Modeling Student Success

by

Jacob Strasler

A Report Submitted to the Faculty of the Milwaukee School of Engineering
in Partial Fulfillment of the Requirements for the Degree of
Master of Science in Machine Learning

Milwaukee, Wisconsin

August 2024

Abstract

Demand for graduates from computing disciplines has been steadily increasing for decades, with corresponding growth in salaries. However, the persistence to graduation for students starting in computing majors remains lower than in many other fields. Despite the high academic capability of many students, retention rates in computer science programs are alarmingly low, with significant numbers of students dropping out or switching majors, often within their first year. Early academic struggles, particularly in the first-year programming sequence, are key indicators of future academic success and retention. While various interventions have been implemented to improve retention and performance, their effectiveness has been mixed, highlighting the need for a more comprehensive understanding of the factors influencing student success.

This study aims to address these challenges by identifying early risk factors and exploring the potential of machine learning techniques to predict retention and penalty grades in computing majors. By taking a holistic view of each student, the research demonstrates that early identification of at-risk students is possible, enabling targeted and resource-efficient interventions to support their success.

1 Introduction

The demand for graduates with computer programming skills is substantial, yet a significant number of students who initially pursue this path drop out or change majors within their first year. Despite being academically capable, many of these students encounter roadblocks that cause them to choose another path. The rate of students starting but not finishing a computing major highlights the pressing need for a deeper understanding of student progress, making it a vital focus of ongoing research in computing education.

This phenomenon is evident even during the first year, with only two-thirds of students worldwide successfully completing their introductory CS1 course [1]. Increasing graduation rates begins with improving retention. Retention, defined as the percentage of students who return to the same institution for their second year, serves as an indicator of a program's alignment with student interests and demonstrates the robustness of its admission criteria. Additionally, most students who successfully complete the first year programming sequence are likely to finish their degree.

In addition to retention, insights to student progress and likelihood of being retained throughout the first-year programming sequence are crucial. Many students who pass the first semester course struggle in subsequent courses, leading to lower likelihoods of completing their computer science degree. Students who successfully complete the initial part of the first-year programming sequence but later earn a penalty grade—defined by the Milwaukee School of Engineering (MSOE) as a D, F, or W—highlight a potential misalignment between early academic performance and the skills necessary for future success. This indicates that a strong first-semester grade may not always reflect the depth of knowledge required for succeeding in subsequent courses. The issue is compounded by the fact that the material in the second course of the first-year programming sequence is typically more challenging and unfamiliar, as these concepts are often not covered in K-12 curriculum. A drop in academic performance from CS1 to CS2 clearly signals that early academic struggles are a direct indicator of retention, underscoring the need for a more comprehensive approach to understanding and addressing the factors that influence student success and persistence.

Confidence and prior experiences are significant predictors of student performance in first-year courses. Alvarado et al. [2] found that both factors significantly predict performance, particularly for male students, indicating a gender disparity. However, curriculum design can mitigate the advantages of prior experience. For example, an objects-first approach in teaching may lessen the impact of prior programming experience. Harrington et al. [3] observed that students with prior programming experience tend to overestimate their grades, highlighting the complexity of how confidence and experience shape student outcomes. These student outcomes, in particular in programming and math courses, strongly influence retention.

Interventions aimed at improving retention and performance in first-year courses have yielded mixed results. Golding et al. [4] found that peer tutoring, which is offered at MSOE, positively impacts students' confidence and academic performance, suggesting that peer support can be a valuable resource. Stephenson et al. [5] further indicate value in peer tutoring as well as various paths through the first-year programming sequence, also available to MSOE students. However, Krause-Levy et al. [6] argue that only resource-intensive interventions, such as supplemental instruction, have a positive effect on student outcomes.

Lyon et al. [7] show that pre-semester coding programs can significantly boost confidence and motivation, especially for students with no prior programming experience. These interventions add

value by both increasing students' understanding of programming concepts and building upon the confidence they may or may not have had upon matriculation but, because of their resource overhead, need to be directed to the students most likely to benefit from them.

This research aims to address the challenge of predicting retention in computing majors by identifying underlying and early risk factors, and exploring the potential of machine learning techniques to predict retention and early risk indicators such as penalty grades. By adopting a holistic view of each student, the study demonstrates that early identification of students at risk of negative outcomes is possible. This early prediction enables the implementation and expansion of targeted, resource-efficient remediation efforts to support these students.

2 Data and Methodology

2.1 Data Context

The data set utilized for the study contained information about incoming computer science, software engineering, and computer engineering students for the 2023-2024 academic year ($n = 224$). Information including math and programming preparedness, academic performance, and university-assessed success ratings was utilized. A summary of these attributes can be seen in Table 1. Data was sourced from various places, including the university's Academics Office, an optional survey sent to incoming students, a data set containing program exam information, and academic data. Of the original 224 students, 218 had sufficient data, meaning they had data for all attributes intended to be used in modeling.

Table 1: Attributes

Attribute	Notation	Description	Source	Used In
Before First Semester				
ALEKSMAX	ALX	Numerical measure of a student's math preparedness using their normalized ALEKS score and ACT/SAT math score and equally weighting them to create a score. Scores range from 0 (worst) to 1.	Computed	Retention
Confidence	CON	Numerical measure of how confident a student is in their existing knowledge of CS1 programming concepts.	Incoming Survey	Retention
After First Semester				
First Semester Programming Course Letter Grade	S1PLG	A discrete representation of a student's letter grade in their first semester programming course.	Programming Exam Data	Penalty Grade; Retention

Continued on next page

Attribute	Notation	Description	Source	Used In
First Semester Programming Course Final Exam Score	S1PFE	A student's numerical score on the final exam in their first semester programming course.	Programming Exam Data	Penalty Grade; Retention
First Semester Math Course Letter Grade	S1MLG	A discrete representation of a student's letter grade in their first semester math course.	Math Data	Penalty Grade; Retention
First Term Grade Point Average	S1GPA	A student's numerical grade point average after their first semester at the university.	Office of Academics	Penalty Grade; Retention
End of First Semester Success Rating	S1SR	A discrete representation of a student's likelihood of success calculated using academic data (ALEKS, S1MLG, S1GPA), engagement metrics (new student survey), and demographic information (Pell eligibility, first-generation status). Scored 0 (best) to 6.	Office of Institutional Effectiveness	Retention
After Second Semester				
Second Semester Programming Course Final Exam Score	S2PFE	A student's numerical score on the final exam in their second semester programming course.	Programming Exam Data	Retention
Second Semester Programming Course Letter Grade	S2PLG	A discrete representation of a student's letter grade in their second semester programming course.	Programming Exam Data	Retention

The path first-year computing students take through the first-year programming sequence is visualized in Figure 1. The first-year programming sequence at MSOE consists of two courses: CSC 1110 (CS1) and CSC 1120 (CS2). Each section has an "A" variant which meets for one additional class period each week and is intended for students who may benefit from an extra meeting based on their incoming programming and math experience. While the "A" variant covers the same learning outcomes, it provides more hands-on time for less experienced students.

The majority of incoming students are first-time students, with just 19 of the 218 incoming students being transfer students who did not start their studies but are in their first term at MSOE. Of this 218, 31 have sufficient programming experience (typically those with transfer or Advanced Placement credit) to take CS2 in the first semester. The remainder of the students enrolled in CS1 or its "A" section.

Students identified as potentially benefiting from an extra meeting are placed into the "A" section of CS1 (CS1-A) based on data collected from the incoming student survey. A student can

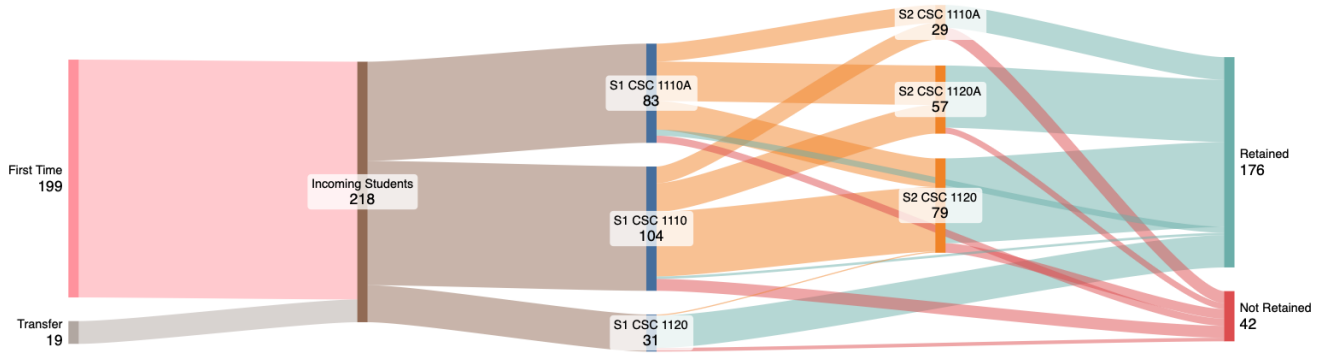

Figure 1: Sankey diagram of student paths through the first-year programming sequence.

opt to take the "A" section of CS2 (CS2-A) in the second semester, and CS1-A section enrollment is not a prerequisite for CS2-A, nor do students who took CS1-A need to take CS2-A.

Student paths do become varied once within the first-year programming sequence, including some students who do not enroll in a second semester course but do enroll for their third semester, but the majority follow the typical CS1 to CS2 path. Most students are also enrolled in a math course for both semesters of the first year. Students' first semester math courses are shown in Figure 2. It should be noted that the sample does not include any students whose first semester was in the spring semester and students who do not enroll in a second semester class are still included as some still enroll for the fall of their second year.

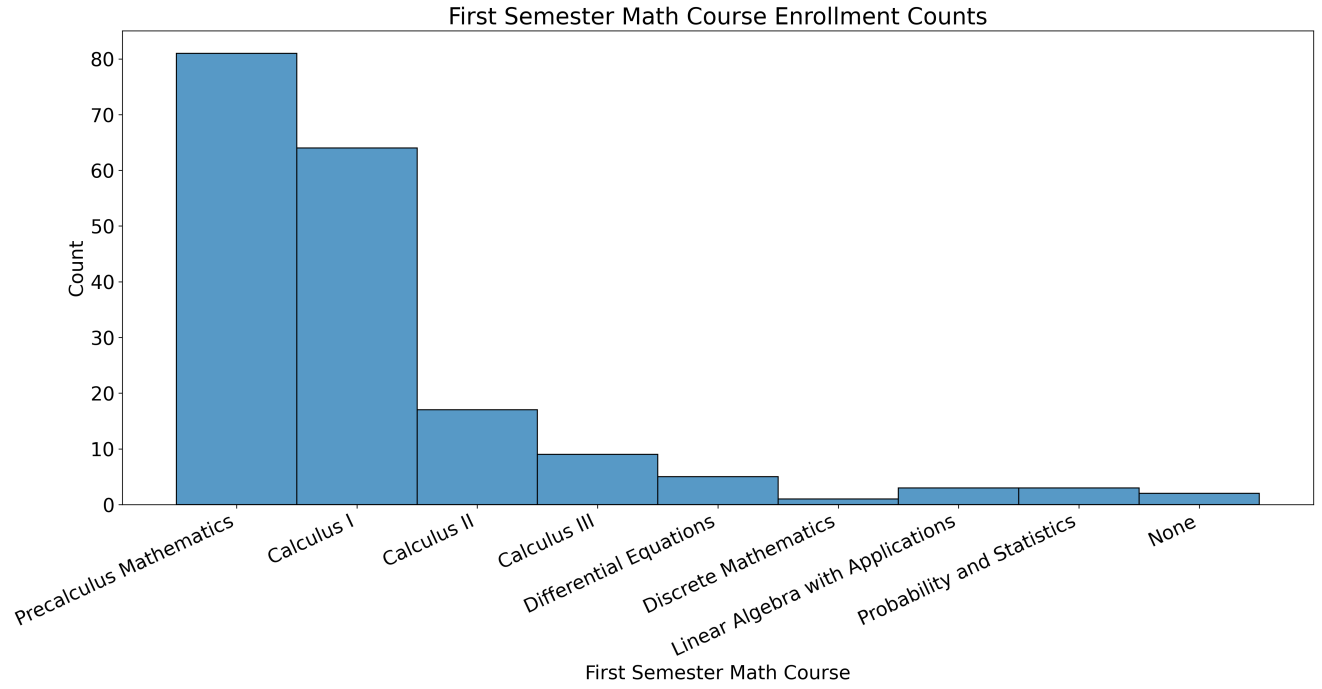

Figure 2: First semester math course enrollment by count.

2.2 Methodology

2.2.1 Data Preparation

A holdout set was created for independent evaluation of all models presented in this work. For both predictive tasks, a different, random 30% selection of the data was used as the holdout set. A relatively high holdout percentage was used to ensure a sufficient sample size for evaluation. These sets were used to evaluate the models' performance. Standard scaling and one-hot encoding were applied to numeric and categorical attributes, respectively. Both problems addressed exhibit imbalance: most students do not receive a penalty grade in the second semester, and most persist to the fall. To correct this, Synthetic Minority Oversampling Technique (SMOTE) is used to balance the ratio of penalty grade to non-penalty grade and persisting to non-persisting cases in the training data.

2.2.2 Model Training

A collection of machine learning classifiers was assessed for each predictive task: logistic regression, random forest, support vector machine, k-nearest neighbors, and gradient-boosted decision trees. For each model except the gradient-boosted decision tree, the scikit-learn [8] implementation was used. For the gradient-boosted decision tree, XGBoost [9] was used. The 70% of the data not included in the holdout set was used as training data. This data was split into training and validation sets, with 20% reserved for validation. Five-fold cross-validation was performed, using receiver operating characteristic area under the curve (ROC AUC) as the evaluation metric. The model with the highest mean ROC AUC score across folds was selected and re-trained on the full training set for evaluation against the holdout set.

2.2.3 Evaluation

Model evaluation was conducted using the holdout set. The models were used to predict probabilities of a student earning a penalty grade or persisting to their second year. ROC AUC scores and confusion matrices were used to analyze model performance. Feature importance scores were also examined to understand the significance of different attributes in predicting the outcomes.

3 Predicting Second Semester Penalty Grades

3.1 Significance of Second Semester Penalty Grades

Earning a second semester penalty grade makes it unlikely a student returns for their third semester: in this study, just 60% of students earning a penalty grade in the second semester are enrolled in a computing major at MSOE the following fall. Early detection allows the university to implement targeted remediation efforts, knowing these students are at a higher risk of earning a penalty grade in the second semester. Earning a penalty grade after passing CS1 may indicate that the first semester grade did not reflect adequate knowledge for success in later courses. However, it could also point to a loss of interest, external influences, or a myriad of other factors. Finally, it's important to note that while a D is considered a penalty grade at MSOE, it is still passing and does not prevent a student from advancing to the next course.

3.2 Description of Second Semester Penalty Grade Data

Students to be evaluated for second semester penalty grade risk are those who successfully completed the first course in the first-year programming sequence and are enrolled in the second course of the programming sequence ($n = 127$). Of these students, 20% earned a penalty grade in the second semester. The paths of the students in the sample through the first-year programming sequence are shown in Figure 3.

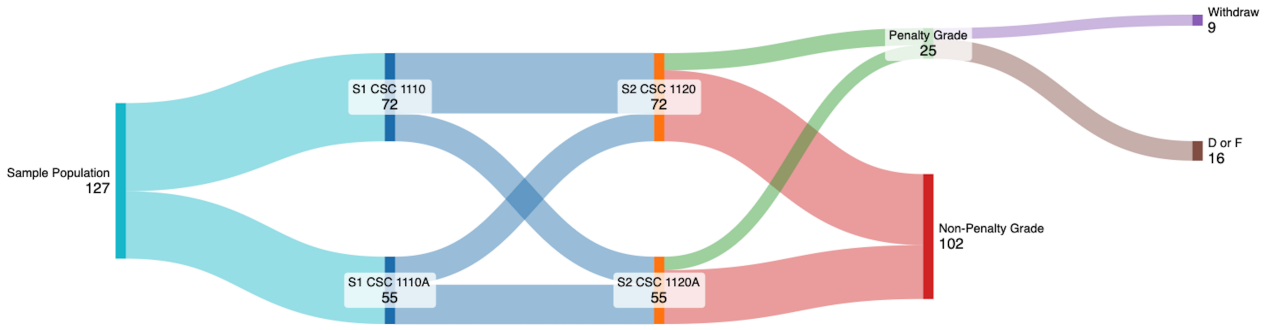

Figure 3: Sankey diagram of student paths through the first-year programming sequence for students who successfully completed CS1 or CS1-A in the first semester.

Students successfully completing the first semester course had a median first semester grade point average of 3.47, approximately equivalent to an "AB" letter grade average. The median final exam score in the first semester was 78.5%, a "C" letter grade. As seen in Figure 4, the distribution of both first semester programming and math course grades are positively skewed, with an "A" being the most common grade in each course. Students who were in the A-section course of the first semester make up 43% of the sample. The majority of students were enrolled in Precalculus Mathematics (28%) or Calculus I (47%), with the remaining students enrolling in more advanced math courses.

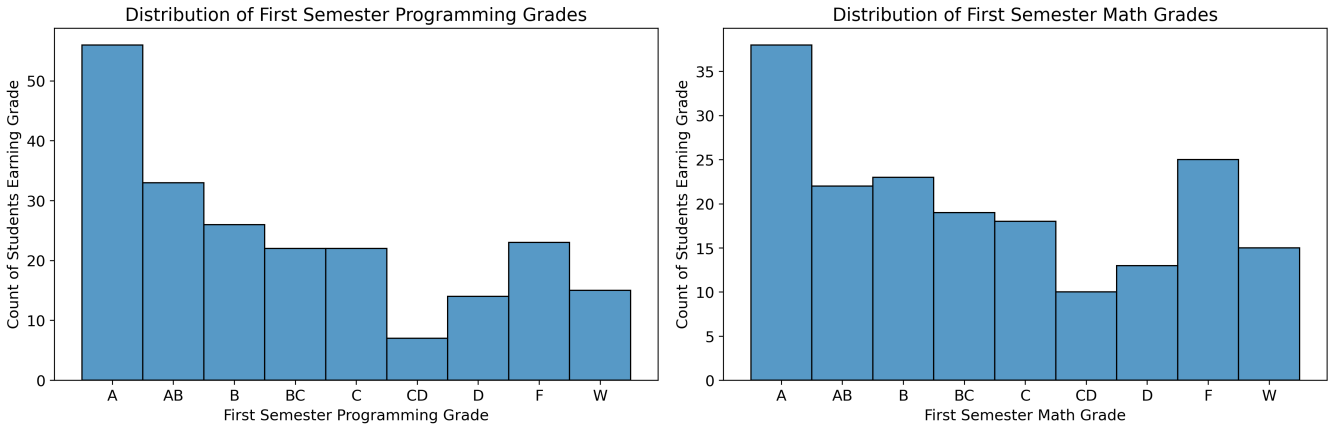

Figure 4: Count of student grades in first semester programming (left) and math (right) courses.

3.3 Second Semester Penalty Grade Modeling Technique

3.3.1 Second Semester Penalty Grade Data Preparation

A holdout for this data set was created for independent evaluation of the models. This set consisted of a random 30% selection of the data ($n = 38$), of which 24% had a penalty grade in the second semester. This left 89 students to be used in model training, with 18% being true cases of earning a penalty grade. The data reserved for training is divided into training and validation sets, with 80% allocated to training and 20% to validation. The models assessed are logistic regression, random forest, support vector machine, k-nearest neighbors, and gradient-boosted decision trees.

3.3.2 Second Semester Penalty Grade Model Training

Model training was conducted as outlined in section 2.2.2, treating this problem as binary classification. The attributes used in the modeling are those available at the end of the first semester: S1PLG, S1PFE, S1MLG, and S1GPA. Other evaluated attributes, such as ALX, CON, the difference between S1PLG and S1PDF, and the initial success rating, did not significantly impact model performance. Among the models tested, the random forest classifier performed best, achieving a mean ROC AUC score of 92% across test sets created using five-fold cross-validation.

3.4 Second Semester Penalty Grade Results

Evaluation was performed using the aforementioned holdout set as was described in section 2.2.3. Analysis of the random forest model indicates that it is effective at identifying students likely to earn a penalty grade in the coming semester. As seen in Figure 5, the ROC AUC score of 88% on the evaluation set indicates that the model does well at distinguishing between positive and negative classes.

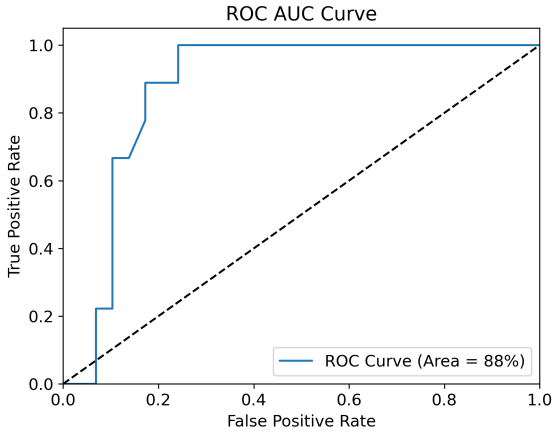

Figure 5: ROC AUC when predicting on the evaluation set whether students earn a penalty grade in the second semester programming course using a random forest classifier.

Further analysis by means of a confusion matrix on the evaluation set, seen in Figure 6, shows that the model is capturing many of the to-be penalty grade earners. In fact, recall is 63%. This does come at the expense of precision, which is just 58%, but this is not necessarily a bad thing:

it is preferable to tolerate false positives to capture more students at risk of a penalty grade at the expense of some remediation costs being used on students not strictly needing it versus students not being identified as at-risk and consequently not receiving the attention necessary.

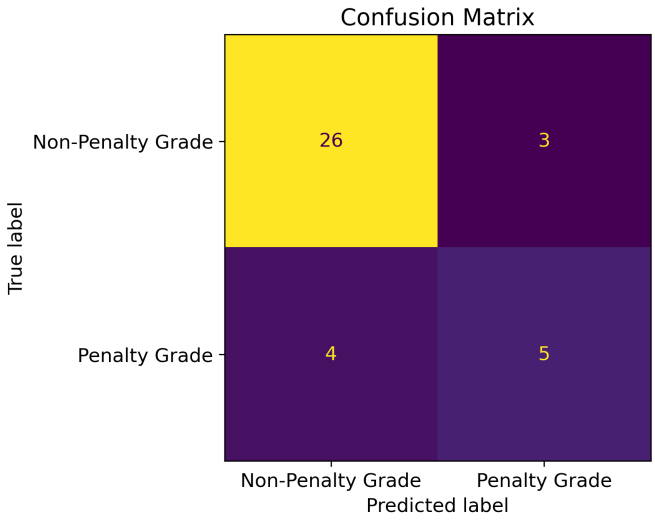

Figure 6: Confusion matrix of when predicting on the evaluation set whether students earn a penalty grade in the second semester programming course using a random forest classifier.

Feature importance for a random forest classifier is calculated by measuring the average decrease in impurity across all trees in the forest when the feature in question is used to split the data. Feature importance scores indicate that S1GPA, S1PLG (in particular whether or not it was a C), and S1PFE are significant predictors of penalty grade outcomes.

An error analysis suggests that the model struggled with students whose S1GPA would not be a cause of concern: the students who had relatively low grade point averages but did not receive a penalty grade and those who had relatively high grade point averages but did receive a penalty grade were the students misclassified by the model. This suggests a potential avenue for future work while also highlighting the nuance involved in student behavior, in particular that penalty grades can be products of external factors a model may never be able to capture.

4 Predicting Second Year Retention

While retention is often looked at from an enrollment-to-graduation point of view, one-year retention rate is "potentially a more meaningful metric for evaluating the impact of CS1—before other factors (later classes, instructors, etc.) add additional noise," [10]. The prediction of how many and which students are likely to return for their second year of programming education has two-fold benefits: early detection allows for early intervention, much like when predicting penalty grades, and it allows for more immediate feedback on institutional changes' effects on retention rates.

4.1 Description of Second Year Retention Data

All students for whom there is sufficient data on can be included in retention predictions ($n = 218$). This includes students who have earned penalty grades in either of the courses and students who started in the second course of the first-year programming sequence, the latter typically being students with Advanced Placement or transfer credit. Of the students used in this sample, a majority (80%) of students are retained, meaning imbalance will be necessarily dealt with when modeling to avoid bias and over-fitting.

The attributes used when modeling this problem will be broken into three groups: before first semester, after first semester, and after second semester. The features that were used for each of these models are indicated in Table 1. An attribute is included in a group according to when it is discoverable. Each group is inclusive of the group(s) before it. Other attributes that were evaluated and did not have a significant impact on model performance are what programming course a student is placed into in the first semester, what programming course they enrolled in in the second semester, high school Advanced Placement computer science course and math course information, and initial success rating and programming course exam grade difference semester-over-semester.

4.2 Second Year Retention Modeling Technique

4.2.1 Second Year Retention Data Preparation

A holdout set was created for independent evaluation of the models. This set consisted of a random 30% selection of the data ($n = 65$), of which 82% are retained students. This left 153 students to be used in model training, with 80% being retained.

4.2.2 Second Year Retention Model Training

Model training was performed as described in 2.2.2. To observe the change in predictive power over time, re-training was done using three different feature sets: incoming features, post-first semester features, and post-second semester features. Each re-training cumulatively included the features from the previous stages. A gradient-boosted decision tree was the most performant of the models across all feature sets, with mean ROC AUC scores of 63%, 73%, and 80%, respectively across test sets of the training data created using five-fold cross-validation. Logistic regression, random forest, support vector machine, k-nearest neighbors, and gradient-boosted decision trees are all assessed for this predictive task.

4.3 Second Year Retention Results

Using the same evaluation set for each model ensures a fair and consistent assessment of the efficacy of different feature sets. This approach allows for accurate comparison across models by minimizing variations that could arise from using different data distributions. By standardizing the evaluation set, any differences in model performance can be attributed directly to the models and feature sets themselves, rather than to inconsistencies in the evaluation data.

The models were used to predict the probability of a student returning for their third semester, treated as a binary classification problem. When reviewing the results, it is important to note that

it is preferable to tolerate some false negative predictions (i.e., incorrectly predicting a student is unlikely to return) in order to identify more students who are at risk of not returning.

4.3.1 Incoming Features

Using the only features available before a student starts their first term, ALX and CON, as inputs, the model performs slightly better than random chance. Figure 7 shows the efficacy of the model using the given feature set. The specificity is just 50% on the evaluation set, meaning the model struggles to correctly identify students who are not likely to return. The median ALX and CON values of students who are predicted to return but did not ($n = 6$) are 60% and 12%, respectively. These values are not egregiously poor as the full sample has median ALX and CON scores of 49% and 15%, respectively, making it unsurprising that the model does not capture these students as unlikely to return. Inspection of feature importance shows the model weights CON as twice as important as ALX.

The lackluster predictive powers of incoming features make sense – there's a lot that influences student outcomes, many of which develop over the course of a school year, and incoming features will not capture this. This suggests that a more holistic view of the student is needed. To do so, post-first semester features are added to incoming features.

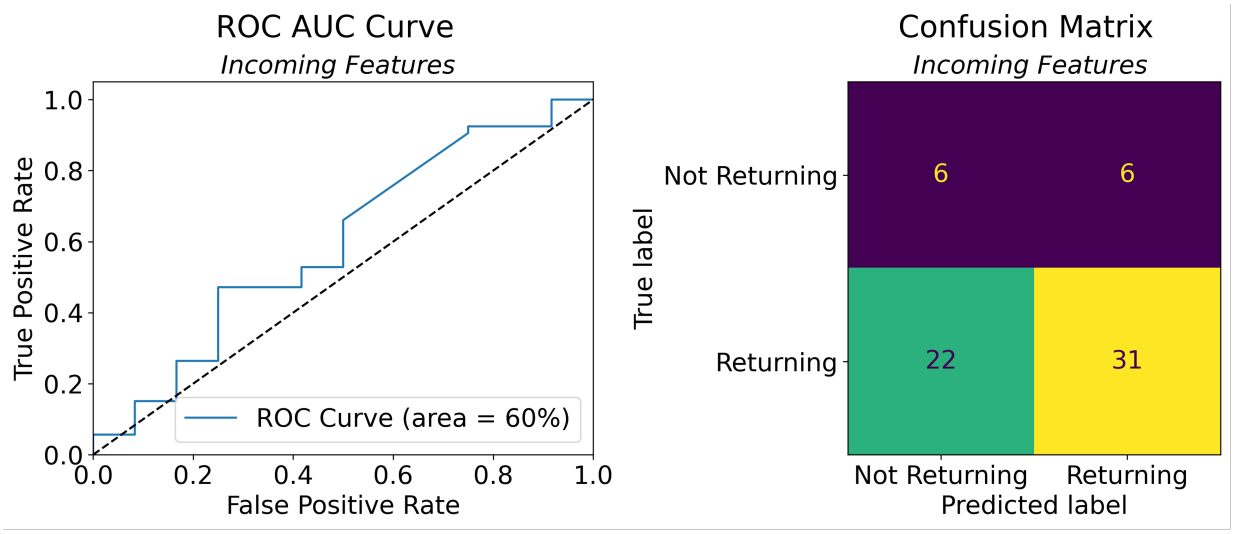

Figure 7: ROC AUC and confusion matrix when predicting on the evaluation set whether students return for their second year using incoming features in a gradient-boosted decision tree classifier.

4.3.2 Post-First Semester Features

First semester features create an opportunity to get a more holistic view of the student while still identifying at-risk students early. Model performance improved with the addition of the post-first semester attributes, as shown in Figure 8. Specificity improved to 58% on the evaluation set and recall improved as the model correctly identifies 17% more of the students who will not return than the incoming features model identified. The differences in predictions between the models

are also statistically significant: a McNemar's test yielded a statistic of 1.0 and a p-value less than 0.01.

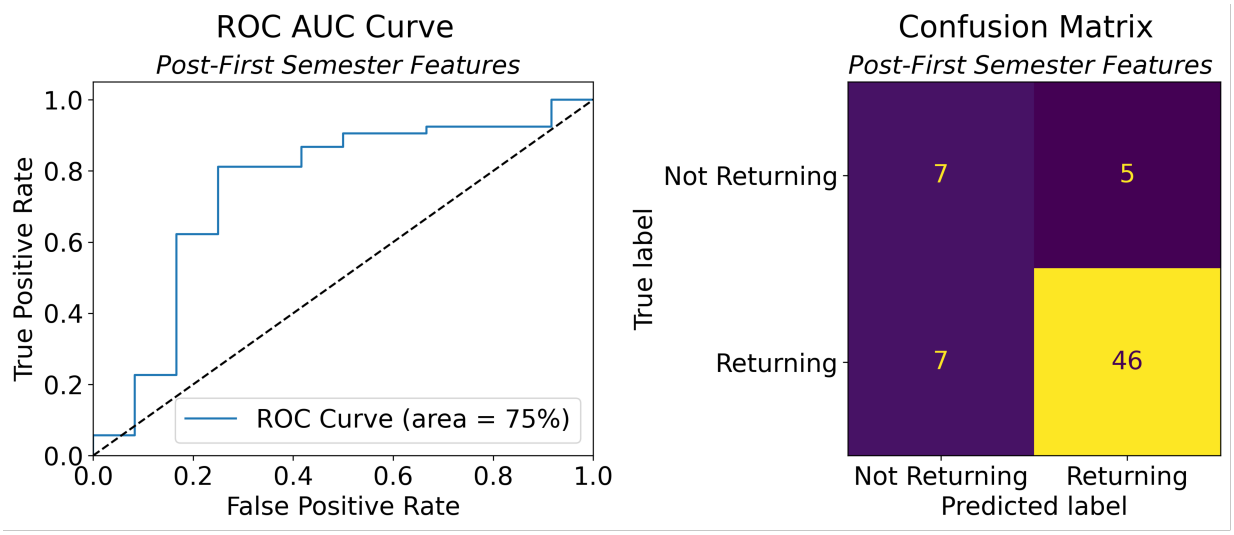

Figure 8: ROC AUC and confusion matrix when predicting on the evaluation set whether students return for their second year using post-first semester features in a gradient-boosted decision tree classifier.

The most effective model for predicting retention with post-first semester attributes was a gradient-boosted decision tree for which feature importance is calculated by measuring the average decrease in impurity each feature contributes across all trees in the model. Feature importance values after first semester attributes are added to the model indicate that having an S1SR rating of zero is the most telling predictor, with W in S1PLG and S1MLG and a student's S1PFE also being highly important. CON is notably not heavily weighted. Of the students predicted to return but do not ($n = 5$), three have an S1GPA of 0.0 and earned a penalty grade in one of or both their programming and math course. The median programming course final exam score is 33%, much worse than the 75% median of the full sample. Two students had a success rating of 0, including one student which finished the first semester with a 3.47 grade point average. This reiterates the challenges external factors play in student outcomes.

4.3.3 Post-Second Semester Features

The final iteration of the model which incorporated post-second semester attributes saw specificity improve to 75% on the evaluation set, and the model correctly identifies 29% more students than the post-first semester features model and 50% more than the incoming features model. Figure 9 shows the improvements made in model performance. The ROC AUC does not increase by a large amount over the post-first semester feature model, and its performance is not statistically significantly improved: a McNemar's test between the post-second semester model and the post-first semester model returns a test statistic of 1.0 with $p = 0.38$. The same test performed between the post-second semester model and the post-first semester model returns a test statistic of 3.0 with $p < 0.01$. The lack of statistically significant improvement with second-semester attributes

suggests the model can be effectively used after the first semester, as later data will not significantly alter predictions. This reinforces confidence in early intervention.

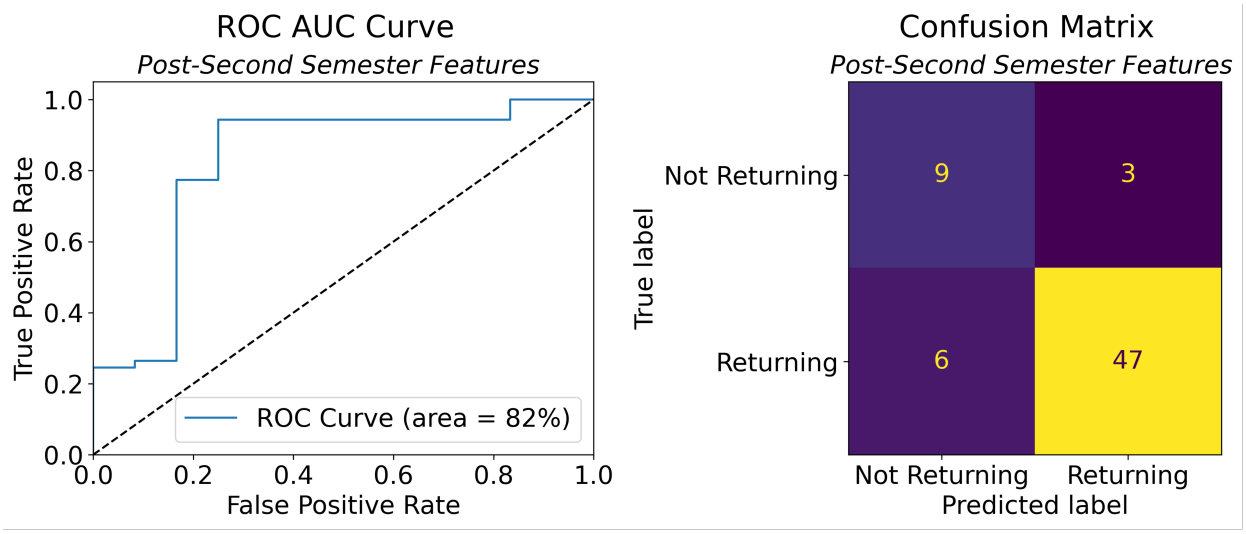

Figure 9: ROC AUC and confusion matrix when predicting on the evaluation set whether students return for their second year using post-first semester features in a gradient-boosted decision tree classifier.

A final error analysis reveals only three students were predicted to return but do not. The model considers the top five features by importance to be an S1PLG of "W", an S1SR of 0, S2PFE, an S1MLG of "W", and CON. The three remaining false positives were misclassified in the same manner by the previous iterations of the model, likely due to their contradictory attributes: these students had low S2PFE's (two zeroes and a 66%), but none dropped the first semester programming nor math course and two of the students had an S1SR of 0 at the end of the first semester. That said, the relatively low rate of false positives is promising.

4.3.4 Second Year Retention Results Summary

As expected, the model performance improves as a more data is available. Results using incoming information are not much better than random chance; however, as features are added to the models after both the first and second semester they become increasingly effective at distinguishing between students who are and are not likely to return. The specificity of the models improves from 50% to 58% when post-first semester attributes are added and finished 75% when post-second semester attributes are added.

5 Future Work

While this study provided significant insights into predicting student success in computing majors, several avenues for future research remain. One important avenue is the inclusion of more years of student data to improve the robustness and generalization of the models. By expanding the data set to cover additional cohorts, the model can capture trends over time, providing a more comprehensive understanding of factors influencing student outcomes.

In addition to having a larger sample size, future research could benefit from integrating additional data sources. While the current study focused primarily on academic and demographic factors, incorporating data such as social engagement metrics and mental well-being indicators could provide a richer and more nuanced picture of the variables affecting student success and retention.

Another crucial area for future work is the simplification of the models used in this study. While random forests and gradient-boosted decision trees have proven effective, they can be complex and difficult to interpret. Future research should explore the development of simpler, more explainable models that prioritize transparency and ease of interpretation. Such models, while potentially less performant, would be more accessible to educators and administrators, enabling them to better understand and act on the predictions. This shift towards simplicity and interpretability would facilitate the practical application of the predictions in an educational setting, ultimately leading to more informed and effective interventions.

Additional interpretability could be achieved through improved feature engineering. This study treats letter grades as categorical features. A more informative approach might involve converting letter grades into their grade point values and using binary indicators instead of discrete values. For example, instead of using a feature like "first semester programming grade of C," further work might consider using "first semester programming grade below C." This method could provide clearer insights and a more interpretable description of feature importance.

Further yet, predicting students' likelihood of retention at additional points in their career could be fruitful. The two most obvious such times are before a student's first semester at the university and the first semester of their second year. Similar to the approach used in this paper for identifying students at risk of under-performance in the second semester, identifying risk in these periods presents another chance for targeted support early in a student's academic career. In particular, the coursework in the third semester is more challenging than that in CS1 and CS2, and, as demonstrated in this study, prior academic success does not necessarily guarantee future success.

In conclusion, expanding the dataset, incorporating diverse data sources, simplifying the predictive models, and focusing on another early-term retention prediction represent key opportunities for advancing research in student success prediction. These efforts will enhance the ability to support students through data-driven insights, contributing to higher retention rates and improved educational outcomes in computing.

6 Conclusion

This study highlights the potential of machine learning models in predicting key academic outcomes, specifically retention and penalty grades among first-year computer science students. The findings reveal that early indicators, such as first-semester performance metrics, play a crucial role in identifying students at risk of earning penalty grades or not returning for their second year. By leveraging these predictive models, educational institutions can implement timely, targeted interventions that effectively support student success, highlighting the practical application of machine learning in educational settings.

The random forest classifier, with its strong performance in predicting penalty grades, highlighted the significance of first-semester GPA, programming course grades, and final exam scores

as predictors. This suggests that academic performance in the first semester is not only reflective of current understanding but also indicative of future challenges students may face. Furthermore, the ability of the gradient-boosted decision tree to predict retention rates allows for early intervention, which can mitigate the loss of students due to academic difficulties or misalignment with the program. This proactive approach enables educators to address potential issues before they lead to attrition, thus improving overall retention rates and student outcomes.

However, this study also acknowledges the limitations inherent in predictive modeling, particularly the challenge of capturing the full range of factors that influence student behavior and success. External factors, such as personal circumstances or non-academic pressures, may affect outcomes in ways that are difficult to quantify. Future research could explore the integration of additional data sources or the development of more sophisticated models to better account for these variables.

In conclusion, this research provides a valuable framework for using machine learning to enhance educational outcomes in computer science programs. By identifying at-risk students early and enabling targeted interventions, institutions can better support student success, ultimately leading to higher retention rates and more graduates prepared to enter the field of computer science.

References

[1] C. Watson and F. W. Li, "Failure rates in introductory programming revisited," in *Proceedings of the 2014 Conference on Innovation & Technology in Computer Science Education*, ser. ITiCSE '14. New York, NY, USA: Association for Computing Machinery, 2014, p. 39–44. [Online]. Available: https://doi.org/10.1145/2591708.2591749

[2] C. Alvarado, C. B. Lee, and G. Gillespie, "New cs1 pedagogies and curriculum, the same success factors?" in *Proceedings of the 45th ACM Technical Symposium on Computer Science Education*, ser. SIGCSE '14. New York, NY, USA: Association for Computing Machinery, 2014, p. 379–384. [Online]. Available: https://doi.org/10.1145/2538862.2538897

[3] B. Harrington, S. Peng, X. Jin, and M. Khan, "Gender, confidence, and mark prediction in cs examinations," in *Proceedings of the 23rd Annual ACM Conference on Innovation and Technology in Computer Science Education*, ser. ITiCSE 2018. New York, NY, USA: Association for Computing Machinery, 2018, p. 230–235. [Online]. Available: https://doi.org/10.1145/3197091.3197116

[4] P. Golding, L. Facey-Shaw, and V. Tennant, "Effects of peer tutoring, attitude and personality on academic performance of first year introductory programming students," in *Proceedings. Frontiers in Education. 36th Annual Conference*, 2006, pp. 7–12.

[5] C. Stephenson, A. D. Miller, C. Alvarado, L. Barker, V. Barr, T. Camp, C. Frieze, C. Lewis, E. C. Mindell, L. Limbird, D. Richardson, M. Sahami, E. Villa, H. Walker, and S. Zweben, *Retention in Computer Science Undergraduate Programs in the U.S.: Data Challenges and Promising Interventions*. New York, NY, USA: Association for Computing Machinery, 2018.

[6] S. Krause-Levy, L. Porter, B. Simon, and C. Alvarado, "Investigating the impact of employing multiple interventions in a cs1 course," in *Proceedings of the 51st ACM Technical Symposium on Computer Science Education*, ser. SIGCSE '20. New York, NY, USA: Association for Computing Machinery, 2020, p. 1082–1088. [Online]. Available: https://doi.org/10.1145/3328778.3366866

[7] L. Lyon, C. Schatz, Y. Toyama, and D. Torres, "Computer science intensive intervention to prepare and engage underrepresented novice students at community college," *Community College Journal of Research and Practice*, vol. 46, pp. 1–13, 03 2021.

[8] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[9] T. Chen and C. Guestrin, "XGBoost: A scalable tree boosting system," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '16. New York, NY, USA: ACM, 2016, pp. 785–794. [Online]. Available: http://doi.acm.org/10.1145/2939672.2939785

[10] L. Porter and B. Simon, "Retaining nearly one-third more majors with a trio of instructional best practices in cs1," 03 2013, pp. 165–170.