A case study of early performance prediction and intervention in a computer science course

Prof. Mariana Silva, University of Illinois at Urbana-Champaign

Mariana Silva is a Teaching Assistant Professor in Computer Science at the University of Illinois at Urbana-Champaign. She has been involved in large-scale teaching innovation activities, such as the development of online course content and assessments for the mechanics course sequence in the Mechanical Science and Engineering Department and the numerical methods class in Computer Science. Silva is currently involved in two educational projects involving the development of online assessments for computer-based testing and creation of collaborative programming activities for computer science classes. She is also involved in a project that aims to create a software that facilitates collaborative problem-solving activities in classrooms, through which both the instructors and students learn more about collaboration skills. Silva is very passionate about teaching and improving the classroom experience for both students and instructors. She has been included in the List of Teachers Ranked as Excellent five times and has received the Engineering Council Outstanding Advisor Award every year since 2014.

Eric G Shaffer, University of Illinois at Urbana-Champaign

Eric Shaffer is a Teaching Assistant Professor in the Department of Computer Science. He teaches a revolving set of courses including Virtual Reality, Computer Graphics, and Scientific Visualization. In addition to teaching, he has done research in the areas of scientific computing, computer graphics and visualization. He has served as a PI or co-PI on grants from a variety of sponsors, including ExxonMobil, the Boeing Company, Caterpillar, and the US Department of Energy. He holds an MS in Computer Science from the University of Minnesota Twin Cities and a BS and PhD in Computer Science from the University of Illinois at Urbana-Champaign.

Nicolas Nytko, University of Illinois at Urbana - Champaign

Nicolas Nytko is a M.S. student in the department of Computer Science at the University of Illinois at Urbana-Champaign. His current research interests are in computer science education and scientific computing.

Dr. Jennifer R Amos, University of Illinois at Urbana - Champaign

Dr Amos joined the Bioengineering Department at the University of Illinois in 2009 and is currently a Teaching Associate Professor in Bioengineering and an Adjunct Associate Professor in Educational Psychology. She received her B.S. in Chemical Engineering at Texas Tech and Ph.D. in Chemical Engineering from University of South Carolina. She completed a Fulbright Program at Ecole Centrale de Lille in France to benchmark and help create a new hybrid masters program combining medicine and engineering and also has led multiple curricular initiative in Bioengineering and the College of Engineering on several NSF funded projects.

A case study of early performance prediction and intervention in a computer science course

Abstract

This study presents the results of a course intervention performed in a large upper-division undergraduate computer science class designed to offer additional resources to students that were identified as at-risk of low performance after completing graded assessments during the first two weeks of the semester. The course uses Python as the required programming language, however not every student that takes the class has prior experience with Python. The disparity in programming skills can greatly affect the overall student's experience in the classroom and potentially their overall course performance. We used data from the first two quizzes and one homework assignment from previous semesters to train a model using machine learning algorithms in order to predict students that were at risk of lower performance. At the end of week 2, students identified as at-risk received an invitation to join a 6-week course, which was created to give students an additional opportunity to work on programming activities using Python. The tasks involved real world examples, designed in a structured way to allow students to complete the solution on their own, without a lot of guidance from the instructors. Focus groups were conducted to capture student perceptions of the course.

Introduction

Identification of at-risk students and effective intervention is a critical task in pedagogy. This task is made significantly more difficult in settings with high student-to-teacher ratios. A course with hundreds of students and a single instructor will typically rely on auto-graded assignments and have limited opportunities for deep student-instructor interaction. As a result, identification of at-risk students requires instructors to actively look for and identify students that are struggling. Moreover, it is critical that identification occur as early as possible in the course, thereby giving the intervention a greater chance of success.

The importance of intervention, and the difficulty of successful intervention in high enrollment environments, has prompted a significant amount of research. A number of studies have analyzed data from introductory courses, looking for correlations between prior experience or demographics and performance in the course[1,2,3]. Other studies have looked at data gathered in-class as well as prerequisite course performance[4,5,6,7]. Inspired by these previous work, our study focuses on using early in-course assessment to identify at risk students. We then go beyond identification, and describe and analyze the performance of our intervention method.

Our target learning environment was a large upper-division undergraduate Computer Science course. The course draws students from multiple departments and resultant varied backgrounds. This is especially, and most problematically, true in terms of programming proficiency and experience. The course requires the use of the Python programming language. However, not every student that takes the class has prior experience with Python. Moreover, the overall programming skill level is not uniform among the students, due to the large diversity of majors in the class. This disparity in programming background can greatly affect the overall students' experience in the classroom and potentially their overall course performance. The diversity of student backgrounds made the use of in-course assessment, as opposed to grades in prior courses, an appealing mechanism for identifying at-risk students. We used early in-course assessments of foundational programming and mathematical skills to gather data about the students. Gathering this data, along with student outcomes, over multiple semesters, enabled us to develop a machine learning based system for identification of at-risk students. We also developed an intervention method, a small breakout course using active learning methods, and assessed its efficacy. We believe the work described in this paper can provide evidence and guidance for instructors who wish to identify at risk students early in their class, as well as design and deploy intervention methods.

Related Works

As mentioned, many researchers have already investigated the practicality of predicting student success through various available student features. However, there appear to be far fewer reports on the actual attempt to perform intervention once at-risk students have been identified. We consider our report regarding the structure, efficacy, and perception of our intervention attempt to be the core contribution of this paper.

The most relevant previous work to our efforts have focused on the use of machine learning to identify at-risk students. A study by Quille and Bergin[8] was based on a model taking into account only self reported programming efficacy, mathematical ability based on entrance exams, and hours per week spent playing video games. A Naïve Bayes classifier was created with promising results. Another study by Wolff, Zdrahal, Herrmannova, and Knoth[9] also trained a Naïve Bayes classifier on basic student demographics and click patterns on the course Moodle LMS webpage in an attempt to predict overall performance. Results found that while performance can vary from class to class, click patterns can overall be useful to predict performance when supplemented with other features.

Relevant work has also been done that takes student data from the semester into account as well when seeking predictive results on student performance. Liao et al.[4] trained a logistic regression predictor on grades in prerequisite classes, in-class "clicker" grades, homework grades, and quiz grades and have found that prerequisite grades are a key feature in predicting performance. Both Koprinska et al.[10] and Romero et al.[11] have additionally used features relating to online class messaging boards, with data such as how many messages were read and how many were posted by the individual student.

A meta analysis of the effectiveness of various intervention studies was performed by Szabo, Falkner, Knutas, and Dorodchi[12], where a total of 129 papers in CS Education were evaluated based on the intervention techniques used and their perceived improvement on student

performance. The analysis found that some of the interventions that had the largest benefit on the most number of students included: "automated feedback, active learning, permission to consult outside sources for an assignment, think-pair-share, among others"[12].

Methods

The motivation of this study was based on two research questions:

- RQ1: Can performance from assessments in the beginning of a course corresponding to prerequisite content be used to predict performance at the end of the semester?
- RQ2: Can we improve the student experience in a course by implementing interventions based on the results of these early predictions?

This intervention study took place in a required upper-division undergraduate computer science course from a large public research university during the Fall 2019 semester. Students that take this course are expected to have taken linear algebra and introductory programming courses as prerequisites. All the course assessments are delivered online, including homework assignments, bi-weekly quizzes, and one final exam. The assessments include a mixture of theory, calculation and programming questions.

In Fall 2019, 355 students were registered in the course by the end of week 2, which is the drop deadline for required courses, and was made up of 50% seniors and 45% juniors and a gender distribution of approximately 25% female and 75% male. By the end of the semester, 329 students completed the final exam, giving a retention rate of 93%. The course had a very diverse distribution of majors, with 44% CS, 20% CS + X, 18% Engineering and 18% students from other majors.

The course has been taught by the same instructor since Spring 2018 and hence the assessment types and point distribution have been very similar for the past few semesters. In Fall 2019, students had 18 homework assignments, 7 quizzes and one final exam, which is consistent with previous semesters. This study used data from Fall 2018 and Spring 2019 semesters to train a model using machine learning algorithms for early prediction of students that were at risk of lower performance. The assumptions were:

- **Early predictions**: the predictions were based on data from the first two weeks of the semester, which included one homework and one required quiz covering basic Python programming knowledge and one optional quiz covering linear algebra concepts. Both topics are considered foundational tools for the course, and lack of knowledge in any of them can be very detrimental to student success. By assessing these skills early in the semester, students get feedback about their level of preparedness, and therefore have the opportunity to set up some time to do extra review, or even drop the course and first get prepared to take the course at a later time.
- **Prediction Model**: After exploring different machine learning algorithms, we trained our model using a Gradient Boosting classifier to perform the binary classification ("at-risk" vs "not at-risk").

- **Students at-risk of lower performance**: to train the model, we considered students with final grade less than 80%, which roughly represents 30% of the class, to be an indicative of lower performance. Students targeted as at-risk may need more studying resources and help in order to achieve a higher grade.

Early predictions using a trained model

This computer science course uses computerized homework assignments and quizzes. For the homework, students have one week to complete the assignment to obtain full credit. The bi-weekly quizzes are administered in a proctored computer-based testing facility, where students are free to reserve their 50-minute slot at any time over a period of 4 days. For each assessment, the instructor has access to the following information: (a) the day the student first opens the homework, or schedules the quiz, (b) the time duration to complete the assessment, and (c) the score. Since one homework and two quizzes were used in this study, a total of 9 features were available to train the prediction model.

We first performed feature extraction using an Extra-Trees Classifier to identify the most important features for the prediction model. Figure 1 shows all 9 features, and their corresponding scores in terms of feature importance. `Day_count` corresponds to the day the student started the assessment. For example, a student that takes the quiz on the first day has a `day_count` equal to zero whereas a student that takes the quiz on the last day has a `day_count` of three. `duration` is the time that a student takes to complete the assessment and `score` is the final grade in the assessment.

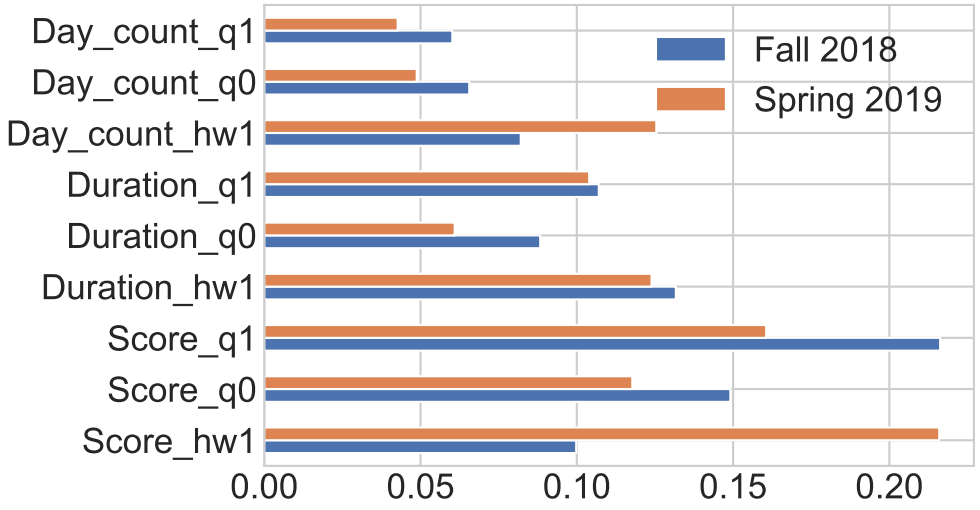


Figure 1: Feature importance for Fall 2018 and Spring 2019. The assessment scores have on average higher importance.

Based on the results from the feature importance analysis, we decided to train the model using the `duration` and `score` for all three assessments, and the `day_count` for the homework.

We used a Gradient Boosting Classifier to perform the binary classification (at-risk vs. not at-risk). The complete data set from Fall 2018 and Spring 2019 consisted of 737 students, and the

model was trained using a set of 471 students. The output of the model is a number between 0 and 1 representing the probability that a student is at-risk. The score metrics of the model depend on the choice of the probability threshold that defines if a student is at-risk or not. The Received Operating Characteristic (ROC) curve illustrated in Fig.2 shows two metrics when using the trained model to predict performance of the students in the test data set. The x-axis has the False Positive Rate (FPR) which gives the probability of an incorrect prediction of students that are actually not at-risk. The y-axis has the True Positive Rate (TPR), which gives the probability of a correct prediction of the students that are actually at-risk. Each point in the graph represents the values of FPR and TPR for varying values of threshold. For this study, after looking at different metrics, we selected a threshold of 0.4.

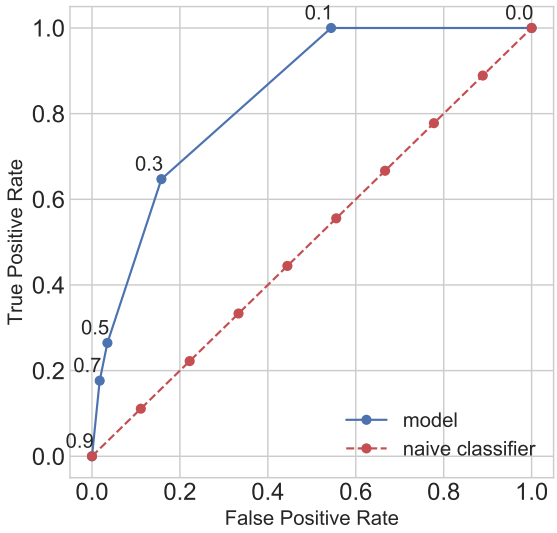


Figure 2: ROC plot

Course intervention

At the end of week 2 in Fall 2019, we used the same features and the trained model to predict the students at-risk. Students that were identified as at-risk received an invitation to join a 1 credit-hour discussion section, created specifically for this intervention study, and therefore closed to students outside of the course. Students were recruited by a faculty researcher who was not involved in the course to consent to participate in the study. Out of the 75 students that were identified as at-risk, only 11 joined the discussion section that started on week 3. Students that were not predicted as at-risk were also aware of this additional discussion section, and 13 of them requested to participate. Therefore, we had the students split into four distinct groups:

- Group A: 11 students who were predicted as at-risk and joined the intervention section
- Group B: 64 students who were predicted as at-risk and did not join the intervention section
- Group C: 13 students not predicted as at-risk and joined the intervention section
- Group D: 267 students not predicted as at-risk and did not join the intervention section

The class met once a week for 80 minutes, and was held in an active learning classroom, where each group table had a large computer monitor and a white board. Students were split into groups of 5 and given a programming problem based on real world examples that required the entire group to collaborate. These programming assignments were presented in the form of a "notebook", containing text, executable Python code, and prompts for students to complete the rest of the code. The tasks were designed to allow students to read the material and run examples at their own pace without much guidance from the instructors (Figure 3).

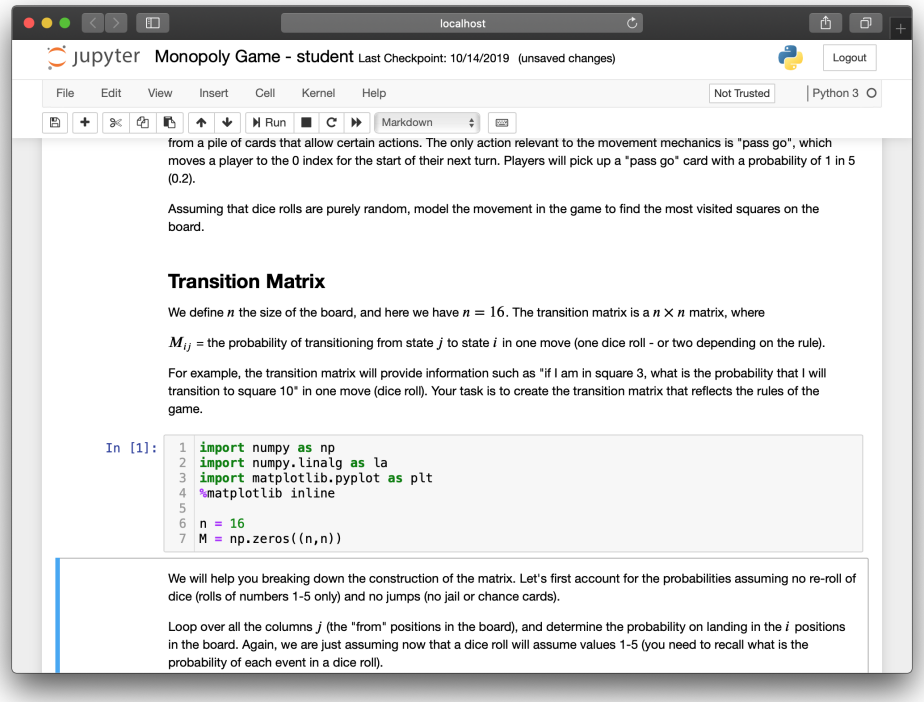


Figure 3: Example screenshot of one of the activities given to the class. Students are given a mixture of background explanation, code snippets, and blank cells to fill in.

While the activities themselves were not completed for a grade, students were encouraged to complete the notebooks to the best of their abilities and get through as much of the activity as they could. Instead, in an effort to foster a low-stress and supportive learning environment, students were graded on their attendance and participation, with full points being awarded if students gave reasonable efforts during the in-class activities.

Student demographics

The students involved in this study had a very diverse background, including 44% CS, 20% CS + X, 18% Engineering and 18% of students from other majors. The gender distribution was 25% female and 75% male. Since we were interested in recruiting as many students as possible for the intervention, and we had a constraint that we could only offer the class once a week, we decided to not enforce a balanced distribution of gender and majors in the intervention discussion section. The demographics distribution per group is shown in Table 1, which excludes the students that

dropped the course. Results from a Chi-Squared Test with a significance level of 0.05 indicate that the observed major frequencies from the four groups do not match the expected frequencies (p-value of 0.028). On the other hand, a similar test indicates that observed gender frequencies from the four groups match the expected frequencies (p-value of 0.34).

	A	B	C	D
CS	56%	41%	8%	47%
CS+X	11%	23%	8%	20%
Eng	0	19%	38%	17%
Other	33%	17%	46%	16%
Male	89%	72%	92%	75%
Female	11%	28%	8%	25%

Table 1: Student major and gender distribution in each one of the groups

Note that the larger groups B and D, corresponding to the students that did not join the intervention, have a demographic distribution similar to the overall class. However, the smaller groups A and C, corresponding to the students that participated in the intervention study, do not have a demographic distribution akin to the overall course distribution. Chi-Squared Tests confirmed that the major distribution of groups B and D match the expected distribution (p-values = 0.93), but groups A and C do not (p-value = 0.042).

Results

At the end of the semester, we were able to identify the group of students with low performance, or the group that was confirmed as at-risk, in this study characterized by the final grade less than 80%. We selected four score metrics to help us answer the first research question: can we use assessment data from the first two weeks of the course to predict a student's performance at the end of the semester? We looked at `accuracy` (percent of overall correct predictions), `recall` (percent of the students confirmed at-risk that were correctly predicted as at-risk), `specificity` (percent of the students confirmed not at-risk that were correctly predicted as not at-risk) and `precision` (percent of predictions at-risk that were correct). The values for each of these metrics is presented in Table 2.

We compared the metrics resulting from the predictions of our trained model with other possible predictions using only the scores from quizzes and homeworks. For example, we investigated two

Prediction method	Accuracy	Recall	Specificity	Precision
ML model	82%	57%	88%	56%
quiz 1 score	74%	57%	79%	41%
quiz 1 and HW 1 scores	75%	50%	81%	42%

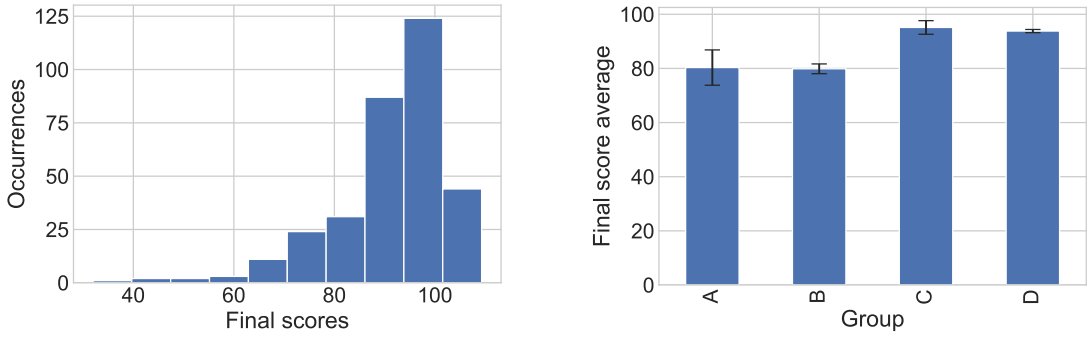
Table 2: Score metrics for the machine learning model used for the predictions in this study and additional score metrics for other possible prediction models, including assessment scores only as thresholds.

other predictors: (a) students with quiz 1 scores below 80% were predicted at-risk, and (b) students with both quiz 1 and homework 1 scores below 80% were predicted at-risk. The corresponding metrics are depicted in Table 2. Note that our trained model performs better than the other two methods.

For this study, the `recall` parameter is the most useful since we want to identify the students at risk that could benefit from the course intervention. Our model gives a 57% recall, which is unfortunately not high, but is better than selecting students at random. The instructor plans to continue gathering data to improve the classification model.

To answer the second research question (can we improve the student experience and performance in the course via an intervention based on early predictions?), we analyzed assessment scores from students in each of the four groups A, B, C and D.

Student final grades



(a) Distribution of final scores

(b) Final score averages

Figure 4: Final score distribution for the entire class, and the corresponding average score for each one of the groups.

Figure 4(a) illustrates the distribution of the course final scores. The overall class average is 91.2% with a standard deviation of 11.7. Some students achieved more than 100% in the course, due to extra credit opportunities offered during the semester. Further investigation of the data indicates that the final score distribution is not normal (Shapiro test with p-value <0.0001), but it deviates from normal in a structured way, consistent with exam distributions reported in Chen et al.[13] and Lord[14]. However, when looking at the final score distribution of individual groups, we noticed that groups A and D have normal distribution (p-value = 0.004 and p-value < 0.0001 respectively) and groups B and C do not have normal distribution (p-value = 0.022 and p-value = 0.46 respectively). To maintain simplicity of the analysis, we utilize statistical tests that assume normality of the data.

Figure 5 shows the final score averages grouped by major and gender. We performed one-way ANOVA tests that confirmed the mean scores between the major groups are not statistically different, with p-value = 0.74. The same was obtained for the gender groups, with p-value of 0.27. Figure 4(b) illustrates the average final scores for groups A-D. One-way ANOVA test indicates that the mean scores are statistically different, with p-value < 0.001. If we combine

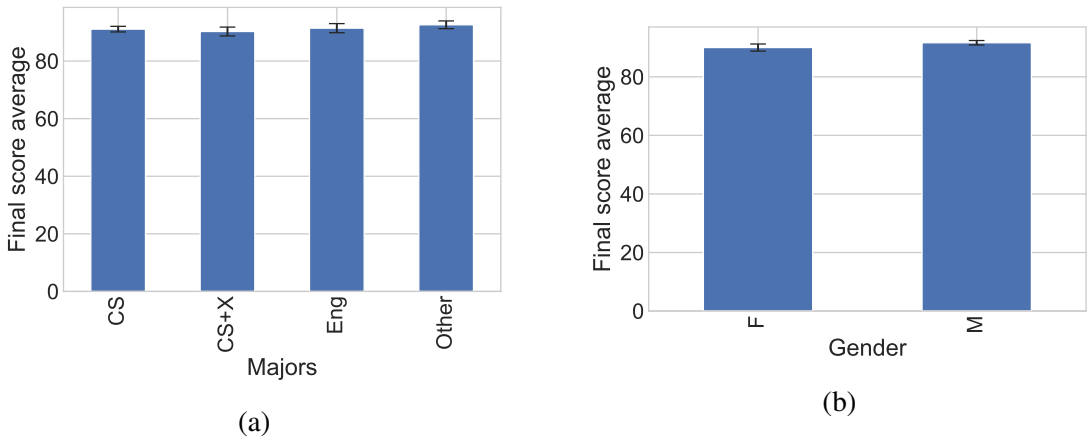


Figure 5: Final score averages with the class grouped by (a) majors and (b) gender

groups A and B (predicted at-risk) and C and D (predicted not at-risk), we obtain mean scores of 94% and 80% respectively, and one-way ANOVA test shows that these two groups are statistically different as well, with p-value < 0.0001.

Since we want to understand the impact of the intervention in student performance, we want to see if there is a difference in the mean values for groups A and B. One-way ANOVA indicates that the mean scores for these two groups are not statistically different (p value is 0.929). Similarly, the same test indicates that the mean scores for groups C and D are are not statistically different (p value is 0.617). Based on these results, we cannot conclude that the intervention helped students to perform better in the course.

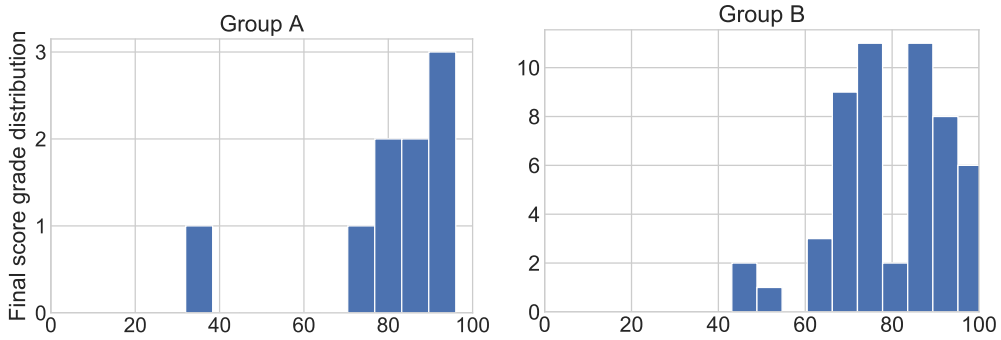


Figure 6: Histogram with distribution of final course scores for groups A and B, consisting of students that were predicted as at-risk.

Figure 6 shows the distribution of final scores for groups A and B, consisting of students that were predicted at-risk. These plots do not include the final scores from the students that dropped the course. Even tough the mean score for these two groups is not statistically different, we can see that the score distributions are different.

Instead of looking at the average scores, we counted the number of students that scored over 80% in each group, as indicated in Table 3. From the group of students that were predicted at-risk, 7 out of 11 students in the intervention group A scored over 80% and 26 out of 64 students from

	A	B	C	D
Total # students	11	64	13	267
Scored over 80%	7	26	12	236

Table 3: Number of students that had a final score over 80%

group B scored over 80%. The frequency of students that were able to achieve the 80% threshold in group A was higher than expected but not statistically significant, based on a permutation test between groups A and B with a p-value of 0.11. From the group of students that were not predicted at-risk, 12 out of 13 students in the intervention group C scored over 80% and 236 out of 267 students from group D scored over 80%. These results were statistically significant with p-value < 0.0001 based on a permutation test between groups C and D.

The lack of statistic significance can be attributed to the small size of the intervention group A. In summary, our quantitative results are not sufficient to conclude the intervention had positive impact in student learning outcomes. In the next section, we will analyze the results obtained from surveys and focus groups.

Student feedback

A faculty researcher from a different department conducted focus groups with students in the class to capture their perceptions of the course. We also asked the students the reasons why they decided (or not) to join the discussion section. Focus groups consisted of 5 or 6 participants each and lasted about twenty-five minutes. During each focus group session, participants were encouraged to share their responses to the following questions:

1. Describe your study habits in CS 357. What were your favorite (most beneficial) course resources that helped you learn the course content?

2. Describe your programming experience before joining the course. Were you familiar with Python?

3. In your first HW, you were encouraged to complete a Python online tutorial. Did you take the online course? If yes, was it helpful?

4. Describe your experience completing the MPs. Did you find the content interesting? Was it difficult? Do you feel you had some disadvantage when completing the MPs, due to a lack/deficiency of Python background?

For groups A and C who joined the intervention,

5. Do you feel the Jupyter notebook activities in CS 199 helped you getting more comfortable writing code in Python?

6. If you were not required to complete the CS 199 Jupyter notebooks (if you were not registered in the class), do you think you would have completed the activities on your own?

7. Do you think working with a group to solve the Jupyter notebook activities was helpful? Would you have preferred to work on the activities by yourself?

For students in groups B and D who did not join the intervention,

5. Did you use the Jupyter notebook activities that were offered as an extra resource for students in CS 357? (the same as used for students in CS 199).

6. What were the reasons for not joining the class?

Survey results for students who enrolled in the course captured reasons ranging from lack of confidence in math or programming ability to just wanting to have more practice. For students who were invited but chose not to enroll, reasons stated all centered around scheduling issues or conflicts with other responsibilities.

Focus group transcripts were analyzed for themes distinguishing between invited students who enrolled versus those who did not enroll. There were 19 participants for the invited but not enrolled group (Group B) and 6 participants for the invited and enrolled group (Group A).

Both focus groups cited similar study habits with redoing homework as the number one study habit. Students also mentioned listening to the recorded lectures and reviewing notes as a good study resource.

Students who were invited and joined the group expressed a lack of confidence in programming ability in Python compared to the group who did not join the class. Many students communicated that this self-perceived lack of ability or experience was what led them to enroll.

Students who enrolled in the intervention course shared that the programming exercises in the extra course added to their understanding of the material and that the group environment was helpful to their understanding. In particular, students expressed that if they were confused in class, they didn't know who to ask or didn't feel comfortable asking questions; whereas in the intervention class, they were sitting in a group working together, so it facilitated asking for help as needed. In particular, two students felt that the team environment was what kept them from dropping the class.

Interestingly, some of the students who did not enroll in the class acquired the intervention Jupyter notebooks and used the exercises as review materials. These students said that they wanted to see what the class offered and admitted that the materials were helpful for their overall success in the course.

Overall, the materials developed for the intervention were perceived as helpful for both groups of focus group students. Students who enrolled in the course felt that the team aspect of the intervention was key to the success.

Discussion

In this paper we described a method for early identification of students at-risk, and an intervention strategy to help those students. Our results indicate that early assessments in the course that are based on prerequisite concepts can be good resources for the identification stage.

The qualitative results from the intervention indicated that students benefited from the additional breakout course, but the results were not significant. Limitations for the qualitative aspects of the study include the sample size and recruiting process allowing for self-selection. To address both

of these limitations, protocols for the study will be modified to allow the faculty researchers to enter the classroom and conduct focus groups with students. This will ensure that study results are representative of the larger population.

In addition, participants who were not invited to the intervention class were not included in the focus groups. After learning that students who were not enrolled had access to the course materials, it would be helpful to know if the rest of the class also had access to or used the intervention materials. Without this information, it is unclear whether the effect size is attributable to the combined intervention of Jupyter notebooks combined with teamwork or if the materials themselves were enough to enhance performance.

We believe that new and more experienced instructors can use this approach to identify at-risk students early in their class and use these methods to measure the success of an intervention.

References

[1] 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 2016 ACM Conference on International Computing Education Research*, ser. ICER '16. New York, NY, USA: Association for Computing Machinery, 2016, p. 211–220. [Online]. Available: https://doi.org/10.1145/2960310.2960329

[2] B. C. Wilson and S. Shrock, "Contributing to success in an introductory computer science course: A study of twelve factors," *SIGCSE Bull.*, vol. 33, no. 1, p. 184–188, Feb. 2001. [Online]. Available: https://doi.org/10.1145/366413.364581

[3] 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 ACM Technical Symposium on Computer Science Education*, ser. SIGCSE '18. New York, NY, USA: Association for Computing Machinery, 2018, p. 687–692. [Online]. Available: https://doi.org/10.1145/3159450.3159452

[4] S. N. Liao, D. Zingaro, C. Alvarado, W. G. Griswold, and L. Porter, "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*. ACM Press, 2019. [Online]. Available: https://doi.org/10.1145/3287324.3287407

[5] 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 International Conference on International Computing Education Research*, ser. ICER '15. New York, NY, USA: Association for Computing Machinery, 2015, p. 121–130. [Online]. Available: https://doi.org/10.1145/2787622.2787717

[6] 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 2017 ACM SIGCSE Technical Symposium on Computer Science Education*, ser. SIGCSE '17. New York, NY, USA: Association for Computing Machinery, 2017, p. 111–116. [Online]. Available: https://doi.org/10.1145/3017680.3017792

[7] V. Ramalingam, D. LaBelle, and S. Wiedenbeck, "Self-efficacy and mental models in learning to program," *SIGCSE Bull.*, vol. 36, no. 3, p. 171–175, Jun. 2004. [Online]. Available: https://doi.org/10.1145/1026487.1008042

[8] K. Quille and S. Bergin, "Programming: predicting student success early in CS1. a re-validation and replication study," in *Proceedings of the 23rd Annual ACM Conference on Innovation and Technology in Computer Science Education - ITiCSE 2018*. ACM Press, 2018. [Online]. Available: https://doi.org/10.1145/3197091.3197101

[9] A. Wolff, Z. Zdrahal, D. Herrmannova, and P. Knoth, "Predicting student performance from combined data sources," in *Educational Data Mining*. Springer International Publishing, Nov. 2013, pp. 175–202. [Online]. Available: https://doi.org/10.1007/978-3-319-02738-8_7

[10] I. Koprinska, J. Stretton, and K. Yacef, "Predicting student performance from multiple data sources," in *Lecture Notes in Computer Science*. Springer International Publishing, 2015, pp. 678–681. [Online]. Available: https://doi.org/10.1007/978-3-319-19773-9_90

[11] C. Romero, S. Ventura, P. Espejo, and C. Martinez, "Data mining algorithms to classify students." 01 2008, pp. 8–17.

[12] C. Szabo, N. Falkner, A. Knutas, and M. Dorodchi, "Understanding the effects of lecturer intervention on computer science student behaviour," in *Proceedings of the 2017 ITiCSE Conference on Working Group Reports - ITiCSE-WGR'17*. ACM Press, 2017. [Online]. Available: https://doi.org/10.1145/3174781.3174787

[13] B. Chen, M. West, and C. Zilles, "Do performance trends suggest wide-spread collaborative cheating on asynchronous exams?" in *Learning at Scale*, 2017.

[14] F. M. Lord, "A survey of observed test-score distributions with respect to skewness and kurtosis," *Educational and Psychological Measurement*, vol. 15, no. 4, pp. 383–389, 1955.