.31 | -2.26* |
| NumExcessRuns | -0.02* | -0.03 | -0.02 | -0.04 |
| NumFeedback Unchanged | <0.001* | <0.001 | <0.001 | <0.001 |
| TotalTime SpentCoding | <0.001* | <0.001 | <0.001 | <0.001 |
| MissingAssignments | 3.99* | 3.97* | 3.27 | 1.27 |
| **Deviance Score** | 1075.70 | 1038.32 | 1023.34 | 1003.68 |

**Table 3: Standardized logistic regression coefficients, for the final exam grade outcome. * indicates *p*-value <.05 after applying the Benjamini-Hochberg procedure.**

| | Standardized Logistic Regression Coefficients "After X weeks" | | | |
|---|---|---|---|---|
| **Data Feature** | 2 | 4 | 6 | 8 |
| Semester | -0.46* | -0.55* | -0.58* | -0.50* |
| IsHonors | 2.16* | 2.28* | 2.27* | 2.12* |
| IsCSMajor | 0.62* | 0.74* | 0.77* | 0.73* |
| PretestScore | 0.04 | 0.02 | 0.02 | 0.04 |
| Solved | 1.93* | 1.79* | 1.51 | 1.11 |
| UnitTestRatio | 2.78* | 5.62* | 6.47* | 6.45* |
| NumSubmissions | 0.02 | 0.08* | 0.11* | 0.12* |
| NumSessions | >-0.001* | >-0.001* | >-0.001 | >-0.001 |
| SyntaxErrorRatio | -13.16 | -19.19* | -27.80* | -38.57* |
| NormStartTime | -6.89* | -7.08* | -7.00* | -5.86* |
| NumRapidGuesses | -0.01 | -0.01 | -0.02 | -0.02 |
| WheelSpinning | -0.25 | -0.24 | -0.81 | -1.74 |
| NumExcessRuns | -0.01 | -0.02 | -0.02 | -0.04 |
| NumFeedback Unchanged | <0.001 | >-0.001 | >-0.001 | >-0.001 |
| TotalTime SpentCoding | <0.001 | <0.001 | <0.001 | <0.001 |
| MissingAssignments | 8.04* | 10.93* | 11.40* | 7.42* |
| **Deviance Score** | 978.42 | 880.68 | 854.56 | 817.04 |

**Table 4: Standardized logistic regression coefficients, for the final course grade outcome. * indicates *p*-value <.05 after applying the Benjamini-Hochberg procedure.**

helping our models make better predictions. Our action log data is from one institution and two CS1 courses, which may affect how generalizable our findings are in other contexts. In addition, our dataset included various features related to student information, such as the semester a course was held, whether a student was in an Honors section, and whether they were a CS major. We used them in the same model and analysis to account for differences across course sections in our dataset, but it would be interesting to test whether some features are better predictors for certain groups of students. As previously mentioned, the decrease in attempts affects effort-based data features, where those who do not attempt an assignment at all and those who complete an assignment perfectly on their first attempt both receive values of 0 for those features. We may redesign these to better fit our data in future studies.

In future work, we may explore what kinds of interventions could be effective at different cut-points. We will explore the relationship between these data features and self-regulated learning (SRL), a self-directed process where students convert mental ability into academic skill [41]. Prior studies show that self-regulation is a key skill for achievement in STEM courses [30], and interventions focused on SRL can increase student motivation [12], which can improve course performance [31, 33]. Because action log data and SRL theories complement each other [36], we believe they can be used together with our chosen features to predict course outcomes.

We plan to explore the demand and feasibility of an instructor dashboard, informed by the data features from this work. Previous studies have created visualizations and user interfaces to present LA [3]. A recent study revealed that LA can inform instructors' educational design decisions [9], but there is a need for greater integration of LA into educational practice.

## 6 Conclusion

In this work, we used 16 common behavioral data features from LA literature to explore early signs of student struggle in an introductory CS course. Our data consisted of coding assignments from the first 8 weeks of a semester, covering 7 introductory coding topics for over 1,100 student samples. Using cut-points at 2-week intervals, we used varying amounts of data to predict students' final exam and course grades, examining how predictive the data features were at different points in the semester. For the final exam and course grade, student accuracy on unit test problems was a statistically significant predictor across all cut-points. The statistical significance of other features changed as the semester progressed. This may be due to the level of complexity in the course materials and/or the context of a feature at that time in the semester. Our work provides a comprehensive evaluation of a large set of LA data features that incorporates the temporal context of course progression into our analysis. In future work, further exploration of the connection between student performance and motivation is needed.

## Acknowledgments

This material is based upon work supported by the National Science Foundation under Grant No. 2418739. We would also like to thank the University of Delaware's Undergraduate Research Program for funding Sammy Alashoush via Winter Fellows Program.

## References


[1] Muhammad Adnan, Asad Habib, Jawad Ashraf, Shafaq Mussadiq, Arsalan Ali Raza, Muhammad Abid, Maryam Bashir, and Sana Ullah Khan. 2021. Predicting at-risk students at different percentages of course length for early intervention using machine learning models. *Ieee Access* 9 (2021), 7519–7539.

[2] N Alam, H Acosta, K Gao, and B Mostafavi. 2022. Early prediction of student performance in a programming class using prior code submissions and metadata. In *Proceedings of the 6th Educational Data Mining in Computer Science Education (CSEDM) Workshop, pages*.

[3] Riordan Alfredo, Vanessa Echeverria, Yueqiao Jin, Zachari Swiecki, Dragan Gašević, and Roberto Martinez-Maldonado. 2024. SLADE: A method for designing human-centred learning analytics systems. In *Proceedings of the 14th learning analytics and knowledge conference*. 24–34.

[4] Kai Arakawa, Qiang Hao, Wesley Deneke, Indie Cowan, Steven Wolfman, and Abigayle Peterson. 2022. Early identification of student struggles at the topic level using context-agnostic features. In *Proceedings of the 53rd ACM Technical Symposium on Computer Science Education-Volume 1*. 147–153.

[5] Ignacio Araya, Victor Beas, Katrina Stamulis, and Héctor Allende-Cid. 2022. Predicting student performance in computing courses based on programming behavior. *Computer applications in engineering education* 30, 4 (2022), 1264–1276.

[6] Austin Cory Bart, Javier Tibau, Eli Tilevich, Clifford A Shaffer, and Dennis Kafura. 2017. Blockpy: An open access data-science environment for introductory programmers. *Computer* 50, 5 (2017), 18–26.

[7] Yoav Benjamini and Yosef Hochberg. 1995. Controlling the false discovery rate: a practical and powerful approach to multiple testing. *Journal of the Royal statistical society: series B (Methodological)* 57, 1 (1995), 289–300.

[8] Cheng-Yu Chung, Yancy Vance M Paredes, Mohammed Alzaid, Kushal Reddy Papakannu, and I-Han Hsiao. 2020. A Longitudinal Study on Student Persistence in Programming Self-assessments.. In *CSEDM@ EDM*.

[9] Alrike Claassen, Negin Mirriahi, Vitomir Kovanović, and Shane Dawson. 2025. From data to design: Integrating learning analytics into educational design for effective decision-making: From Data to Design. In *Proceedings of the 15th International Learning Analytics and Knowledge Conference*. 558–567.

[10] Blaženka Divjak, Abhinava Barthakur, Vitomir Kovanović, and Barbi Svetec. 2025. The Impact of Learning Design on the Mastery of Learning Outcomes in Higher Education. In *Proceedings of the 15th International Learning Analytics and Knowledge Conference*. 726–737.

[11] Augie Doebling and Ayaan M Kazerouni. 2021. Patterns of academic help-seeking in undergraduate computing students. In *Proceedings of the 21st Koli Calling International Conference on Computing Education Research*. 1–10.

[12] Laura Dörrenbächer and Franziska Perels. 2016. More is more? Evaluation of interventions to foster self-regulated learning in college. *International journal of educational research* 78 (2016), 50–65.

[13] Max Fowler, Binglin Chen, Matthew West, and Craig B Zilles. 2021. How productive are homework and elective practice? Applying a post hoc modeling of student knowledge in a large, introductory computing course (Full Paper).. In *EDM (Workshops)*.

[14] Qiang Hao, Brad Barnes, Robert Maribe Branch, and Ewan Wright. 2017. Predicting computer science students' online help-seeking tendencies. *Knowledge Management & E-Learning: An International Journal* 9, 1 (2017), 19.

[15] Muntasir Hoq, Peter Brusilovsky, and Bita Akram. 2023. Analysis of an explainable student performance prediction model in an introductory programming course. In *the 16th International Conference on Educational Data Mining (EDM 2023)*.

[16] Petri Ihantola, Arto Vihavainen, Alireza Ahadi, Matthew Butler, Jürgen Börstler, Stephen H Edwards, Essi Isohanni, Ari Korhonen, Andrew Petersen, Kelly Rivers, et al. 2015. Educational data mining and learning analytics in programming: Literature review and case studies. *Proceedings of the 2015 ITiCSE on working group reports* (2015), 41–63.

[17] Ayaan M Kazerouni, Stephen H Edwards, and Clifford A Shaffer. 2017. Quantifying incremental development practices and their relationship to procrastination. In *Proceedings of the 2017 ACM Conference on International Computing Education Research*. 191–199.

[18] Juho Leinonen, Francisco Enrique Vicente Castro, and Arto Hellas. 2022. Time-on-task metrics for predicting performance. *ACM Inroads* 13, 2 (2022), 42–49.

[19] Juho Leinonen, Krista Longi, Arto Klami, and Arto Vihavainen. 2016. Automatic inference of programming performance and experience from typing patterns. In *Proceedings of the 47th ACM technical symposium on computing science education*. 132–137.

[20] Soohyun Nam Liao, Daniel Zingaro, Michael A Laurenzano, William G Griswold, and Leo Porter. 2016. Lightweight, early identification of at-risk CS1 students. In *Proceedings of the 2016 acm conference on international computing education research*. 123–131.

[21] Qianhui Liu and Luc Paquette. 2024. Applying Sequence Analysis to Understand the Debugging Process of Novice Programmers. In *CEUR Workshop Proceedings*, Vol. 3796. CEUR-WS.

[22] Jose Llanos, Víctor A Bucheli, and Felipe Restrepo-Calle. 2023. Early prediction of student performance in CS1 programming courses. *PeerJ Computer Science* 9 (2023), e1655.

[23] P. McCullagh and J.A. Nelder. 1989. *Generalized Linear Models, Second Edition*. Chapman & Hall. http://books.google.com/books?id=h9kFH2_FfBkC

[24] Abdallah Namoun and Abdullah Alshanqiti. 2020. Predicting student performance using data mining and learning analytics techniques: A systematic literature review. *Applied Sciences* 11, 1 (2020), 237.

[25] National Academies of Sciences, Medicine, Division on Engineering, Physical Sciences, Computer Science, Telecommunications Board, Global Affairs, Board on Higher Education, and Committee on the Growth of Computer Science Undergraduate Enrollments. 2018. *Assessing and responding to the growth of computer science undergraduate enrollments*. National Academies Press.

[26] Poorvaja Penmetsa. 2021. *Investigate effectiveness of code features in knowledge tracing task on novice programming course*. North Carolina State University.

[27] Andrew Petersen, Michelle Craig, Jennifer Campbell, and Anya Tafliovich. 2016. Revisiting why students drop CS1. In *Proceedings of the 16th Koli Calling International Conference on Computing Education Research*. 71–80.

[28] Juan D Pinto, Yingbin Zhang, Luc Paquette, and Aysa Xuemo Fan. 2021. Investigating elements of student persistence in an introductory computer science course. In *5th Educational Data Mining in Computer Science Education (CSEDM) Workshop*.

[29] Thomas W Price, David Hovemeyer, Kelly Rivers, Ge Gao, Austin Cory Bart, Ayaan M Kazerouni, Brett A Becker, Andrew Petersen, Luke Gusukuma, Stephen H Edwards, et al. 2020. Progsnap2: A flexible format for programming process data. In *Proceedings of the 2020 ACM Conference on Innovation and Technology in Computer Science Education*. 356–362.

[30] Fernando Rodriguez, Hye Rin Lee, Teomara Rutherford, Christian Fischer, Eric Potma, and Mark Warschauer. 2021. Using clickstream data mining techniques to understand and support first-generation college students in an online chemistry course. In *LAK21: 11th international learning analytics and knowledge conference*. 313–322.

[31] Teomara Rutherford, Hye Rin Lee, Austin Cory Bart, Andrew Rodrigues, and Megan Englert. 2024. How self-beliefs, values, and belonging change and relate with performance during introductory computer science. *Computer Science Education* (2024), 1–37.

[32] Nils Rys-Recker and Qiang Hao. 2024. Early identification of struggling students in large computer science courses: A replication study. In *2024 IEEE 48th Annual Computers, Software, and Applications Conference (COMPSAC)*. IEEE, 88–93.

[33] Mihaela Sabin, Amruth N Kumar, Bonnie K MacKellar, Renée McCauley, Tammy VanDeGrift, and Stephanos Matsumoto. 2025. The Self-Directed Disposition: What Computing Students Say. In *Proceedings of the 30th ACM Conference on Innovation and Technology in Computer Science Education V. 1*. 107–113.

[34] Katie Siek and Jasmine Batten. 2024. Taulbee Survey 2024: Annual Report. *Computing Research Association* (2024).

[35] David H Smith IV, Qiang Hao, Vanessa Dennen, Michail Tsikerdekis, Bradly Barnes, Lilu Martin, and Nathan Tresham. 2020. Towards Understanding Online Question & Answer Interactions and their effects on student performance in large-scale STEM classes. *International Journal of Educational Technology in Higher Education* 17 (2020), 1–15.

[36] Yige Song, Eduardo Oliveira, Paula De Barba, Michael Kirley, and Pauline Thompson. 2025. Investigating validity and generalisability in trace-based measurement of self-regulated learning: A multidisciplinary study. In *Proceedings of the 15th International Learning Analytics and Knowledge Conference*. 339–350.

[37] B Tabarsi, Ally Limke, Heidi Reichert, Rachel Qualls, Thomas Price, Chris Martens, and Tiffany Barnes. 2022. How to catch novice programmers' struggles: Detecting moments of struggle in open-ended block-based programming projects using trace log data. In *Proceedings of the 6th Educational Data Mining in Computer Science Education (CSEDM) Workshop, Virtual*.

[38] Jodi L Tims, Cindy Tucker, Mark A Weiss, and Stuart Zweben. 2023. Computing Enrollment and Retention: Results from the 2021–22 Undergraduate Enrollment Cohort. *ACM Inroads* 14, 4 (2023), 24–43.

[39] Spencer Yoder, Muntasir Hoq, Peter Brusilovsky, and Bita Akram. 2022. Exploring sequential code embeddings for predicting student success in an introductory programming course. In *6th Educational Data Mining in Computer Science Education (CSEDM) Workshop, pages*.

[40] Amelia Zafra and SebastiáN Ventura. 2012. Multi-instance genetic programming for predicting student performance in web based educational environments. *Applied Soft Computing* 12, 8 (2012), 2693–2706.

[41] Barry J Zimmerman. 2002. Becoming a self-regulated learner: An overview. *Theory into practice* 41, 2 (2002), 64–70.