Lightweight, Early Identification of At-Risk CS1 Students

Soohyun Nam Liao[1], Daniel Zingaro[2], Michael A. Laurenzano[3], William G. Griswold[1], Leo Porter[1]

[1]University of California, San Diego

[2]University of Toronto, Mississauga

[3]University of Michigan

ABSTRACT

Being able to identify low-performing students early in the term may help instructors intervene or differently allocate course resources. Prior work in CS1 has demonstrated that clicker correctness in Peer Instruction courses correlates with exam outcomes and, separately, that machine learning models can be built based on early-term programming assessments. This work aims to combine the best elements of each of these approaches. We offer a methodology for creating models, based on in-class clicker questions, to predict cross-term student performance. In as early as week 3 in a 12-week CS1 course, this model is capable of correctly predicting students as being in danger of failing, or not, for 70% of the students, with only 17% of students misclassified as not at-risk when at-risk. Additional measures to ensure more broad applicability of the methodology, along with possible limitations, are explored.

Categories and Subject Descriptors

K.3.2 [**Computer Science Education**]: Computer and Information Science Education

Keywords

Peer Instruction; CS1; clickers; prediction

1. INTRODUCTION

Early identification of struggling students could be highly valuable for instructors and students alike. Instructors could explore possible intervention strategies to help struggling students, and students who are made aware that they are likely struggling may be spurred to change study habits or seek additional assistance. Moreover, communicating with students about their performance may play a role in the general effort to better personalize education in small and large classrooms and online learning environments.

To identify struggling students, research from the 1970s through the 1990s focused primarily on static personal features (GPA, gender, etc.). Recent work has begun using dynamic data, such as assignments [2] and in-class performance [12] to identify these students.

Porter et al. provided promising results that easy-to-obtain, in-class clicker data is correlated with final exam scores; however, that analysis was limited to a single term [12]. Next, Ahadi et al. demonstrated that machine learning models based on a combination of static and assignment submission data could be used to predict students in the bottom half of final exam scores; however, that analysis was done in a CS1 course with a very large number of assignments [2]. That latter work focused on modeling a single term, although the authors did show the potential to use machine learning to perform cross-term predictions.

The present work aims to build upon the strengths of both of these works by using easy-to-obtain, in-class clicker results to predict student outcomes across terms. By doing so, we aim to provide modeling practices whose data is easy to generate (requiring little course change), and where the time-consuming collection of sensitive student demographics or background information is not required.

The modeling is performed on a Peer Instruction (PI) CS1 course in Python. Our goal is to predict a final exam score using in-class clicker question data, after which classification decisions ("struggling" or "not struggling") can be made. This two-step process gives flexibility to the instructor, as they can choose the threshold at which to intervene.

In addition, recognizing that not all CS1 courses employ PI, we explore the use of the same questions as quiz questions rather than in-class clicker questions. We find that model accuracy declines only slightly, suggesting that our approach may be applicable to lecture-based courses as well.

We offer the following in this work:

- Despite confounding factors between terms (different students, different assignments, different topic ordering, and different exams), we demonstrate that naturally-collected clicker data from one term can be used to create a statistical model capable of identifying struggling students early in the other term (week 3 of 12).

- We provide details on building a model for a course and offer guidance on selecting thresholds based on instructor need. For those not using PI, we provide evidence that the use of questions as quizzes can provide similar prediction accuracy.

- Our model for the studied CS1 course accurately predicts 22% of students as struggling and 48% as not struggling. Only 17% of students are miscategorized as not needing assistance when assistance was needed.

ICER '16, September 08-12, 2016, Melbourne, VIC, Australia

DOI: http://dx.doi.org/10.1145/2960310.2960315

2. BACKGROUND

Many researchers over the last several decades have reported correlations between early-term predictors and late-term outcomes (see Robins [13] for a review). These kinds of relationships are of interest for at least two reasons: in terms of teaching, they help us determine who is likely to succeed or fail in a particular CS course; in terms of the discipline at large, they can inform research that seeks to increase representation or performance of particular student subgroups.

Much of the early work used static student factors that cannot change or are difficult to change through the duration of the course. A typical such study is that of Wilson and Shrock [18], in which the authors report relationships between 12 student factors and midterm exam score. These factors include gender, programming experience, non-programming computing experience, comfort level, and attributions of success and failure.

These kinds of factors, existing before the course starts, are called presage factors [5]. Process (or dynamic) factors, on the other hand, capture aspects of learning and the interaction between student, context, and content. Recent availability of educational technology such as clickers and instrumented programming IDEs means that we have a wealth of process data at our disposal [7]. Process factors are often more powerful predictors than are presage factors [2, 17]. In addition, we see process factors, moreso than presage factors, as under our purview to influence. With early detection of concerning process data (low class attendance, maladaptive programming patterns, incorrect responses to formative feedback), we may be able to intervene and set a new path for these students. A prerequisite for such intervention, of course, is our ability to use process data to make accurate, early predictions of student success.

Much of the relevant literature in this area concerns predicting performance in CS1 [2, 3, 12, 16, 17]. For example, one study [16] collected data on assignment submission time to deadline, time elapsed between snapshots, edit distance between snapshots, and other features of the programming process, and used this data to predict whether each student would fail, pass, or excel in a CS1 course. Two weeks into the six-week course, the authors could make this prediction with 64% accuracy.

Research on the early prediction of students' CS performance in courses other than CS1 is sparse. In one work, the authors study the predictive power of weekly multiple-choice tests, homework grades, and previous CS1 performance on CS2 exam grades [4]. Student data was anonymized in such a way that weekly test scores could not be linked with exam scores. Instead, comparisons of grade distributions suggest, two-thirds of the way through the course, that test scores could be used to predict exam scores. Other significant predictors of exam score include scores on all team-based assignments (16-25% of variance explained) and CS1 grade (41-60% of variance explained).

The two prior papers most closely aligned to the present work are those of Porter, Zingaro and Lister [12] and Ahadi, Lister, Haapala and Vihavainen [2].

In the earlier of these papers, the authors leverage Peer Instruction (PI) clicker data as a naturally-occurring source of what students know [12]. The authors found significant correlations between clicker question correctness and final exam scores. The case is made that one can predict exam scores using only the first three weeks of course clicker data. While the work suggests as much, it does not follow through by predicting student success in a future offering of the course. While correlations within one semester are interesting in themselves, such correlations do not tell us whether we really could predict and potentially intervene on students who will be at-risk as a course progresses.

In the latter paper, the authors report on a 6-week CS1 course [2]. The interest is in what can be predicted after the first week of class, rather than after the first three weeks of class. This was a traditional lecture-based course, not a PI course. As such, the PI fine-grained clicker data was not available. In lieu of clicker data, the authors use age, gender, grade average, major, prior experience, number of steps (total keypresses) taken on each assignment, and correctness scores on each assignment. There were 24 assignments given in the first week of class. Using machine learning techniques, the authors built a well-performing model on the training data (first term) and applied it to the test data (second term). Interestingly, the features with highest information-gain — and therefore those selected for the model — are largely the process predictors (e.g., number of steps taken on various assignments), not the presage predictors. The best-performing model was correct for 86-90% of the training data students and 71-80% of the test data students.

One concern with the process predictors from Ahadi et al. [2] is that some of the assignments asked in the first week resemble end-of-term outcomes. For example, one assignment has students write a program that repeatedly asks for and plots numbers within a given range; this requires integrative knowledge of variables, conditionals, and loops. Although the predictive value is clear, there are concerns that the nature and number of these early-term assignments may dissuade students without considerable prior programming experience.

The present paper draws ideas from and extends both of these papers. First, from Porter et al. [12], we use PI clicker data; no sensitive student data is collected or required. In contrast, the data used by Ahadi et al. [2] includes student surveys, student records' access, instrumented IDEs, and a large number of early programming assignments, all of which complicate data collection and course administration. Second, we leverage the idea of using the model built in one term to make predictions in a subsequent term [2, 17]. Such prediction across terms serves as evidence of a model that generalizes outside of its immediate context.

3. METHOD

The data are collected from a CS1 course taught in Python at a large North American research university during two consecutive fall terms ($n = 171$ and $n = 142$, respectively). Each offering was 12 weeks long and had three weekly 50-minute lectures. Student work included two term tests, weekly pair-programming labs, two large programming assignments, and a final exam. The second term, but not the first, also included short, weekly programming exercises. Both terms were taught by the same instructor using the PI pedagogy [10, 11, 19]. As further explained below, PI focuses on students discussing conceptual questions and responding with electronic clickers. The PI materials used in this study are based on those of a prior study [12].

The course content and lecture materials between the two terms are quite comparable as they were taught by the same instructor. We examined the PI clicker questions used in each course offering and found that 88% of all clicker questions appeared in both terms. Student responses to early

matched questions (first three weeks), along with final exam scores, were used as the training and test data. Throughout the analysis, the former term is used as training data and the latter as test data.

3.1 Peer Instruction Format

PI is an increasingly popular active learning pedagogy in computer science courses [9, 11]. For clarity, we outline the core components of a PI class, the different votes we collect from students, and possible interpretations of these votes.

In a PI course, instruction is centered around a series of questions that students solve in class. Although the number of questions varies, the courses studied here had 3-5 questions per lecture. As part of the process, students are asked to select their answer, often using in-class electronic response systems (clickers). An individual PI cycle follows a well defined process:

Individual Vote: Students are shown the question and asked to solve it individually. They then record their answer using the clicker, the results of which are transmitted to the instructor. As an approximation, this vote can be viewed as student understanding prior to in-class instruction.

Group Vote: Students are then asked to discuss the problem in small groups and come to a consensus. They then respond again using the clicker. As an approximation, this vote can be viewed as student understanding after discussion. While it is possible that students could vote the same as their peers without understanding the chosen response, prior research suggests that this is not a major concern [10].

Classwide Discussion: Based on student responses, the instructor leads a classwide discussion about the question, aiming to both engage students in a discussion of why particular response choices were made and clarify to the class why certain responses are correct or incorrect.

(Optional) Isomorphic Vote: For some questions deemed to be particularly important for learning, the instructor asks a follow-on question after the classwide discussion. This isomorphic question is a different question on the same concept just discussed. The students respond to this question individually. As an approximation, the isomorphic vote can be viewed as student understanding at the conclusion of the PI process on that concept.

Through a participation grade, students are rewarded for attending class and participating in PI questions, not on providing correct responses.

3.2 Data Analysis and Modeling

In this section, we describe the process for creating and applying the model. The steps include: partitioning the data, preprocessing the data, using Principal Components Analysis to reduce data dimensionality, building the model, using the validation set to determine a classification threshold, and applying that model and threshold on the test set. Details are provided to encourage replication.

Defining Struggling Students: This decision depends on the course and the instructor. For our study, we spoke with the course instructor and defined "struggling" to be those students who score in the bottom 40% of students on the final exam. This threshold could be modified based on instructor preferences.

Data Set Partitioning: The data from the first term was split into two subsets: training set and validation set. The validation set is used to optimize the number of predictors in the trained model and to determine thresholds to best classify students. Only after the model is fully constructed and appropriate thresholds determined is it applied to the test set (students in the second term). Table 1 illustrates the size of each set for our model. The size ratio of the training set to the validation set is 2:1.

Table 1: Data Set Size After Preprocessing

Term 1		Term 2
Training Set	Validation Set	Test Set
117	54	142

Data Preprocessing: Two standard steps are conducted to prepare the data for analysis. First, to account for differences in difficulty of exams across terms, we convert raw final exam scores to scaled scores. The scaling is done by first determining each exam score's z-score, then scaling all z-scores between the maximum and minimum z-score. Final exam scores of the training set, validation set, and test set are thereby scaled to the range [0,1].

The second step addresses clicker responses that are missing. A student may fail to answer some clicker questions (e.g., by being absent, arriving late, forgetting to click, or leaving early), but our modeling approach does not handle missing data. If we simply omit students who fail to answer one or more questions, then we would lose the data of the vast majority of students in the class. We therefore use data imputation. Data imputation is essentially informed guessing: we guess how the student would have responded based on their responses and the responses of other students. The accuracy of such guesses is impacted by the number of questions that the student legitimately answered. (We investigate the impact of students who commonly miss class in Section 4.4.) Data imputation is performed using the well-established R mi package [15].

Dimensionality Reduction: As many clicker questions were asked each week, using all responses would lead to a model that overfits the data. To prevent this overfitting, we use Principal Components Analysis (PCA) to reduce the number of dimensions in the data using the R Caret package [8]. PCA extracts a given number of predictors (i.e., composites of clicker questions) that best represent the data. We explore using different numbers of predictors to determine the best number for our data by optimizing the model accuracy with regard to the validation set (the test set is not used in this process).

Building the Linear Regression Model: We create a linear regression model using the principal components chosen by PCA and predict a scaled final exam score for each student. The prediction model was created and tested using the R Caret package [8]. In the training phase, we use k-fold cross-validation (k=10) to optimize the regression model parameters. The trained model also provides variable importance information, which describes the predictive power of each clicker question.

Classifying Students: The output of the model, when applied to either the training or test data, is a predicted final exam score. This prediction can then be used to classify students. The following are the possible student classifications:

- **Correct Non-Intervention.** Our model correctly predicts that a student will be in the top 60% on the final exam.
- **Correct Intervention.** Our model correctly predicts that a student will be in the bottom 40% on the final exam.

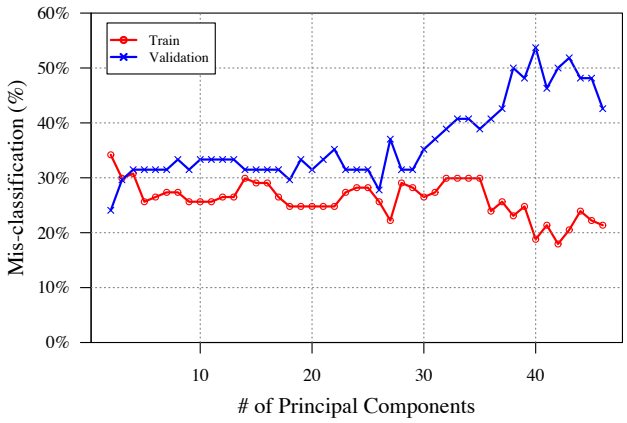

Figure 1: Impact on Accuracy of the Number of Principle Components

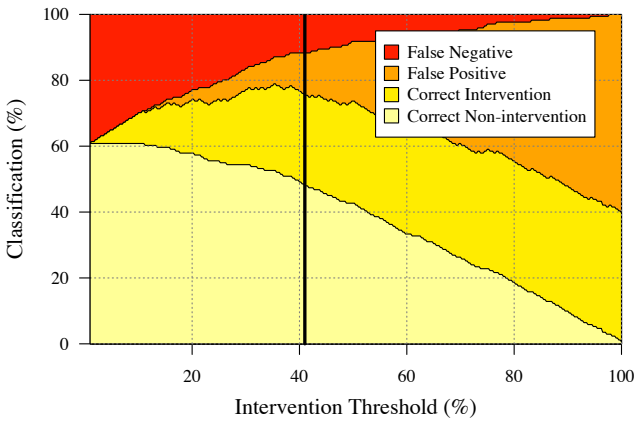

Figure 2: Intervention Threshold for Training and Validation Set

- **False Negative.** Our model predicts that a student will be in the top 60% on the final exam, but they end up in the bottom 40%.
- **False Positive.** Our model predicts that a student will be in the bottom 40% on the final exam, but they end up in the top 60%.

Given that we have predicted final exam scores, we could simply apply the original threshold that defines struggling students (in our case, the bottom 40%) to the predicted final exam results. However, due to error variance in predicted scores, this choice may not be ideal in that it could result in a large number of false positives or false negatives. As such, we instead use the validation set (again, not the test set) to determine an appropriate threshold for classifying students. We refer to this second threshold as the classification threshold or intervention threshold. We explore tradeoffs inherent in determining this threshold in Section 4.1.2.

To summarize, there are two relevant thresholds in our modeling approach. The first is the percentage of students that should be classified as struggling (this can be based on common exam outcomes and final grades). The second is the threshold that we use to classify students as in need of assistance, taking into account the likelihood of misclassifying students.

4. RESULTS

In this section, we model student performance across terms. We describe steps taken to build the model and apply it to identify struggling students. We then explore the model further to learn about influential PI votes, class attendance, and applicability to lecture classes.

4.1 Constructing the Model

4.1.1 PCA Variable Selection

We use the training and validation set to determine the appropriate number of principal components to use in building the model. We define model accuracy as the correct classification rate. Figure 1 illustrates the misclassification rate of the training set and the validation set with respect to the number of principal components. The misclassification rate is the proportion of students for whom a prediction of being in the top 60% or bottom 40% would be incorrect. This figure demonstrates that fewer than 30 principal components provides good validation set accuracy. Additional components may improve the accuracy of the model for the training set, but at the potential expense of the validation and test sets. With a view toward choosing fewer rather than more components, and a desire to optimize performance on the training and validation sets, we chose nine principal components for our model.

4.1.2 Intervention Threshold

We next set the threshold for the percentage of the class on which to intervene. This intervening score could simply be set to the same point at which we consider students to be struggling (e.g., the bottom 40%). However, the choice of intervention threshold has a large impact both on model accuracy and on the number of students who are identified as potentially needing help. The higher the threshold, the more false positives (students receiving help who do not need it) and the more instructor resources are spent helping those students. In turn, the lower the threshold, the more false negatives (students not getting help who need it).

One can consider the two threshold extremes to better understand the tradeoffs present in selecting an intervention threshold. First, one might choose a threshold of 0%. This threshold would cause no students to be identified for an intervention, meaning that we never help a student who does not need help (zero false positives), but also never help students who do need help (maximum false negatives). Second, one might choose a threshold of 100%, in which case we help everyone that needs help (zero false negatives), but also help everyone else (maximum false positives). The question then becomes: how should we balance these tradeoffs? The answer ultimately comes down to instructor discretion based on available resources and the cost of intervention.

As an instructor cannot know the results for their present class, they can use student data from the prior term to help choose an appropriate intervention threshold. To help visualize the impact of the instructor's threshold decision, Figure 2 shows the impact of the intervention threshold for the first term data (training set and validation set combined). In this figure, we see that as we increase the threshold from 0% to 40%, our method tends to more accurately predict poor performers. However, as we continue to raise the threshold, we introduce false positives at an increasing rate.

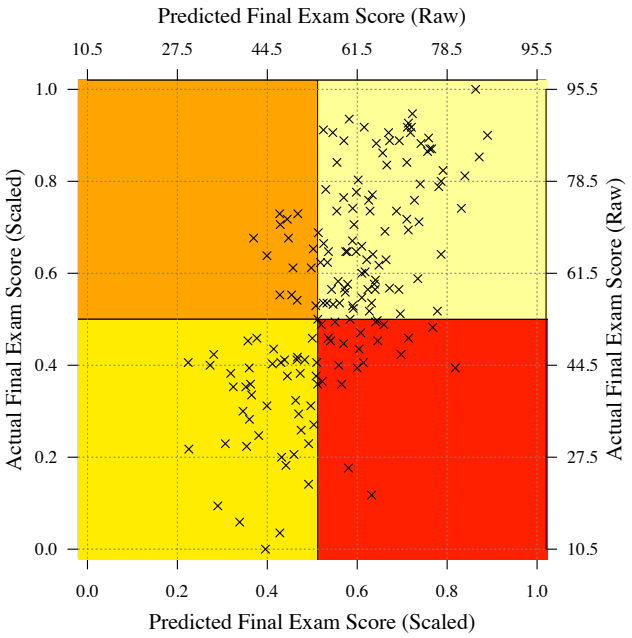

Figure 3: Model Accuracy for Training Data

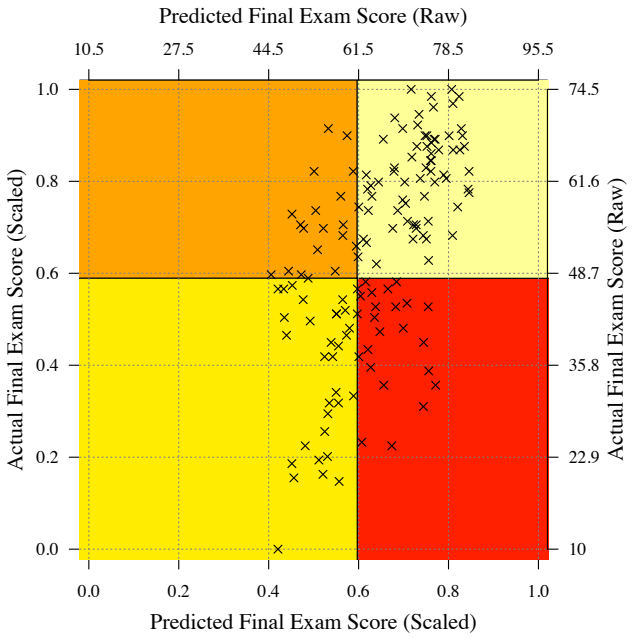

Figure 4: Model Accuracy for Test Data

For the remainder of this analysis, we chose the intervention threshold with the highest accuracy (i.e, that minimized the misclassification rate). Other threshold decisions to minimize false positives or false negatives would be possible and would be at the instructor's discretion.

We use the validation set alone (not with the training set) to determine the intervention threshold with the highest accuracy; that threshold is 41%. This means that to identify students who are in the bottom 40% of the class, we will conclude that students whose predicted final exam score is in the bottom 41% will struggle. Although the small difference between 40% and 41% may suggest that the alternative threshold is unnecessary, when we use different model parameters we found larger thresholds (e.g. 55%). We next examine our model for the training data in the context of this threshold and then apply this threshold to the test data.

4.1.3 Model Accuracy for Training Set

Although the ultimate goal of developing this model is to examine its accuracy on the test set, here we begin by examining the model's accuracy for the training set. Figure 3 plots the model's predicted final exam score per student (x-axis) against their actual final exam score (y-axis). There are two labels per axis. The "Raw" scores are the actual scores on the exam out of 105 points. The "Scaled" scores are scaled by the students' z-scores and are hence between 0 and 1. Recall that the reason for the scaled scores is that exams across multiple terms may have different difficulty.

The linear model in this figure, built using nine principal components, is fairly accurate ($rho : 0.628, R^2 : 0.395$). To examine the prediction accuracy, we focus on the four colored quadrants. Keeping with the color convention from Figure 2, we can identify the regions of correct predictions (intervention, no intervention, false positive, and false negative) for the threshold chosen above.

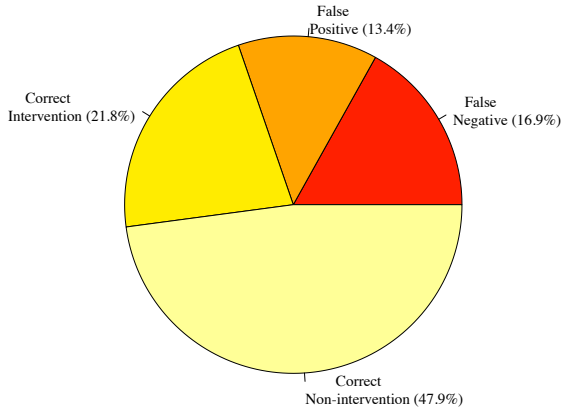

Figure 5: Classification Accuracy for Test Data

4.2 Applying the Model

4.2.1 Model Accuracy for Test Set

Figure 4 provides the plot of predicted versus actual final exam score for the test data. The model is again reasonably accurate ($rho : 0.574, R^2 : 0.329$) despite the differences between terms (ordering of topics, different students, different exams, etc.). We revisit these differences in Section 5.

4.2.2 Student intervention accuracy

While the model provides a predicted final exam score, our intended use is to apply the 41% threshold to determine students who are likely to fall into the bottom 40% of exam scores. Figure 5 provides the accuracy of our student classifications. 70% of students are accurately predicted. 13% of predictions are false positives and 17% are false negatives. Recall that one can increase the intervention threshold to

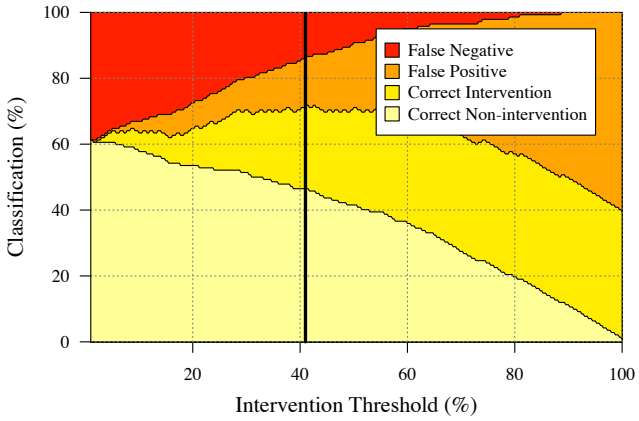

Figure 6: Threshold Impact for Test Data

decrease the number of false negatives, but at the expense of increasing false positives.

As mentioned above, an instructor's choice of intervention threshold impacts false negatives and false positives. Figure 6 demonstrates the impact of the intervention threshold on the test set. This figure is shown only to demonstrate the tradeoffs in the context of the test set results; recall that the threshold was determined using the validation set. Such a figure of the test set results would not be available until the end of the later term, too late for intervention to be useful.

4.3 Model Influences

Figure 7 provides the importance scores for the early-term clicker questions as determined by the model. In the top half of the figure, the questions are labeled by the relevant vote in the PI process: individual (solo), group, or isomorphic (iso). The key take-away here is that both isomorphic and individual votes play a larger role, particularly among the very top predictors, than group votes. This corresponds with prior findings that group votes can be noisy due to confounds between actual learning and copying perceived correct answers [10].

In the bottom half of the figure, the questions are organized by the time they occurred in the term.[1] Here we see that questions from week 2 are among the very top predictors. Week 2 content includes functions, particularly return types, and boolean expressions and conditionals. As expected, questions from week 1 and week 3 also appear among the top 15 predictors.

4.4 Impact of Student Attendance

The results above include all students in the test set, regardless of their class participation. Is it reasonable to expect the model to predict a student's final exam score when they have only attended a small number of classes? To study this question, we examined the model's accuracy on only students who answered 70% or more of the clicker questions. For this large subset of students, the model predicting final exam scores for the test set remains similarly accurate ($rho : 0.556, R^2 : 0.309$).

The classification of poorly-performing students is more accurate than when all students are included. Figure 8 provides the resulting classification accuracy. The model's accuracy for student classification has increased from 70% to 76%, demonstrating the importance of class attendance for model accuracy.

[1]"Other" occurs because some questions that appeared in the first 3 weeks of the test set were in later weeks in the training set due to minor reorganization of topics in the course.

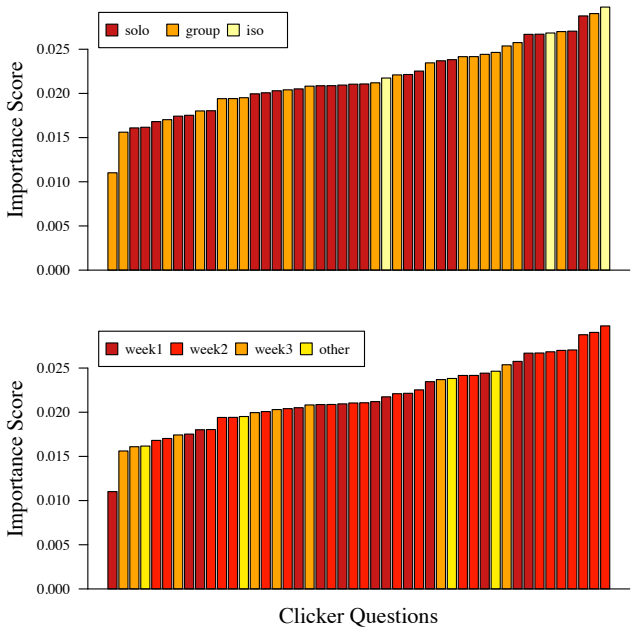

Figure 7: Question Importance for the Model

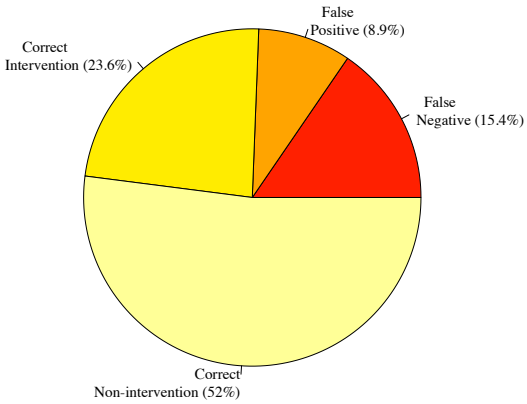

Figure 8: Model Classification Accuracy for Only Frequently-Responding Students in Test Data

4.5 Applicability to Non-PI Classes

One core aim of this effort is to ensure that the modeling is made more accessible to other instructors who wish to benefit from predictions in their classes. Although PI has gained considerable traction in computing [11], a large number of instructors may not wish to adopt PI and/or clickers. In this section, we examine the possibility of using the PI clicker questions as either before-class quizzes or brief start/end-of-class quizzes. The questions are available at [1].

To explore this idea, we built a new model using only individual clicker votes. These votes may be representative of student thinking before class as they occur before the group and classwide discussion. One confound, however, is

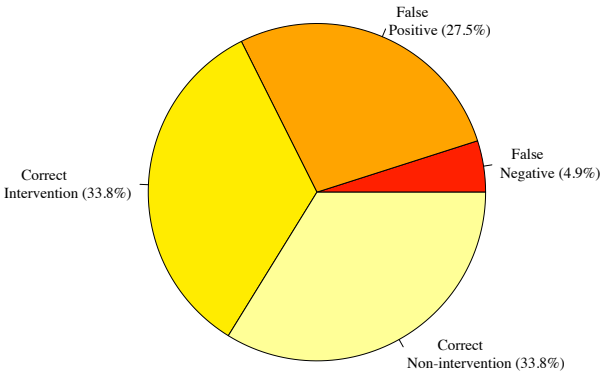

Figure 9: Classification Accuracy for Test Data When Limited to Only Using Individual Votes

that they may occur mid-way through a class and students may have learned from prior class content. Alternatively, one might ask these questions after the class, in which case the results may more closely resemble the isomorphic votes. As the isomorphic votes were highly predictive in our model, the results in this section may understate the expected modelling ability of questions used at the end of class.

Compared to the full model, this new model is only marginally worse at predicting student final exam scores in the test set (rho : $0.546, R^2$: 0.299) and at classifying poor performers. Figure 9 shows that the classification accuracy is 68% for all students, compared to 70% accuracy when we included the group and isomorphic votes. Although false negatives have been reduced significantly, this is simply due to selecting a higher intervention threshold for maximum accuracy, which, in turn, increased the number of false positives. The overall accuracy suggests that instructors who wish to adopt this approach with lightweight multiple-choice quizzes may be able to successfully predict low performing students without the requirement of adopting PI.

5. DISCUSSION

This work demonstrates the potential to use student clicker data to predict low-performing students across terms. The requirement on instructors is lightweight as they can either use the same clicker questions across terms or, potentially, quizzes including these multiple choice questions. After building the linear model, instructors can then make their own decisions about how to identify poor performers based on the two threshold values: the percentile on the final exam that is considered low-performing and how to optimize their classification threshold.

5.1 Revisiting Results

Cross-Term Differences: The ability to predict student outcomes across terms may seem straightforward given that the clicker questions changed little from the first term to the second. However, a number of differences are evident even though the same instructor taught both terms. The differences include: different students, different in-class student inquiries and resultant discussion, different assignments, different topic ordering (topics were rearranged slightly to address assignment changes), and most critically, different exams. An additional confound is the noise and missing data inherent in clicker responses graded on participation. Although the modeling approach outlined here may help mitigate some of these effects (e.g., using student percentiles to reduce the impact of differing exam difficulties), the robustness of the modeling technique is all the more surprising when these differences and confounds are recognized.

Question Quality: The quality of the model is implicitly based on the quality of the clicker questions and their utility for identifying student misconceptions across a variety of topics. Two major factors contributed to the quality of the questions in this course. The first is that CS1 has been widely studied by the computer science education research community and the design of these questions was informed by that research. The second is that the instructor has taught the course for a number of years and has refined the questions over this time period. As a result, we provide these questions to those wishing to adopt this approach [1].

Helping Poor Performers: This paper is not the first to recognize the inherent benefits of identifying low-performing students early in the term [2, 12], but the approach outlined here vastly reduces barriers to adoption (e.g., large number of coding assignments in the first week). Once it is easy to identify these students, instructors will be tasked with acting on these results. Interventions to improve students' outcomes deserve further attention and are the focus of ongoing work.

Low-Performing Threshold: The thresholds for poor performers have a large impact on the classification accuracy and, potentially, instructor resources dedicated to intervention plans. We chose a bottom 40% threshold based on cutoffs chosen by the instructor, but other values could be chosen. We caution, however, that the modeling technique outlined here is more effective at classifying large groups than small groups. As such, this particular model may struggle to identify small subsets of students (e.g., the bottom 5%) as identifying such outliers is both difficult in general and ill-suited for this model.

As mentioned previously, an instructor's choice of intervention threshold impacts the number of students for whom intervention is offered. An instructor can optimize this threshold to maximize accuracy, minimize false negatives, or perform an intervention for some particular number of students. Resource requirements may, at least partially, constrain the instructor. For example, should the instructor wish to e-mail students in jeopardy, they may optimize for a fairly low false negative rate (accepting more false positives) because the intervention is inexpensive. In contrast, if the instructor aimed to have a special in-person session for struggling students, they may optimize based on room sizes or the availability of instructional staff.

5.2 Comprehensive Picture of Early CS1

Prior work suggests fruitful links between early identification of students and informing what we know about CS concepts that are challenging to students [2, 12]. Porter et al. [12] focus on individual clicker questions and their relationship to exam outcomes. However, due to the type of modeling used, a number of highly correlated clicker questions on the same topic may all appear important. Ahadi et al. [2] study predictors in the larger context of a machine-learning model, but the predictors are scores on code-writing assignments that do not isolate individual concepts.

Our method allows us to again benefit from the best features of each of these prior works. By examining highly-predictive clicker questions in the context of a comprehen-

Table 2: Topics among top 10 predictive questions, ordered by importance rank. Questions whose votes appeared multiple times include the importance rank and corresponding vote categories, respectively.

Rank(s)	Week	Vote	Topic
1,5,7	2	iso, gr, ind	Code tracing through nested function calls where variables used are in or out of scope
2,3	2	ind,gr	Logic/ Boolean expression evaluation and boolean variable assignment
4	2	ind	The difference between a function printing a value versus returning a value
6	2	iso	Given a boolean expression with variables, determine what values those variables would need to have to evaluate as false
8	1	ind	Code tracing through a single function call where the function has multiple arguments
9	1	gr	Variable types (integer versus double)
10	3	gr	Code Tracing through nested conditionals

sive model, we gain a more complete picture of the critical topics and concepts in the first three weeks of the course.

To do this, we examined the top ten questions in terms of importance from Figure 7. As each question may be answered multiple times through the individual, group, and optional isomorphic vote, a single question may appear multiple times among the top predictors. Although this may seem contradictory to the notion that highly-correlated results should be pruned from the model, recall that each of these votes represents a different point in student understanding: before discussion, after discussion, and after instructor explanation, respectively. Indeed, two questions appeared for multiple votes, resulting in eight unique questions.

The topics of these questions appear in Table 2. Critical topics from the beginning of CS1 appear in the top predictors, including variables, types, boolean expressions, function calls, parameters, scope, and conditionals. Loops are absent from this list because they do not appear in the course until Week 4. None of these topics, nor their ranking, appears particularly contradictory to what one might expect as top predictors from an introductory course [6, 14].

5.3 Threats to Validity

There are two categories of threats to validity. The first is with regard to the building of the model while the second focuses on the course itself.

5.3.1 Model Construction

Class size: The construction of this model required a fairly large class as we partition the training data into a 2/3 training set and 1/3 validation set. Moreover, among the remaining training set, k-fold cross validation (k=10) is necessary to avoid overfitting. As such, the applicability of our technique to smaller classes is unknown.

Model Robustness and Overfitting: In exploring successful model construction on the training set, a number of parameters were explored and the resultant models had similar degrees of correctness. Use of the test set was limited in order to avoid overfitting to the test set. However, in the research process, the test set was queried more than once. The robustness and similarity of results (e.g., classification accuracy between 68%-73%) across these queries suggest that overfitting did not occur, however whenever a test set is examined more than once, overfitting/overtuning becomes a concern.

5.3.2 Cross-Term Course Repetition

Questions: As mentioned, the clicker questions in this study had a basis in the literature and have been used in multiple courses. As the model is based on the question results, modeling for different questions may yield different results.

Exam: The questions on the final exams across the two terms are completely different because exams are made public at the instructor's institution. As previously mentioned, the differences between the exams (both in terms of question difficulty and conceptual coverage) may lead to lower model accuracy. By contrast, reusing significant portions of exams could yield increased modeling accuracy.

Instructor: The same instructor taught both terms of the course. Whether the model applies to another instructor using the same questions is unknown and is part of ongoing work.

6. CONCLUSION

This work proposes a lightweight modeling technique based on prior-term data to identify low performers in a course. The approach relies solely on the naturally-occurring student responses to clicker questions in a Peer Instruction classroom. Should an instructor not use Peer Instruction, our results suggest that simply asking the same multiple-choice questions before or after class could produce similarly accurate results.

At the heart of the prediction methodology is the instructor's goals for identifying poor performers. As such, they can choose to trade off overestimating the number of poor performers to ensure they reach everyone in need or underestimating the number of poor performers to avoid spending course resources on those who may not need help. For our CS1 course in python, our approach results in a model that accurately predicts 70% of students in the test set as either needing or not needing assistance. We also show that we are able to more accurately predict students who attend class more frequently and that the important topics identified by the model can inform our view of the early weeks of CS1.

7. ACKNOWLEDGMENTS

Thank you to the anonymous reviewers for their helpful feedback on this work. This work was supported by NSF grant 1140731.

8. REFERENCES

[1] Peer instruction for computer science. peerinstruction4cs.org, 2013.

[2] A. Ahadi, R. Lister, H. Haapala, and A. Vihavainen. Exploring machine learning methods to automatically identify students in need of assistance. In *Proceedings of the Eleventh international Conference on Computing Education Research*, pages 121–130, 2015.

[3] S. Bergin, A. Mooney, J. Ghent, and K. Quille. Using machine learning techniques to predict introductory programming performance. *International Journal of Computer Science and Software*, 4(12):323–328, 2015.

[4] H. Danielsiek and J. Vahrenhold. Stay on these roads: Potential factors indicating students' performance in a CS2 course. In *Proceedings of the 47th ACM Technical Symposium on Computer Science Education*, pages 12–17, 2016.

[5] M. de Raadt, M. Hamilton, R. Lister, J. Tutty, B. Baker, I. Box, Q. Cutts, S. Fincher, J. Hamer, P. Haden, M. Petre, A. Robins, Simon, K. Sutton, and D. Tolhurst. Approaches to learning in computer programming students and their effect on success. *Higher Education in a changing world: Research and Development in Higher Education*, 28:407–414, 2005.

[6] K. Goldman, P. Gross, C. Heeren, G. L. Herman, L. Kaczmarczyk, M. C. Loui, and C. Zilles. Setting the scope of concept inventories for introductory computing subjects. *Transactions on Computing Education*, 10(2):1–29, 2010.

[7] P. Ihantola, A. Vihavainen, A. Ahadi, M. Butler, J. Börstler, S. H. Edwards, E. Isohanni, A. Korhonen, A. Petersen, K. Rivers, M. A. Rubio, J. Sheard, B. Skupas, J. Spacco, C. Szabo, and D. Toll. Educational data mining and learning analytics in programming: Literature review and case studies. In *Working group report of the 20th annual conference on Innovation and technology in computer science education*, pages 41–63, 2015.

[8] M. Kuhn. Building predictive models in R using the caret package. *Journal of Statistical Software*, 28(1), 2008.

[9] C. Lee, S. Garcia, and L. Porter. Can peer instruction be effective in upper-division computer science courses? *Transactions on Computing Education*, pages 12–22, 2013.

[10] L. Porter, C. Bailey-Lee, B. Simon, and D. Zingaro. Peer instruction: Do students really learn from peer discussion in computing? In *Proceedings of the Seventh international Conference on Computing Education Research*, pages 45–52, 2011.

[11] L. Porter, D. Bouvier, Q. Cutts, S. Grissom, C. Lee, R. McCartney, D. Zingaro, and B. Simon. A multi-institutional study of peer instruction in introductory computing. In *Proceedings of the 47th ACM Technical Symposium on Computing Science Education*, pages 358–363, 2016.

[12] L. Porter, D. Zingaro, and R. Lister. Predicting student success using fine grain clicker data. In *Proceedings of the tenth annual conference on International computing education research*, pages 51–58, 2014.

[13] A. Robins. Learning edge momentum: A new account of outcomes. *Computer Science Education*, 20(1):37–71, 2010.

[14] A. Robins, P. Haden, and S. Garner. Problem distributions in a CS1 course. In *Proceedings of the 8th Australian conference on Computing education*, pages 165–173, 2006.

[15] Y.-S. Su, A. Gelman, J. Hill, and M. Yajima. Multiple imputation with diagnostics (mi) in R: Opening windows into the black box. *Journal of Statistical Software*, 45(1):1–31, 2011.

[16] A. Vihavainen. Predicting students' performance in an introductory programming course using data from students' own programming process. In *IEEE 13th International Conference on Advanced Learning Technologies*, pages 498–499, 2013.

[17] C. Watson, F. W. Li, and J. L. Godwin. No tests required: Comparing traditional and dynamic predictors of programming success. In *Proceedings of the 45th ACM Technical Symposium on Computer Science Education*, pages 469–474, 2014.

[18] B. C. Wilson and S. Shrock. Contributing to success in an introductory computer science course: a study of twelve factors. *SIGCSE Bulletin*, 33:184–188, 2001.

[19] D. Zingaro. Peer instruction contributes to self-efficacy in CS1. In *Proceedings of the 45th ACM technical symposium on Computer Science Education*, pages 373–378, 2014.