Paper ID #36869

Predicting Success in Programming I

Britton D Wolfe (Chairperson and Professor)

Chair, Computer Science Grove City College

Eli Christopher Lowry

Predicting Success in Programming I

Abstract

We investigate the degree to which student scores on a diagnostic quiz can predict success in a first programming course. The diagnostic questions do not require programming knowledge, nor are they programming questions, to facilitate early feedback and avoid measuring student confusion about programming syntax. Instead, the questions attempt to assess fundamental reasoning skills that we hypothesize are important for success as a programmer. A multiple linear regression finds a significant correlation between scores on some of the diagnostic questions and exam scores in the course, but there remains substantial variance in exam scores unexplained by the model, even after accounting for students' major and different instructors. Nonetheless, students with low scores on one particular diagnostic question have a much higher risk of earning a D, F, or W in the course (44%) compared to the other students in the course (13%), as computed on a validation data set.

Introduction

Content throughout introductory programming courses—commonly called "CS1" courses—relies crucially on concepts covered in the first few weeks of the course. If a student does not build an understanding of the foundational concepts covered during those first weeks, they will be unable to understand the remaining material in the course. Such students either struggle through the rest of the course as they try to learn past material and keep up with the current topic, or they withdraw from the course entirely. Thus, helping students understand the concepts in the first weeks of the course is essential to their continued success. Methods for identifying and helping students who are likely to struggle in CS1 courses would be widely applicable, particularly since those courses are known to have a low passing rate (68%) [1].

Students' programming behavior—the steps they take to solve a problem and the correctness of their answers—during those first weeks can provide some indication of students' propensity for success in the course [2]. However, such an approach is limited by the amount of data the instructor can collect about programming behaviors in the first weeks.

Our work examines how well a novel diagnostic quiz can predict student success in CS1, using a quiz that requires no prior knowledge of programming and contains no programming questions. Such a quiz is easily administered on the first day of class, providing quick feedback to help the instructor identify high-risk students. By not using programming questions, the quiz avoids the fact that programming questions conflate students' understanding of syntax with their logical reasoning skills. Syntax understanding is expected to be poor for new programmers [3], even those who end up doing well in the course [4], so errors of both kinds occur throughout early programming assignments [5].

Related Work

Quille and Bergin **[6]** conducted an extensive survey of research on predicting success in CS1. Of the 47 studies they examined in depth, they found only three (**[2]**, **[6]**, **[7]**) that were conducted after 1990, validated their model on a separate set of students (typically longitudinal studies), used a large sample size, and were able to predict student success with high accuracy while still early in the CS1 course. This section discusses those studies as well as some earlier studies that set the stage for that work. **Table 1** lists the predictors that each study examined and which of those predictors the authors noted as most significant.

Wilson and Shrock **[8]**, **[9]** examined correlations between several student attributes and success in CS1. Comfort level with programming and math background were the most important predictors of success. Comfort level was measured with a questionnaire about "asking and answering questions in class, in lab, and during office hours; anxiety level while working on computer assignments; perceived difficulty of the course; perceived understanding of concepts in the course as compared with classmates; and perceived difficulty of completing the programming assignments" **[8]**. Measuring comfort level thus requires that students already have experience in the class, which would not allow early detection of high-risk students.

Rountree et al. **[10]** found that the "strongest single indicator of success was the grade the student expected to achieve at the beginning of the course." Their subsequent work **[11]** went beyond studying single predictors, using decision trees to examine combinations of factors. Students' expectations about what grade they would receive continued to be the most prevalent factor in identifying high-risk students, followed by student background subject. The classifier for predicting high-risk students had a precision of 53%. Out of 77 students classified as high-risk, 41 actually failed the course, compared with a 27% failure rate for the entire class.

Bergin and Reilly **[12]** examined 15 possible predictors, finding that student's comfort level with programming and perception of their programming performance were the strongest individual predictors. However, the perception of programming performance was surveyed in the second semester of the course, which would not permit early detection of high-risk students. The combination of students' perception of programming performance, comfort level, high school math score, and gender accounted for 79% of the variance in programming performance.

Quille and Bergin **[6]** revisited that earlier work, confirming that high school math scores and student's programming self-efficacy are significant predictors of success. They explored several combinations of other features, finding evidence that age, time spent playing computer games, and expected grade (as in **[11]**) are also predictive of success. Various models they constructed trade off sensitivity for specificity, but their top-performing models had accuracy between 67% and 70%, sensitivity between 78% and 83%, and specificity between 51% and 59%.

Ahadi et al. **[2]** found that students' programming behavior during the first week of class was more important than age, gender, or past programming experience in predicting success. While they achieved accuracy of 71% on validation data from a later semester, their measure of "programming behavior" relies on an instrumented development environment that can track how many steps students take to solve several of the 24 programming assignments given during the first week of their CS1 class. That information will not be available in many CS1 classes.

*Table 1: Predictors of success examined by prior studies. Asterisks * indicate the predictors noted by authors as most significant.*

	[8] [9]	[10] [11]	[12]	[6]	[2]	[7]
Personal Information						
Gender	X	X	*X	X	X	
Age		X	X	*X	X	
Enrollment status		X				
Year of study		X				
Work style preference (individual or group)	X		X			
Part-time employment			X	X		
Intrinsic goal orientation						
Intrinsic questioning				X		
Attribution of success/failure	X			X		
Test anxiety				X		
Major, Interest in Computer Science						
Intended major		X			X	
Strongest background, self-reported: humanities, science, commerce		*X				
Encouragement to pursue computer science	X		X			
"Keenness" to take the class		X				
Intention to enroll in second-year CS courses		X				
Previous Academic and Computer Experience						
Previous programming experience	X	X	X	X	X	
Previous computer usage	X		X			
Computer game playing				*X		
Math background	*X[1]	X				
High school exit exam scores: math			*X	*X		
High school exit exam scores: science			X			
Social media usage				X		
Grade average (university)					*X	
Expectations and Experience in the Course						
Comfort level with programming or programming self-efficacy	*X		*X	*X		
Perceived understanding in the course			*X			
Self-efficacy	X					
Expected difficulty		X				
Expected workload		X				
Expected grade		*X		*X		
Skills Assessment						
Testing "numerical and letter sequencing, arithmetic reasoning, problem translation, logical ability"			X[2]			
Correctness, number of steps to solve early programming problems					*X	
"Clicker" data from in-class questions						*X
Other						
Institution type				X		
Time to complete survey				X		

[1]Number of high school semesters. [2]Some items highly correlated.

Liao et al. **[7]** used students' "clicker" responses through the first three weeks of the course to predict success in a variety of computer science classes, including CS1 courses at two institutions. Like the study by Ahadi et. al. **[2]**, they achieved good results (sensitivity of 76% and specificity of 74% in one course, and 64% and 0.62% respectively, in the other), but required specialized data that may not be available in many introductory programming classes (i.e., student "clicker" responses).

Table 1 illustrates that a wide range of possible predictors have been explored, with few being consistently included in the research studies (gender, age, and previous programming experience). Surprisingly, none of these studies found previous programming experience to be among the most significant predictors. Students' programming self-efficacy was consistently noted as being a good predictor of success in the course **[6]**, **[9]**, **[12]**.

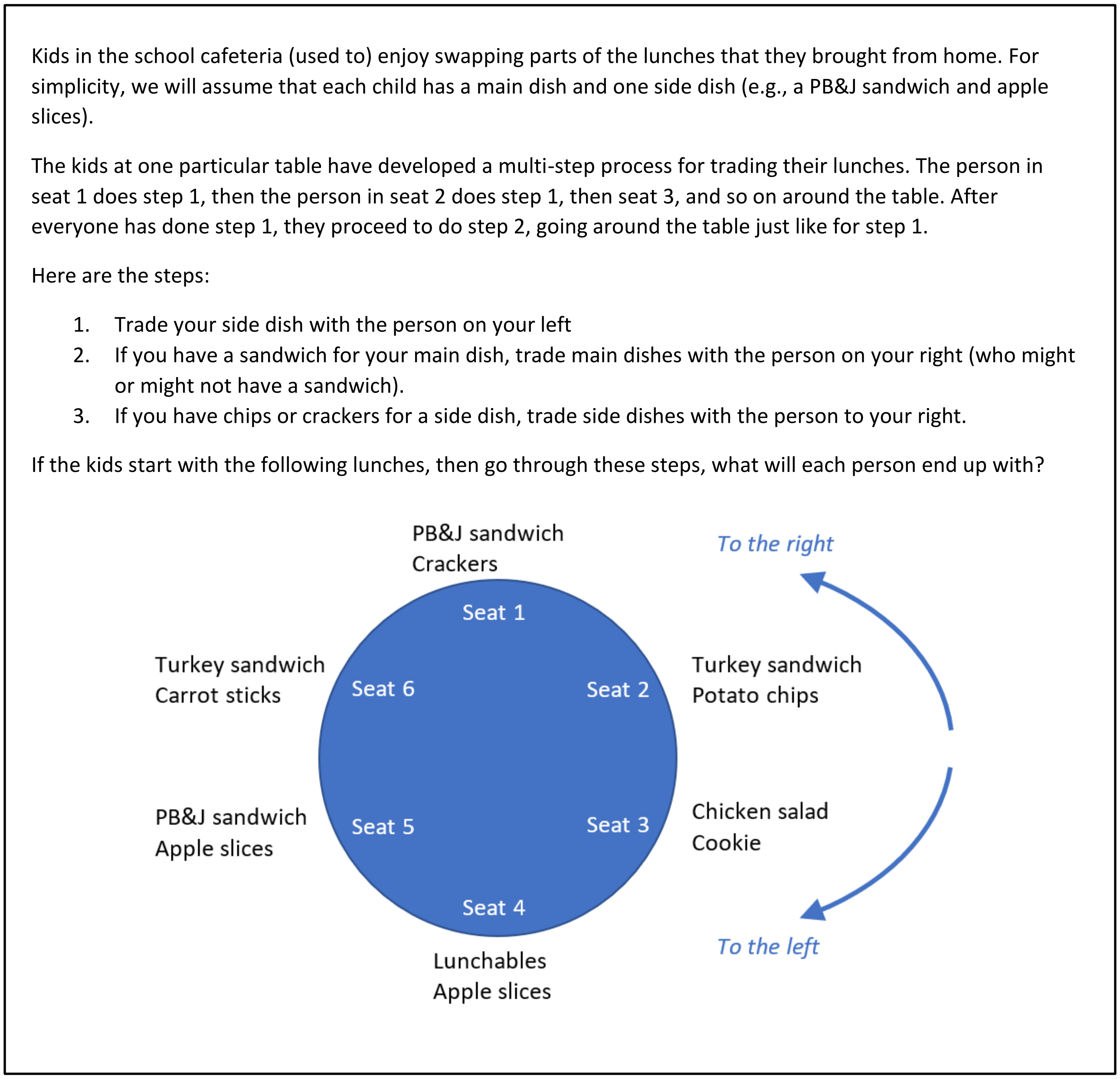
As noted above, the last three studies mentioned here (**[2]**, **[6]**, **[7]**) constitute the recent research that validated results on a separate data set, used a large sample size, and were able to predict student success with high accuracy while still early in a CS1 course. Our work adds to this list of studies, while requiring less extensive data collection. Specifically, Ahadi et al.'s method **[2]** requires an instrumented development environment, Liao et al's **[7]** requires "clicker" data from student responses to in-class questions, and Quille and Bergin's **[6]** requires exit scores from high school math exams in addition to student survey data. Our novel diagnostic quiz can be conducted easily during the first day of class, providing similar accuracy to these other approaches that require more extensive data collection (see **Table 5**). We confirmed this result using a validation set of students, whose data was unused until final model evaluation.

Data Collection

The diagnostic quiz introduced in this study consists of four multi-part questions. These questions did not require any prior knowledge of programming, nor were they directly about programming. There was one question focused on each of the following four skills: logical reasoning (Question 1), checking text for context-dependent rule violations (Question 2), attention to mundane detail over time (Question 3), and stepping through an algorithm (Question 4, in **Figure 1**). Each student was given a score for each question based on how close their answer was to the correct answer and how long they took to complete the question. Some students accidentally took the quiz more than once or were interrupted during their attempt. In these cases, students received the score of their latest attempt but their time to complete the questions was computed as the total time it took them over all their attempts.

Students' scores on two exams during the semester and the final exam were consolidated into one overall exam score, an average of their final exam score and the average semester exam score. That is, the final exam score has equal weight to the average of the exams during the semester. We also collected each students' instructor, course modality (face-to-face or online), and type of major—computer science, other STEM major, or other major.

All students taking the CS1 course in the Fall 2020 and Spring 2021 semesters at a small, undergraduate, liberal arts college in the Midwest United States were asked to complete the diagnostic quiz during the first week of class. Students received course credit for attempting the quiz, even if their answers were not correct.



Kids in the school cafeteria (used to) enjoy swapping parts of the lunches that they brought from home. For simplicity, we will assume that each child has a main dish and one side dish (e.g., a PB&J sandwich and apple slices).

The kids at one particular table have developed a multi-step process for trading their lunches. The person in seat 1 does step 1, then the person in seat 2 does step 1, then seat 3, and so on around the table. After everyone has done step 1, they proceed to do step 2, going around the table just like for step 1.

Here are the steps:

1. Trade your side dish with the person on your left
2. If you have a sandwich for your main dish, trade main dishes with the person on your right (who might or might not have a sandwich).
3. If you have chips or crackers for a side dish, trade side dishes with the person to your right.

If the kids start with the following lunches, then go through these steps, what will each person end up with?

Figure 1: Example question from the diagnostic quiz (Question 4)

There were five sections of the course from which we collected data. Four were in the fall, which had three face-to-face sections and one online section. There was one face-to-face section in the spring. Among the five sections, there were three different instructors, one of whom taught only the online section, while each of the other two taught two face-to-face sections. Overall, there were 135 students enrolled at the beginning of the course, and 119 students who took the final exam. The online section included 16 students, 10 of which were dual-enrolled, while the other 6 were undergraduate students.

Figure 2 shows the distribution of student major category—computer science, other STEM major, or other (non-STEM) major—of the students enrolled at the start of the course. Each of the three categories is well-represented in the data. The students are primarily freshmen, but there are several dual-enrolled high school students, sophomores, and upperclassmen (juniors or seniors) as well (**Figure 2**).

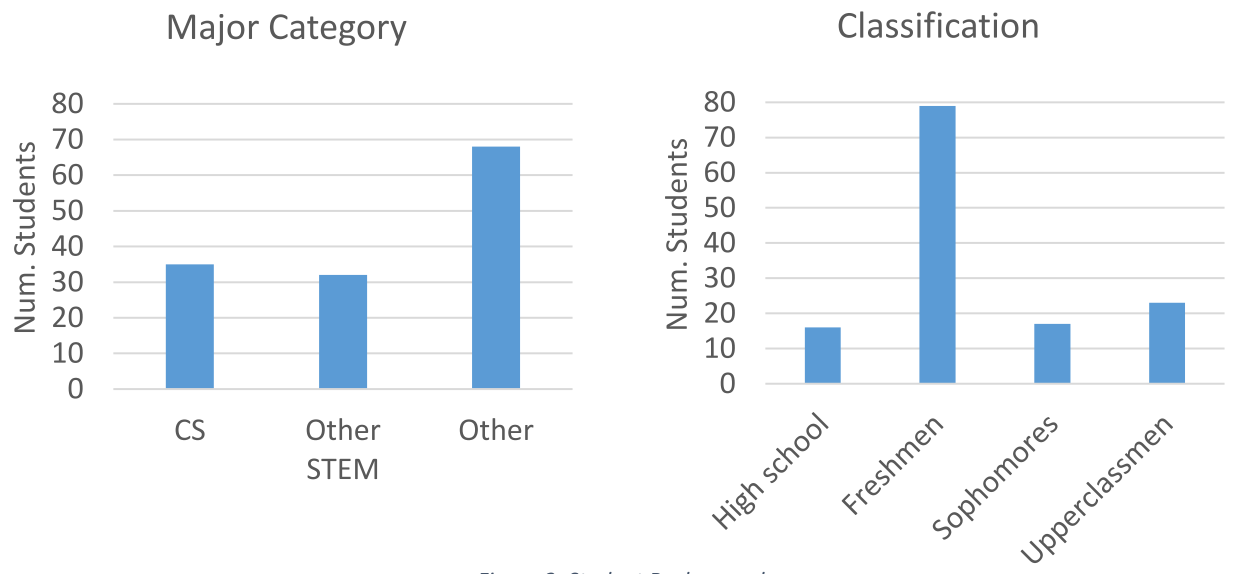


Figure 2: Student Background

Not all of these students had a full set of data, which consists of results from the diagnostic quiz, two exams during the semester, and the final exam. Of the 103 students that completed the quiz, 89 also had semester exam and final exam scores, thereby completing the course. The other 14 students withdrew from the course. Of those who withdrew, nine did so before taking the first exam in the course, while the other five took the first exam but not the second. Those students had an average score of 30.9% on the first exam. Thus, for those students who completed the diagnostic quiz but did not record a full set of exam scores, we imputed an exam average of 30.9%.

Before fitting models to the data, we held out 20% of the students' records for model validation. To represent the range of student outcomes in both the validation set and the training set, the validation set was selected by taking the students in each course section, sorting by the final grade in the course, then picking every fifth student for the validation set, resulting in 30 students total. The remaining 105 students were used for model fitting.

Results: Linear Models for Predicting Exam Scores

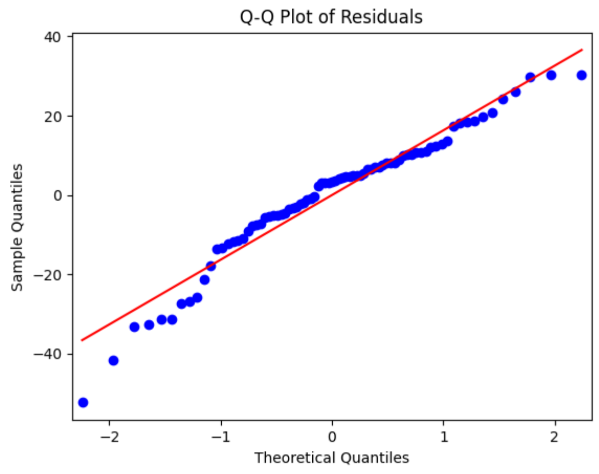
We calculated a multiple linear regression to predict exam score based on the scores from the four individual questions on the quiz. Each question score was scaled to the range [0.0, 1.0]. A significant regression equation was found ($F(4, 74) = 3.786$, $p = 0.00743$), with an R^2 of 0.170 and adjusted R^2 of 0.125. At the $p=0.05$ level of significance, only Question 4 was a significant predictor of exam score ($p=0.005$). Students who earned a perfect score (correct answers and efficient completion) for Question 4 earned an average of 21.7 more points on their exams than those who answered Question 4 completely incorrect. However, these question scores alone explain little of the variance in students' exam scores, indicated by the low R^2 value.

We added two variables to our model to explain some of the additional variance: the instructor and the student's major category (CS, other STEM, or other non-STEM). Each of these categorical variables was coded as a set of n-1 binary variables. For example, there was one binary variable for CS major and one for other STEM major, with no variable for non-STEM majors to avoid singularities when fitting the model with the constant. A significant regression equation was found (F(8, 70) = 4.913, p = 7.68e-05), with an R^2 of 0.360 and adjusted R^2 of 0.286.

The coefficients for the variables are presented in **Table 2**, along with the t-values and p-values for each. At the p=0.05 level of significance, both the student's major and their response to Question 2 on the quiz were significant predictors of exam score. However, the Question 2 coefficient is negative, so higher scores on the quiz indicate lower exam scores, an unexpected result. CS majors score 7 points higher on the exams than other STEM majors, who score 13.3 points higher than non-STEM majors. A Q-Q plot of the residuals indicates that they are reasonably close to normally distributed (**Figure 3**). The plot of residuals vs. predicted value (**Figure 3**) indicates that the model is more accurate for higher exam scores. However, the low R^2 value indicates that there remains substantial variance in the resulting exam scores, even after accounting for instructor and student major.

Table 2: Linear model predicting exam scores.

Independent Variable	**Coefficient**	**t**	**P > |t|**
Question 1 score	-4.5692	-1.006	0.318
Question 2 score	-13.1928	-2.221	**0.030**
Question 3 score	14.9413	1.922	0.059
Question 4 score	13.8953	1.955	0.055
Instructor 1	-1.3739	-0.324	0.747
Instructor 2 (online only)	10.8495	1.586	0.117
CS major	20.2997	3.971	**0.000**
Other STEM major	13.2699	2.641	**0.010**
Constant	62.2468		



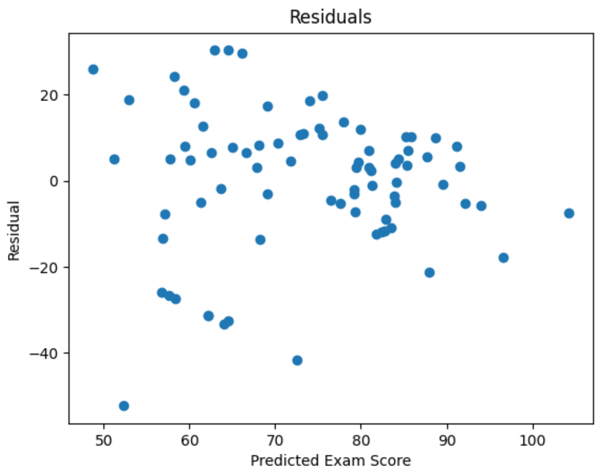


Figure 3: Analysis of linear model residuals. Q-Q plot of residuals (left) and residuals vs. predicted exam score (right).

Results: Classifying Course Outcome

While the previous section described linear models to predict exam score, this section examines a classifier model to predict one of two outcomes for each student. The two outcomes were (1) the student passed the course with an A, B, or C; or (2) the student earned a D or F in the course or withdrew from the course (earned a W). To avoid overfitting the data, we used a single independent variable for the classifier: students' scores on Question 4. We selected Question 4 based on visual inspection of precision-recall curves[1], in addition to the fact that Question 4 was the most significant predictor of exam scores in our initial linear model.

Overall, 23% of the students earned a D, F, or W in the course. When looking at the relationship between Question 4 score and the likelihood of earning D, F, or W in the course, three phases emerged in the data:

- 24 students scored 0.3 or below on Question 4. Of those, 13 earned a D, F, or W in the course (54%).
- 39 students scored between 0.3 and 0.8 on Question 4. Of those, 5 earned a D, F, or W in the course (13%).
- 16 students scored 0.8 or higher on Question 4. None of them earned a D, F, or W in the course (0%).

Students scoring poorly on Question 4 are at a high risk of doing poorly in the course, while students with high scores on Question 4 are virtually certain to pass the course with at least a C. Our classifier model predicts the majority outcome for these three score ranges, leading to the following decision rule: predict D/F/W for students with scores 0.3 or less and predict A/B/C for other students. That classifier produces the confusion matrix in **Table 3**. In this analysis, a D/F/W outcome or high-risk student is considered a "positive" case. The model achieves a precision of 54%, recall of 72%, and accuracy of 80%.

Table 3: Confusion matrix (training data)

Predicted outcome →	*D/F/W*	*A/B/C*
Actual outcome		
D/F/W	13 (true positives)	5 (false negatives)
A/B/C	11 (false positives)	50 (true negatives)

Validating Results

Our work investigating several variations of different kinds of models increases the risk of overfitting the data. To protect against overfit (i.e., inflated) accuracy numbers on the training data, we evaluated our final model—predicting a D/F/W result if and only if a student scores 0.3 or less on Question 4—on the validation data that was not used for model selection. The model achieves a precision of 44%, recall of 67%, and accuracy of 71% on that data, a slight decrease

[1] None of the other questions produced a precision of more than 40% for any value of recall.

from the results on the training data, but not so much as to indicate significant model overfitting. This confirms that Question 4 on the quiz correlates with students' outcomes in the course. The confusion matrix is presented in **Table 4**. Whereas in the training data, several students scored 0.8 or higher on Question 4, in the validation data, only one of the 24 students scored above that mark. As in the training data, scoring above 0.8 guaranteed an outcome of passing the course with an A, B, or C.

Table 4: Confusion matrix (validation data)

Predicted outcome → / **Actual outcome**	*D/F/W*	*A/B/C*
D/F/W	4 (true positives)	2 (false negatives)
A/B/C	5 (false positives)	13 (true negatives)

Table 5: Accuracy of different models

	Accuracy	Sensitivity (Recall)	Specificity	Precision	Neg. predictive value
Predicting if combined final grade/exam question grade is below median, using programming behavior (instrumented development environment) **[2]**					
Training data	91%	93%	88%	89%	93%
Validation data	71%	--	--	--	--
Predicting if exam scores fall in bottom 40%, using "clicker" data **[7]**					
Validation data, Python course	**75%**	76%	**74%**	**66%**	82%
Validation data, Java course	63%	64%	62%	53%	72%
Predicting "weak" programming performance, using survey data, math scores **[6]**					
Validation data, original PreSS model	67%	**78%**	53%	--	--
Training data, support vector machines	70%	79%	57%	--	--
Training data, logistic regression	70%	78%	59%	--	--
Training data, deep neural network	69%	83%	51%	--	--
This study: predicting D, F, or W in course, using one quiz question					
Training data, Java course	80%	72%	82%	54%	91%
Validation data, Java course	71%	67%	72%	44%	**87%**

Measures not reported are marked as --. Results on training data are shown in gray.

Table 5 shows how our model accuracy, sensitivity, specificity, precision, and negative predictive value compare with results from the literature. Our model's accuracy is tied for the highest among those applied to Java courses (the language in our CS1 course), with only the clicker-based model for a Python course reporting higher accuracy. Our model has the advantage of using a much simpler data source: one diagnostic quiz question. For different measures of model quality other than overall accuracy, different models perform the best. Clicker data is the best for specificity and precision, student background information and math test scores are best for sensitivity, and our quiz question is best for negative predictive value. That is, our model is particularly good at identifying students who are *not* high-risk. Future work will examine combining the quiz results with another predictor of student success that is particularly good at identifying students who *are* high-risk (i.e., high sensitivity), such as high school math test scores. These potentially complementary predictors could be combined to improve accuracy while still using a model that can identify high-risk students on the first day of class.

Summary

We introduced a diagnostic quiz designed to predict student success in CS1 without requiring any programming knowledge. This quiz can be administered the first day of class, helping to identify high-risk students very early, before they fall irreparably behind in the course. The most predictive question from that quiz asks students to trace the steps of a simple algorithm and describe the result. Students' scores on that question have a significant positive correlation with their exam scores in the class, and students scoring low on that diagnostic question are more than three times as likely to earn a D, F, or W in the class than the other students in the class. This is true in both the data we used to select the model and the validation data. The model accuracy is comparable to other models in the literature, but it does not require specialized data like instrumented development environments or student "clicker" responses, which will not generally be available in CS1 classes.

References

[1] C. Watson and F. W. Li, "Failure Rates in Introductory Programming Revisited," in *Proceedings of the 2014 Conference on Innovation and Technology in Computer Science Education (ITiCSE 2014)*, 2014.

[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 Annual Conference on International Computing Education Research (ICER 2015)*, 2015.

[3] A. Stefik and S. Siebert, "An Empirical Investigation into Programming Language Syntax," *ACM Transactions of Computing Education,* vol. 13, no. 4, pp. 1-40, November 2013.

[4] P. Denny, A. Luxton-Reilly, E. Tempero and J. Hendrickx, "Understanding the Syntax Barrier for Novices," in *Proceedings of the 16th Annual Joint Conference on Innovation and Technology in Computer Science Education (ITiCSE 2011)*, 2011.

[5] N. C. Brown and A. Altadmri, "Investigating Novice Programming Mistakes: Educator Beliefs vs. Student Data," in *Proceedings of the Tenth Annual Conference on International Computing Education Research (ICER 2014)*, 2014.

[6] K. Quille and S. Bergin, "CS1: How will they do? How can we help? A decade of research and practice," *Computer Science Education,* vol. 29, no. 2-3, pp. 254-282, 2019.

[7] S. N. Liao, D. Zingaro, K. Thai, C. Alvarado, W. G. Griswold and L. Porter, "A Robust Machine Learning Technique to Predict Low-Performing Students," *ACM Transactions on Computing Education,* vol. 19, no. 3, 2019.

[8] B. C. Wilson and S. Shrock, "Contributing to Success in an Introductory Computer Science Course: A Study of Twelve Factors," *SIGCSE Bulletin,* vol. 3, no. 1, pp. 184-188, 2001.

[9] B. C. Wilson, "A Study of Factors Promoting Success in Computer Science Including Gender Differences," *Computer Science Education,* vol. 12, no. 1-2, pp. 141-164, 2002.

[10] N. Rountree, J. Rountree and A. Robins, "Predictors of Success and Failure in a CS1 Course," *SIGCSE Bulletin,* vol. 34, no. 4, pp. 121-124, 2002.

[11] N. Rountree, J. Rountree, A. Robins and R. Hannah, "Interacting Factors That Predict Success and Failure in a CS1 Course," *SIGCSE Bulletin,* vol. 36, no. 4, pp. 101-104, 2004.

[12] S. Bergin and R. Reilly, "Programming: Factors That Influence Success," *SIGCSE Bulletin,* vol. 37, no. 1, pp. 411-415, 2005.