Exploring the Value of Different Data Sources for Predicting Student Performance in Multiple CS Courses

Soohyun Nam Liao
University of California, San Diego

Daniel Zingaro
University of Toronto Mississauga

Christine Alvarado
University of California, San Diego

William G. Griswold
University of California, San Diego

Leo Porter
University of California, San Diego

ABSTRACT

A number of recent studies in computer science education have explored the value of various data sources for early prediction of students' overall course performance. These data sources include responses to clicker questions, prerequisite knowledge, instrumented student IDEs, quizzes, and assignments. However, these data sources are often examined in isolation or in a single course. Which data sources are most valuable, and does course context matter? To answer these questions, this study collected student grades on prerequisite courses, Peer Instruction clicker responses, online quizzes, and assignments, from five courses (over 1000 students) across the CS curriculum at two institutions. A trend emerges suggesting that for upper-division courses, prerequisite grades are most predictive; for introductory programming courses, where no prerequisite grades were available, clicker responses were the most predictive. In concert, prerequisites and clicker responses generally provide highly accurate predictions early in the term, with assignments and online quizzes sometimes providing incremental improvements. Implications of these results for both researchers and practitioners are discussed.

CCS CONCEPTS

• **Social and professional topics → Computing education**; **Computer science education**; **CS1**;

KEYWORDS

machine learning, prediction, student outcomes, low-performing students, CS1, CS2, Data Structures, Architecture

ACM Reference Format:
Soohyun Nam Liao, Daniel Zingaro, Christine Alvarado, William G. Griswold, and Leo Porter. 2019. Exploring the Value of Different Data Sources for Predicting Student Performance in Multiple CS Courses. In *Proceedings of the 50th ACM Technical Symposium on Computer Science Education (SIGCSE '19), February 27-March 2, 2019, Minneapolis, MN, USA.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3287324.3287407

SIGCSE '19, February 27-March 2, 2019, Minneapolis, MN, USA

ACM ISBN 978-1-4503-5890-3/19/02…$15.00
https://doi.org/10.1145/3287324.3287407

1 INTRODUCTION

Failure rates in computer science courses can be high, both in introductory computing [24] and in follow on courses [15]. This problem is exacerbated by current high enrollments [6] because growing class sizes make it increasingly difficult for instructors to identify who among their students is in trouble. As a result, the computing education research community has been studying automated means of identifying at-risk students early in the term so that instructors might be able to intervene. Some prior studies focused on introductory courses, looking at relationships between student outcomes and pre-class experiences or characteristics (SAT score, gender, prior programming experience, achievement goals etc.) [12, 14, 19, 22, 26, 29]. Other work has leveraged educational technology to collect student data, such as programming behavior, types of questions asked on online forums, and clicker data from Peer Instruction [17, 21, 23]. Much of this work has focused on understanding characteristics of low- or high-performing students based on data from a single term, though a few recent studies have predicted overall course performance across terms by using machine learning models built with clicker data [11] and assignments [3, 5].

One limitation of previous prediction studies is that they largely focus on introductory programming courses, leaving a gap in our understanding of the factors that might be predictive for later courses. In addition, many of those studies focus on single data sources, rather than the ways in which multiple sources might work in concert.

In this study, we collect grades in prerequisite computer science courses (when applicable), assignments, online quizzes, and clicker data from five different CS courses (both lower- and upper-division) across two separate institutions. In each case, we examine the value of each of these data sources in building a predictive model, where the model is trained on data from one term and then applied to students in a subsequent term. We find, as one might expect, that for upper-division courses, grades in prerequisite courses are particularly strong predictors. Adding clicker data improves the model in general for all five courses, including two introductory programming courses where prerequisite CS grades are not available. By contrast, assignments and online quizzes improve the model only marginally, compared to prerequisite courses and clicker data. In concert, prerequisite grades and clicker data provide particularly accurate predictions. As such, instructors wishing to identify low-performing students early in the term may wish to prioritize the collection of these data sources.

Table 1: Dataset Details

ID	Course Name	Inst†	Size‡	Course Topics	Prerequisites
CS1P	CS1 in Python	A	192 / 142	variables, datatypes, functions, conditionals, loops, and sorting algorithms	None
CS1J	CS1 in Java	B	373 / 374	variables, methods, conditionals, loops, and objects	None
CS2J	CS2 in Java	B	169 / 176	arrays, lists, stacks, queues, sorting algorithms, and runtime analysis	CS1
DataStr	Advanced Data Structures	B	197 / 191	trees, balanced search trees, hashtables, tries, Huffman coding, and graph search	CS2, Programming Tools Lab, Discrete Mathematics, Computer Organization
Arch	Introductory Computer Architecture	B	339 / 266	ISA, performance, single/multi-cycle processors, pipelining, and caches	Computer Organization, Digital Logic, Digital Logic Lab

† Institution ‡ Size of Training Set / Test Set

Table 2: Data Sources Used in the Present Study

ID	Description	Preprocessing Method	CS1P	CS1J	CS2J	DataStr	Arch
p	Final grades from prerequisite courses	Numerical grade			✓	✓	✓
c	In-class clicker correctness	Average correctness	✓	✓	✓	✓	✓
a	Take-home assignment grades	Average score		✓	✓	✓	✓
q	Online quiz grades	Average score	✓	✓	✓	✓	✓

2 BACKGROUND

Accurately predicting student performance in CS courses is a long-standing goal of CS education researchers [20]. Given the elevated failure rates often reported in CS courses [24], the hope is to be able to identify at-risk students in a timely manner, so that there is opportunity to alert students or help them make progress.

Much of the relevant work relies on static factors that can be measured at the start of a course but that cannot be easily changed. Such factors include gender, prior experience, achievement goals, and self-efficacy [19, 26, 29]. While some success has been met using such approaches [18], correlations between these factors and course outcomes tend to be low or do not consistently replicate [25].

Fortunately, we can also avail ourselves of data generated by modern educational technology. In contrast to static data, this dynamic data is generated as the course unfolds, and provides a more nuanced understanding of how students progress through and interact with course material. One strand of this research leverages Peer Instruction (PI) clicker data, using student in-class responses from early in the term to predict course outcomes [11, 17]. Another strand uses instrumented software to capture student process and behavioral data, such as total keypresses when programming [3], time and changes between compilations [25], and time spent with particular elements in online modules [10].

We note two related gaps in the body of literature summarized above. First, the extant research push is largely focused on making predictions in CS1 courses, with very little action in other courses [8]. Second, students taking CS1 courses are very likely in their first year of study, so such research cannot include student grades in prerequisite courses. Some work suggests that grades in prerequisite courses are useful for predicting student progression through a CS degree program [4], but that work does not consider dynamic predictors of performance. In summary, while in CS1 the efficacy of static predictors tends to pale in comparison to that of dynamic predictors, it is not known to what extent prerequisite grades and dynamic predictors may work together to predict outcomes in more advanced courses. The present paper addresses these research gaps: we study a variety of courses in the undergraduate CS curriculum, and (for more advanced courses) make predictions using both static and dynamic predictors.

3 METHOD

Our work was guided by two central research questions:

- **RQ1:** What is the value of each of our data sources for predicting student performance in computing?
- **RQ2:** How do prediction accuracies improve as we use multiple data sources?

Given these research questions, we provide background information on our datasets and how we create models to predict student performance and evaluate those predictions.

3.1 Our Datasets

We obtained our data from five courses across the CS curriculum at two North American research-focused institutions with institutional approval to study human subjects. Table 1 provides course details, including course topics and numbers of students in each dataset. Note that for each course we have two terms' worth of data, one used for training and the other for testing. This cross-term modeling approach captures authentic variation in natural teaching environments, including changes in courses and exams [3, 11]. Our data includes two CS1 courses, one from each institution; these courses were taught completely separately and not coordinated across institutions.

Table 2 describes the modeling features used for the present study. We selected relatively common course components within

these five courses to predict student performance: course prerequisites, clicker responses, assignments, and online quizzes. Course prerequisites (see the Prerequisites column in Table 1) are not available for CS1, as it is the first CS course taken by students at these institutions. Clicker data was available for all five courses, as all instructors used Peer Instruction [7] in class. Peer Instruction involves the use of conceptually-rich multiple-choice questions to which students respond individually and then in groups. The questions are designed to target student difficulties and misconceptions, and are informed by teaching experience, research literature, and available question banks. Assignment grades were incorporated into four of the models; no assignment data was available for CS1P, because the first assignment in that course was due later in the term. Arch students completed individual assignments. In the other courses, pair programming was required (CS1J) or encouraged (CS1P, CS2J, DataStr). Online quiz data was available in all courses through pre-class reading quizzes; CS2J also had review quizzes assessing what students learned in the past week.

3.2 Model Generation

Our model generation process can be divided into four major steps: defining early prediction, defining low-performing students, preprocessing the data, and training a model. We used the glm R function [2] to generate and evaluate prediction models.

Defining early prediction. Similar to prior studies, we produce predictions using only data collected during the first 3 weeks of each term [11, 17]. By making predictions this early in the term, the belief is that potential interventions may be possible even before the first midterm exam.

Defining low-performing students. We defined low-performing students to be those who are ranked in the bottom 40% on the final exam. This is based on prior literature [11] where instructors selected the bottom 40% to be the group of students likely in need of assistance.

Preprocessing. We preprocessed each data source as indicated in the Preprocessing Method column of Table 2. Each data source was distilled into a single value per student, except for prerequisites where we used one value per prerequisite course grade. We collected final letter grades from the prerequisite courses, and then converted these letter grades into numerical grades on a 0-4.33 scale, so A+ became 4.33, A became 4.0, A- became 3.67, and so on. For clicker data, we used records of student responses to each clicker question, following prior work [17] stating that student clicker correctness is correlated with final exam performance. Specifically, we assigned -1, 0, and 1 for incorrect, no response, and correct, respectively, and calculated the average of all clicker questions for each student. Similarly, for assignments and quizzes, we calculated average grades, after scaling the maximum possible score of each assessment item to be 1.0 to correct for different maximum possible scores. As a final step, we normalized each data feature to have a normal distribution.

Training. We trained a Logistic regression model [27] to perform binary classification (low-performing vs. high-performing) for each course. We selected logistic regression after exploring a number of different models for accuracy, simplicity, and ability to work well with a small number of input features. The model was trained on data from an earlier term with the data sources as features and with the binary outcome labels of low-performing or high-performing based on final exam score.

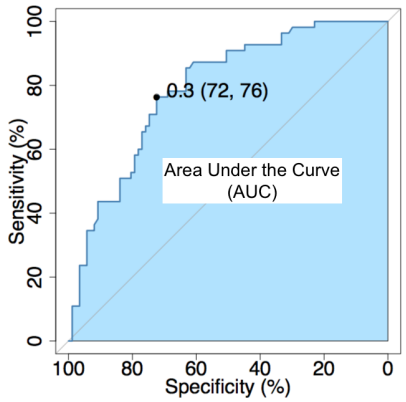


Figure 1: An Example ROC Curve

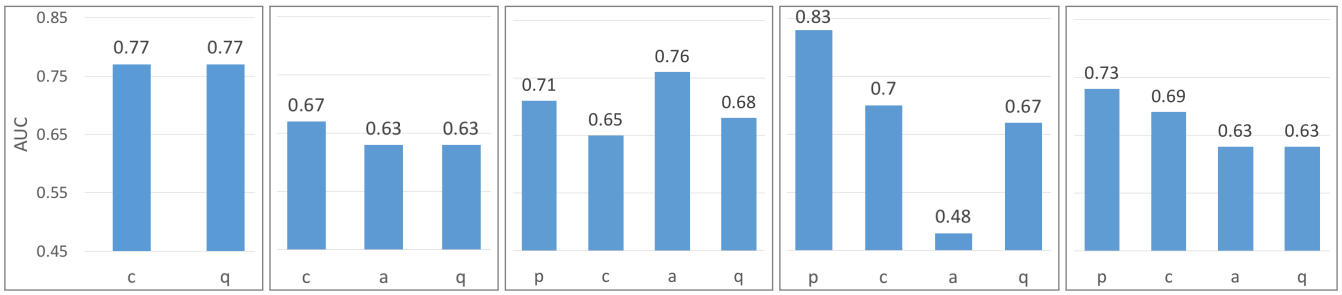
3.3 Model Analysis

Given the model created using the previous term of data (training set), we make predictions on the subsequent term (test set) and evaluate those predictions using a common graph in machine learning called a Receiver Operating Characteristic (ROC) curve, which plots the performance of a model as its parameters are varied. An example ROC curve is provided in Figure 1. The x-axis plots how accurately a model identifies high-performing students (specificity) and the y-axis plots how accurately a model identifies low-performing students (sensitivity). The output of the logistic regression model is a number between 0 and 1 representing the probability that a particular student is low-performing. We can trade off the specificity and sensitivity of our model by varying the probability threshold we use to decide whether a student is classified as high- or low-performing. This trade-off is what is shown in the ROC curve; each point on the curve corresponds to a different selected value for the probability threshold. For example, the point corresponding to a probability threshold of 0.3 is shown in Figure 1. This example point, for this model at this threshold, produces a specificity of 72% and a sensitivity of 76%.

When evaluating the overall modeling performance, the Area Under the Curve (AUC) on an ROC curve is commonly used as it captures, in effect, how close to an ideal curve is provided by the model. Figure 1 visualizes the AUC in blue. Overall, a high AUC (closer to 1.0) implies a better model as it is better capable of capturing sensitivity and specificity.

3.4 Analysis of Multiple Sources

Ultimately, we use a combination of multiple data sources to train and test the model using the AUC described above. We would like to know whether adding a data source leads to improvement in the model. However, one cannot directly determine whether a model is statistically better than another based on resultant AUC curves. As such, we begin by determining whether adding a specific data source is statistically significant, and to do so, we run a likelihood ratio test [13] (using the lrtest R package [1]). This test compares two models, before and after adding a specific data source, and



(a) CS1P (b) CS1J (c) CS2J (d) DataStr (e) Arch

Figure 2: AUC When Modeling with Individual Data Sources

returns a p-value to show whether adding that data source statistically changes the generated model. We define statistical significance at the $p < 0.05$ level. There is an important distinction between this analysis and the model analysis from Section 3.3 in that likelihood ratio tests are performed on the training set data only, whereas evaluation with an AUC occurs on a test set. This has the important consequence for our evaluation that a statistically significant improvement in the model's accuracy for the training set, evidenced by the likelihood ratio, may not necessarily translate to an improved result for the AUC on the test set (due to potential overfitting, changes across terms, etc.).

4 RESULTS

We first examine the value of different data sources (RQ1) and then explore how model accuracy improves as we use multiple data sources (RQ2).

4.1 RQ1: Value of Different Data Sources

To determine which data sources are most valuable, first we examine the resultant AUC of each data source when used in isolation and then we determine whether adding each source statistically significantly improves the model using likelihood ratio analysis.

4.1.1 Value of Data Sources in Isolation. Figure 2 provides the resultant AUC for the test set when using each data source individually, collected from the five courses. Recall that CS1 courses do not have any course prerequisites at our institutions, so they do not include results from prerequisite data. Moreover, CS1P did not have any assignments due within the first three weeks of the term, so Figure 2a does not have results on assignments.

On average, prerequisite data (p in Figure 2) returns the best AUC (0.76) out of all data sources. This implies that for courses beyond CS1 (even as early as CS2), prerequisite grades are a powerful predictor of student outcomes. The next highest average AUC (0.70) is provided by clicker data (c), confirming prior work that this data source as a valuable predictor of student outcomes [11, 17]. Online quizzes and assignments provide similar AUC for CS1J and Arch; assignments are better for CS2J; and quizzes are better for DataStr. Although the Average AUC is 0.68 and 0.63 for online quizzes and assignments, respectively, the inconsistencies between courses means no clear trend between online quizzes and assignments has emerged. Overall, the trend is that prerequisites are more valuable than clickers and clickers are more valuable than assignments or quizzes. In DataStr, we see that assignments have effectively no predictive power, as the associated AUC is worse than random guessing (i.e., AUC < 0.5). This is further discussed in Section 5.

Table 3: Statistical Significance of Adding Data Sources

	CS1P	CS1J	CS2J	DataStr	Arch
'p' only	N/A	N/A	***	***	***
Add 'c' to 'p'	***	***	*		***
Add 'a' to 'pc'	N/A		*	*	***
Add 'q' to 'pca'		***			***

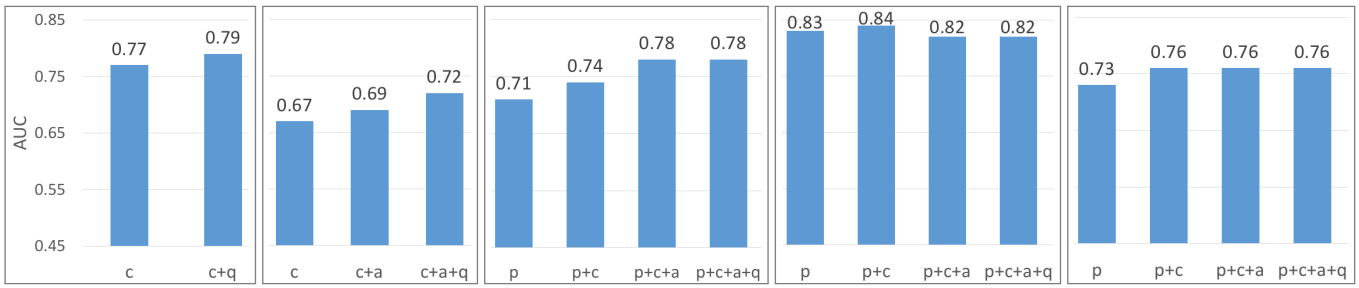
$^{*}p < 0.05$, $^{**}p < 0.01$, $^{***}p < 0.001$

4.1.2 Value of Data Sources Relative to Each Other. Next, we determine which data sources add statistical significance to the model accuracy. For example, although online quizzes in isolation have some predictive power, perhaps the variance explained by that data source is already better explained by the clicker data.

To perform this evaluation, we test how well the generated model performs as we add one additional data source at a time. Specifically, the initial model with one data source is compared against a null model with only the intercept term to determine the statistical significance of the initial model. We then determine whether adding an additional data source improves the model for the training set.

Likelihood ratio analysis depends heavily on the order that one adds the data sources. For example, if one adds data sources ordered from least predictive to most predictive, then the least predictive are more apt to be deemed significant. As such, we used the common approach in likelihood ratio analysis of adding data sources from most to least predictive (based on our results from Section 4.1.1). We therefore start with prerequisite data, then sequentially add clicker, assignment, and online quiz data. (We additionally experimented with a variety of different orders, and the overall conclusions remained the same.)

Table 3 shows whether adding each data source makes any statistically significant changes in the generated model. The initial model based only on either prerequisite data (CS2J, DataStr, Arch) or clicker data (CS1P, CS1J) is always meaningful compared to the null model. Clicker data is statistically significant overall, not only when compared to the null model (CS1P, CS1J), but also when added to the model with prerequisite data only (CS2J, DataStr, Arch). Also, adding assignment data is statistically significant in all courses but CS1J. Lastly, adding online quiz data on top of prerequisite, clicker,



(a) CS1P (b) CS1J (c) CS2J (d) DataStr (e) Arch

Figure 3: AUC When Modeling with Combination of Data Sources

and assignment data statistically significantly improves only the model for two courses (CS1J, Arch).

Overall, this statistical analysis confirms findings from Section 4.1.1 that the value of our data sources, from most to least, is as follows: prerequisites, clickers, assignments, and online quizzes.

4.2 RQ2: Combining Different Data Sources

Similar to Section 4.1.2, we examine how the model improves as we add additional data sources. Rather than focusing on the statistical significance of the model for the training set, however, here we look at the resultant AUC for the test set when building a model with a combination of data sources. Because the statistical analysis in Section 4.1.2 was on the training set only, and the analysis here is for the training and test sets combined, we again note that data sources found "significant" in Section 4.1.2 may not translate to improved accuracy here.

Figure 3 shows how AUC changes as we add more data to the model. Overall, AUC improves as we add more data, except in DataStr and Arch. In those two cases, the model with both prerequisites and clickers performs quite well, but the additional data does not improve (and sometimes hurts) the model accuracy. The fact that prerequisites and clickers are more meaningful than the other sources for DataStr and Arch was also noticeable when we evaluated each data source individually (see Figure 2d and Figure 2e). One might suppose that there is a ceiling to the possible prediction accuracy in our courses, given different student demographics, different topic orderings, and different exams each term. It is unclear whether such a ceiling effect is limiting further prediction accuracy for DataStr.

For CS1P, CS1J, and CS2J, the model generally improves as more data sources are added (with the exception of online quizzes for CS2J). This finding may suggest that the multiple data sources collectively improves accuracy; but one might also argue that accuracy is generally quite high already with just two data sources, and that the additional sources only provide marginal improvements.

5 DISCUSSION

This section discusses implications of our results and provides suggestions for instructors wishing to use various data sources to identify low-performing students. We also explain why we selected these data sources and possible threats to validity of our results.

5.1 Implications

Our work adds to our understanding of how to predict student outcomes by examining the value of different data sources. To our knowledge, this is the first work that uses various data sources, collected in multiple courses across the CS curriculum, from multiple institutions, to perform cross-term predictions.

Prerequisite Grades. Of the four data sources, prerequisite course data is the most predictive. This predictive power relates to the number of prerequisite courses for each course. For instance, CS2J had only one prerequisite (CS1), while Arch and DataStr had two and three, respectively; in turn, the AUC of the prerequisite-only model of CS2J is the lowest, followed by Arch, then by DataStr.

The predictive power of prerequisite grades is unsurprising when one considers that a final grade in a course represents a student's overall academic performance throughout the term. Specifically, the final grade includes direct observations of a student's class performance, often including attendance, assignment grades, midterm exam grades, final exam grades, and so on. Moreover, it likely also captures some underlying factors that lead to student success on these assessments (motivation, study strategies, background knowledge, etc.). Lastly, a single term of data represents a full 10-12 weeks (at our institutions) of student performance. Recall that our other data sources were collected over a comparably brief three-week period. Given these differences in duration, it is somewhat astonishing that our other sources are as powerful as they are.

One important benefit of prerequisites being so predictive for upper-division courses is that this data is available at the beginning of a term, before collecting any data from the course itself. This suggests the possibility of offering support before the course officially begins. For example, refresher sessions could be offered, and students could be emailed to encourage participation.

Clicker Responses. Clicker data is the second best predictor of student performance, supporting prior findings that use clicker data for prediction [11, 17]. Peer Instruction questions may be particularly predictive as they are designed to address core concepts and highlight common misconceptions. In addition, these questions in our courses are graded on participation rather than correctness [16]. This gives students the freedom to represent their understanding accurately, as there is no grade incentive to solicit assistance. For those who have adopted Peer Instruction in their courses, clicker data is automatically available. Scoring clicker questions presents

a small overhead, but is likely performed as part of teaching the course. Those not using Peer Instruction, by contrast, could use multiple-choice questions as part of online quizzes instead [11].

Assignments and Online Quizzes. Our AUC results imply that the additional value of assignments or online quizzes beyond prerequisite and clicker data is relatively small. Two exceptions are the addition of assignment and quiz data to CS1J, and the addition of assignment data to CS2J. In each of these courses, neither the clicker-only model (for CS1J) nor the prerequisite-only model (for CS2J) was as predictive as for the other courses, so this may have allowed more room for improvement from additional data sources. Why prerequisite data and clicker data were less predictive for these two classes is enigmatic, and could stem from a variety of factors (key differences in exams across terms, multiple CS1 courses feeding a single CS2 course, quality of clicker questions, etc.). We leave this as a topic for future inquiry. In addition to the AUC results, the likelihood ratio test results showed that online quiz data does not statistically significantly improve the generated model for most courses. Thus, if an instructor already has prerequisite course grades and clicker data, it may not be worth the additional effort to add assignments and online quizzes to a prediction model.

The poor performance of assignments as a predictor of success in DataStr was initially surprising. However, upon reflection, we observed that there were only two DataStr assignments given in the first three weeks of the term: one was a pre-class survey and the other was a programming assignment that is easier than other assignments given later in the course. The relative low difficulty of the programming assignment (and subsequently high grades on average) may explain the lackluster performance of assignments as a predictor of overall course performance.

Combining Data Sources. Overall, our results demonstrate that prediction accuracy often improves when adding additional data sources. When choosing to collect data, instructors may wish to prioritize the more predictive data sources of prerequisites and clicker responses. Instructors may consider augmenting the model with other data sources, with the proviso that accuracy gains may be limited.

5.2 Ease of Collecting Data Sources

We selected the data sources used in this study as they are generally easy to obtain. Three of the sources (clickers, assignments, and quizzes) are often already part of course assessments. The remaining source (prerequisites) is already available in the school database (although we recognize that access to this data may be limited depending on institution). One might be able to obtain grades in prerequisites through a start-of-term survey of students, though there may be concerns about reminding students of previously poor performance at the outset of a new course. These sources differ from some of those used in prior studies, where data was gathered using an extra instrument, such as a survey, pre- or post-test, or programming IDE [3, 29]. In contrast, we anticipate no significant burden for instructors to use the data sources in our study other than retrieving the required data and, in the case of clicker questions, marking the correct answer if not marked.

Prior work has demonstrated a relationship between prior programming knowledge and success in CS1 [9, 28]. We considered using a measure of prior knowledge as an alternative to prerequisite grades but did not do so because the data is a burden for instructors to collect, and because collecting accurate data requires asking students to solve programming-related questions that might be intimidating to those lacking prior knowledge. An alternative to explore in the future may be APCS grades or high school math grades, but in our experience instructors rarely have access to such data.

5.3 Threats to Validity

Although we included multiple courses, both lower- and upper-division, the data from the upper-division courses (DataStr and Arch) were collected only at a single institution. That said, these upper-division courses were taught by different instructors, so our results are not biased to a single instructor. In addition, we used logistic regression as it provided strong prediction accuracy for a binary outcome given only a few features per student. To arrive at that modeling technique, we tried a variety of models commonly used in machine learning (e.g., SVM and random forest) and selected logistic regression based on its consistently strong results. As our findings are based on using logistic regression, our results may be biased by our choice of modeling technique.

6 CONCLUSION

Our work analyzes the predictability of different data sources and how prediction performance improves when combining multiple data sources. We collected students' prerequisite course grades, clicker correctness, assignment grades, and online quiz grades, and applied logistic regression to produce cross-term predictions (as one might do to authentically predict in practice). Overall, our results demonstrate that prerequisite course grades (when available) predict student performance most strongly, followed by clicker data, assignments, and online quizzes. Combining prerequisite and clicker data often improves the model's prediction accuracy; however, adding assignments and quizzes to that already accurate model often does not increase the accuracy further.

This work adds to our knowledge of predicting outcomes in computing by comparing different data sources collected from the same population to determine which are most meaningful. The data sources selected for the work are those generally available to instructors, reducing the barrier for instructors to adopt these techniques. We note also that prerequisite and clicker data are available early: prerequisite data before the start of class, and clicker data from the first lecture (though several weeks of clicker data are likely required for accurate modeling [11]). The findings are generally consistent across five different computing courses (including both lower- and upper-division courses) and are collected from two institutions. Overall, our findings suggest that instructors wishing to identify low-performing students should leverage prerequisite course grades and clicker questions to make that determination.

7 ACKNOWLEDGEMENTS

We greatly appreciate the reviewers for their helpful feedback. This work was supported in part by NSF award 1712508.

REFERENCES

[1] https://www.rdocumentation.org/packages/lmtest/versions/0.9-36/topics/lrtest.

[2] https://www.rdocumentation.org/packages/stats/versions/3.5.1/topics/glm.

[3] 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 11th Conference on International Computing Education Research*, pages 121–130, 2015.

[4] A. Anthony and M. Raney. Bayesian network analysis of computer science grade distributions. In *Proceedings of the 43rd Technical Symposium on Computer Science Education*, pages 649–654, 2012.

[5] K. Castro-Wunsch, A. Ahadi, and A. Petersen. Evaluating neural networks as a method for identifying students in need of assistance. In *Proceedings of the 48th Technical Symposium on Computer Science Education*, pages 111–116, 2017.

[6] CRA Enrollment Committee Institution Subgroup. Generation CS: Computer science undergraduate enrollments surge since 2006. Computing Research Association, 2017.

[7] C. H. Crouch and E. Mazur. Peer instruction: Ten years of experience and results. *American Journal of Physics*, 69, 2001.

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

[9] D. Hagan and S. Markham. Does it help to have some programming experience before beginning a computing degree program? In *ACM SIGCSE Bulletin*, volume 32, pages 25–28, 2000.

[10] L. Leppänen, J. Leinonen, P. Ihantola, and A. Hellas. Predicting academic success based on learning material usage. In *Proceedings of the 18th Conference on Information Technology Education*, pages 13–18, 2017.

[11] S. N. Liao, D. Zingaro, M. A. Laurenzano, W. G. Griswold, and L. Porter. Lightweight, early identification of at-risk CS1 students. In *Proceedings of the 12th Conference on International Computing Education Research*, pages 123–131, 2016.

[12] A. Lishinski, A. Yadav, J. Good, and R. Enbody. Learning to program: Gender differences and interactive effects of students' motivation, goals, and self-efficacy on performance. In *Proceedings of the 12th Conference on International Computing Education Research*, pages 211–220, 2016.

[13] M. Natrella. NIST/SEMATECH e-handbook of statistical methods. http://www.itl.nist.gov/div898/handbook, 2010.

[14] C. G. Petersen and T. G. Howe. Predicting Academic Success in Introduction to Computers. *Association for Educational Data Systems*, 12(4):182–191, 1979.

[15] L. Porter, C. Bailey Lee, and B. Simon. Halving fail rates using peer instruction: a study of four computer science courses. In *Proceedings of the 44th technical symposium on Computer science education*, pages 177–182, 2013.

[16] 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 Technical Symposium on Computer Science Education*, pages 358–363, 2016.

[17] L. Porter, D. Zingaro, and R. Lister. Predicting student success using fine grain clicker data. In *Proceedings of the 10th Conference on International Computing Education Research*, pages 51–58, 2014.

[18] K. Quille and S. Bergin. Programming: Predicting student success early in CS1. a re-validation and replication study. In *Proceedings of the 23rd Conference on Innovation and Technology in Computer Science Education*, pages 15–20, 2018.

[19] V. Ramalingam, D. LaBelle, and S. Wiedenbeck. Self-efficacy and mental models in learning to program. *SIGCSE Bulletin*, 36:171–175, 2004.

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

[21] M. M. T. Rodrigo, R. S. Baker, M. C. Jadud, A. C. M. Amarra, T. Dy, M. B. V. Espejo-Lahoz, S. A. L. Lim, S. A. Pascua, J. O. Sugay, and E. S. Tabanao. Affective and behavioral predictors of novice programmer achievement. In *Proceedings of the 14th Conference on Innovation and Technology in Computer Science Education*, pages 156–160, 2009.

[22] V. L. Sauter. Predicting computer programming skill. *Computers & Education*, 10(2):299–302, 1986.

[23] M. Vellukunnel, P. Buffum, K. E. Boyer, J. Forbes, S. Heckman, and K. Mayer-Patel. Deconstructing the discussion forum: Student questions and computer science learning. In *Proceedings of the 48th Technical Symposium on Computer Science Education*, pages 603–608, 2017.

[24] C. Watson and F. W. Li. Failure rates in introductory programming revisited. In *Proceedings of the 19th Conference on Innovation and Technology in Computer Science Education*, pages 39–44, 2014.

[25] 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 Technical Symposium on Computer Science Education*, pages 469–474, 2014.

[26] 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.

[27] J. R. Wilson and K. A. Lorenz. Short history of the logistic regression model. In *Modeling Binary Correlated Responses using SAS, SPSS and R*, pages 17–23, 2015.

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

[29] D. Zingaro, M. Craig, L. Porter, B. A. Becker, Y. Cao, P. Conrad, D. Cukierman, A. Hellas, D. Loksa, and N. Thota. Achievement goals in CS1: Replication and extension. In *Proceedings of the 49th Technical Symposium on Computer Science Education*, pages 687–692, 2018.