Utilizing Machine Learning to Predict Student Success and Improve Course Resource Management in Computer Science
(Technical Topic)

Understanding the Current Shortcomings of Diversity and Inclusion in Computing Education that Limit Success Rates of Underrepresented Minority Students
(STS Topic)

A Thesis Project Prospectus Submitted to the

Faculty of the School of Engineering and Applied Science
University of Virginia • Charlottesville, Virginia

In Partial Fulfillment of the Requirements of the Degree
Bachelor of Science, School of Engineering

Zachary Goldstein

Fall, 2020

Technical Project Team Members:

Zachary Goldstein

On my honor as a University student, I have neither given nor received unauthorized aid on this assignment as defined by the Honor Guidelines for Thesis-Related Assignments.

Signature _____

Approved _____ Date _____
Kathryn A. Neeley, Associate Professor of STS, Department of Engineering and Society

Approved _____ Date _____
Aaron Bloomfield, Assistant Professor, Department of Computer Science

Introduction

The U.S. Bureau of Labor Statistics estimates that the growth in software developer positions alone is expected to be 22% from 2019 to 2029 (Bureau of Labor Statistics, 2020). Unsurprisingly, the number of students interested in computer science and related fields has skyrocketed, with over a 100% increase in undergraduate students in the past decade as illustrated by Figure 1. However, as the interest in CS increases, the challenges of helping students succeed become more complex given the increasing variety of technical, ethnic, and socioeconomic backgrounds. Without understanding the factors that dissuade students from pursuing or limit success in computing degrees, instructors and computing departments will continue to fail to properly allocate resources that would otherwise improve the situation of struggling students.

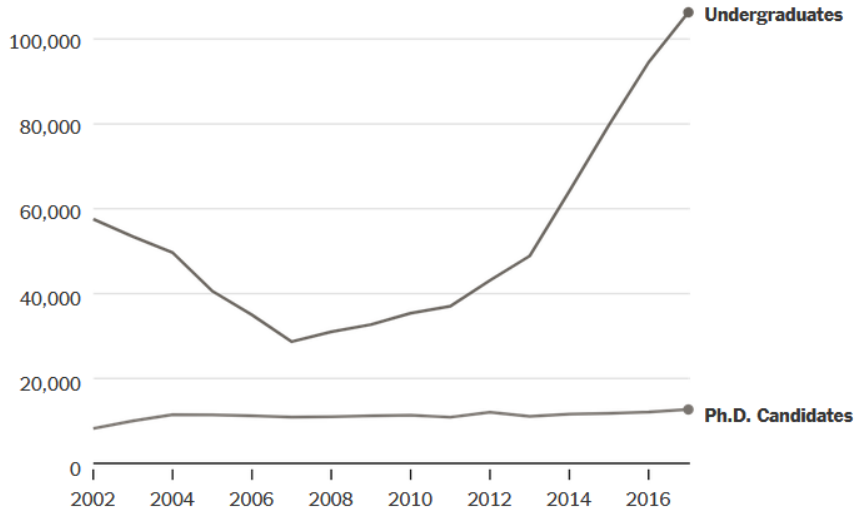

Figure 1: Number of CS undergraduate and Ph.D. students enrolled in American universities from 2002 to 2017. The number of undergraduate students is increasing rapidly. (NYTimes, 2019)

My technical project will investigate the feasibility to apply code complexity analysis and machine learning to determine if there are correlations between code patterns and graded

outcomes that, when combined with other student variables, can be used to develop a model to predict student success through machine learning. On the other hand, my STS topic looks to understand the factors that reduce retention rates and engagement in CS courses from an ethnic and racial perspective, with particular focus on underrepresented minority (URM) students. Leonard et al. (2013) examined that for engineering students, a "lack of interaction with student organizations and technical societies, as well as with peers and faculty can have a significant impact on higher-education degree completion." The STS research will analyze the missing or lacking support systems and social group opportunities for URMs in the early stages of their post-secondary computing education.

Technical Topic: Utilizing Machine Learning to Predict Student Success and Improve Course Resource Management

Transitioning from introductory to intermediate computer science courses, computing students are faced with projects and assignments that require longer code and provide less structure and guidance in their implementation. Although students are expected to be prepared for this additional workload from prior experiences, Mansur (2020) explains that "in intermediate (post CS2) programming courses, many students fail to complete one or more projects on time," inhibiting their success in the course. If students that are struggling to complete their large assignments cannot be identified, instructors will not be able to properly allocate their limited instructional resources to help students that are struggling succeed in the course. One potential reason for this shortcoming could be a limited grasp on the target concept material, but less attention is given to what degree student code quality (or lack thereof) and complexity play a role in the ability to complete assignments and succeed in these courses. Research conducted by Mansur suggests that student coding habits and strategies (e.g., time

management, testing, debugging) play a key role in the successful completion of large coding assignments. As such, the technical project will primarily focus on the feasibility to determine student success (i.e. course grades) from students' submitted code and background factors via machine learning. If at-risk students are detected early, instructors will be able to better allocate resources to those students that are at risk of dropping out or failing the course.

To build such a model that can predict student success, data from students that previously enrolled in the course must be gathered. As mentioned previously, the areas of interest are both students' code and their backgrounds. However, because source code is difficult to analyze directly, the technical component will first look into designing a preprocessing algorithm to examine students' code automatically. Common metrics, such as commenting and modularity, can be parsed and summarized from the source code. Another metric of interest is cyclomatic complexity – a metric used for software to indicate how complex a length of code is. A high cyclomatic complexity is typically positively correlated with an increased number of faults and errors, which makes it an area of interest by software testing teams in industry (Watson et al., 1996, pg. 1). However, cyclomatic complexity hasn't been imported to perform a static complexity analysis of student source code, which could prove to be a valuable data source. By performing these analyses, programming submissions can be converted to structured, statistical data. The data points will then be matched and tabulated with background factors such as gender, school year, and past grades (see Figure 2 for additional potential factors) to build a student profile. These student profiles characterize a student's situation and help the analysis find similarities and differences across unique students.

Code Factors	Background Factors	Target Label
Number of methods Number of lines Code commenting Cyclomatic complexity Debugging strategy Language features	Gender School year Assignment grades Exam grades Past courses Past course grades Time spent on assignments	Course grade

Figure 2: Table representing potential coding-related factors and background factors to collect data on. The target label is the variable to predict. (Created by Author)

Once the student profiles have been gathered, machine learning algorithms can be used to analyze the dataset and produce a model (see Figure 3). Machine learning is an application of artificial intelligence where computer programs systematically improve their decision-making ability through experience (i.e., by analyzing available data). The technique has had success in predicting student course grades. For example, using in-class questions as the primary input, machine learning has been able to predict student outcomes in different courses reasonably well (Liao et al., 2019, pg.13). Categorical machine learning algorithms, such as random forest and logistic regression classifiers, will be applied across the available variables mentioned previously to develop a model that can be used to predict and label students that are struggling in a course. In other words, machine learning algorithms can identify patterns and correlations between coding-related factors, background factors, and course grades. Statistically significant correlations between one or many factors and the outcome mean that there are certain conditions that significantly increase or decrease student performance and success. If a correlation is successfully identified, then a model incorporating those correlations can both be utilized in the present to inform instructors which students to allocate resources. Additionally, common limiters of success can be used to inform long-term planning to improve course delivery for introductory and intermediate CS classes in future semesters.

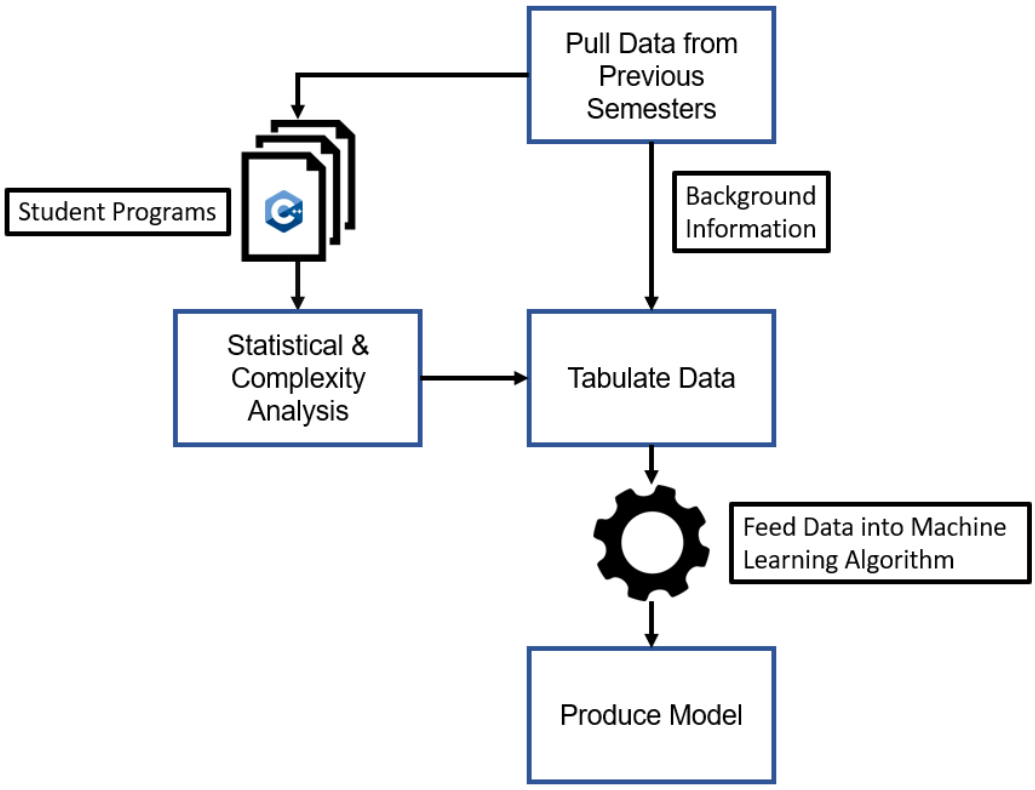

Figure 3: Code is collected and fed through algorithms to collect complexity and statistical data. This data, along with background information, is put into a single data set which is used to generate a predictive model via machine learning. (Created by Author)

STS Topic: Understanding the Current Shortcomings of Diversity and Inclusion in Computing Education that Limit Success Rates of Underrepresented Minority Students

At many universities, the University of Virginia included, the proportion of students in underrepresented minority (URM) groups in computing education and general engineering is significantly below that of the overall student population. As seen in Figure 4, URMs comprised less than 40% of all computing degrees conferred in 2015. However, the success and retention rates for underrepresented racial groups, such as African American and Hispanic, are also significantly lower than other groups, particularly Caucasian. Martin (2018) points out that "graduation rates for URMs are declining" with recent graduation rates for African American and Latinx students being approximately 10.8% and 5.4-6.4% respectively for all college graduates across all majors. This trend is not new, and without identifying the missing

opportunities for URMs in the early stages of their post-secondary computing education, this disparity will persist. Therefore, a careful examination of differences in financial support, access to academic resources and student groups, and interaction with faculty and advisors could provide insight into how to better serve these students (Leonard et al., 2013, pg. 2).

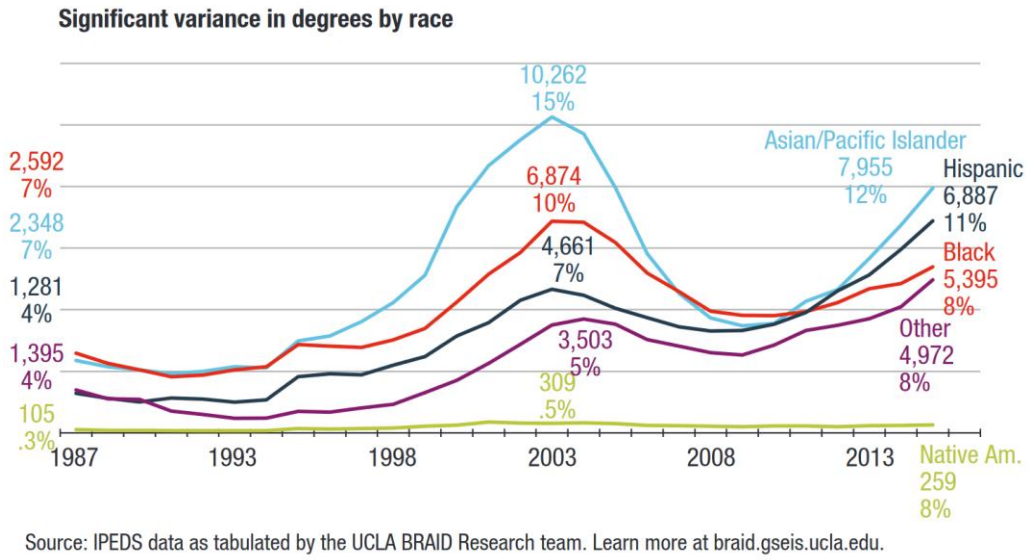

Figure 4. Number of bachelor's computing degrees conferred to minorities in the US by race/ethnicity (1987-2015) (Whitney et al., 2018, p. 29)

Although a difference in financial background mentioned previously is a well-documented problem, the key issue of interest is that underrepresented minorities fail to engage and form social groups of a similar ethnic and cultural background in a major that is predominately male and Caucasian (Leonard et al., 2013, pg. 1). In particular, environments with a lack of diversity, such as engineering and computing and information sciences, may cause URM students to feel that they don't belong in the major. One difficulty comes from the lack of diversity in terms of applicants enrolled in secondary education. Applicant asymmetry continues to be a systemic problem that requires intervention during primary and a secondary education with primarily minority serving schools. Despite this shortcoming, failing to promote inclusion and support current URM university students will also continue to perpetuate a significant

asymmetry between ethnic and racial groups that achieve computing majors. Cintron et al. (2019) recognize that computing students may need "different types and levels of support for students from" URM and non-URM backgrounds; failing to determine what support is necessary will hinder retention and engagement.

Therefore, improving URM retention and success requires identification of the missing support systems and obstacles that make underrepresented minority students less inclined to pursue or stay in computing. Through an actor-network approach, I will investigate how URMs interact with peers, advisors, instructors, and student organizations. The research will first analyze historical data, as exploring which courses URM students are disproportionately stopping at or dropping mid-semester will help determine whether or not URMs are receiving a different quality of education or support than their peers. The research will also investigate student sentiment regarding their computing experience to identify pain points along the process that are less likely to be displayed by non-URM students. From here, the research will provide further directions or possible solutions based on the evidence and discoveries found throughout the process. By mapping the relationships between these actors and understanding which factors significantly benefit or hinder URM retention and success, computing education faculty can design better support systems and courses and promote inclusion and diversity.

Conclusion

Investigating the underlying causes that inhibit student success in computing education provides the potential to improve student outcomes at the University of Virginia and other post-secondary institutions. The technical component of the thesis project applies across all students analyzes if specific programming techniques gained from prior experiences have a direct correlation to graded outcomes to a statistically significant degree. If so, models built using

machine learning may be constructed that can help direct resources to students in need of additional assistance. On the other hand, the STS component narrows in on the dimension of diversity and aims to look at ways that underrepresented minority students are disproportionately affected by the current, relatively undiversified social environment in computing and engineering as a whole. In this case, mapping the points of interaction between URMs and other actors may provide critical insight into how computing departments can increase diversity and inclusion as students commit to and declare their majors. If both of these objectives are completed successfully, computing departments can further investigate ways to make changes at the course delivery level and a student support and inclusion level.

References

Cintron, L., Chang, Y., Cohoon, J., Tychonievich, L., Halsey, B., Yi, D. & Schmitt, G. (2019) Exploring underrepresented student motivation and perceptions of collaborative learning-enhanced CS undergraduate introductory courses. *2019 IEEE Frontiers in Education Conference (FIE)*, Covington, KY, USA, pp. 1-6, doi:10.1109/FIE43999.2019.9028463

Cohoon, J. & Tychonievich, L. 2020. Lessons learned from providing hundreds of hours of diversity training. *Proceedings of the 51st ACM Technical Symposium on Computer Science Education (SIGCSE '20).* Association for Computing Machinery, New York, NY, USA, 206–212. doi:10.1145/3328778.3366930

Jaime (2018). An in-depth explanation of code complexity. *Codacy.* Retrieved from: https://blog.codacy.com/an-in-depth-explanation-of-code-complexity/

Leonard, S., Pearcy, B., Shehab, R. & Walden, S. (2013) Minority student informed retention strategies. *2013 IEEE Frontiers in Education Conference (FIE)* Oklahoma City, OK pp 567-573, doi:10.1109/FIE.2013.6684887

Liao, S., Zingaro, D., Laurenzano, M., Griswold, W. & Porter, L. (2016). Lightweight, early identification of at-risk CS1 students. *ICER '16: Proceedings of the 2016 ACM Conference on International Computing Education Research.* pp. 123-131. doi:10.1145/2960310.2960315

Liao, S., Zingaro, D., Thai, K., Alvarado, C., Griswold, W. & Porter, L. (2019). A robust machine learning technique to predict low-performing students. *ACM Transactions on Computing Education* 18. doi:10.1145/3277569

Mansur, R. (2020). Understanding coding behaviors in intermediate CS students. *SIGCSE '20: Proceedings of the 51st ACM Technical Symposium on Computer Science Education* doi:10.1145/3328778.3372694

Martin, S. (2018) Engineering retention: improving inclusion and diversity in engineering. *Murray State Theses and Dissertations*. 105 Retrieved from: https://digitalcommons.murraystate.edu/etd/105

Singer, N. (2019) The hard part of computer science? Getting into class. *NYTimes.* Retrieved from: https://www.nytimes.com/2019/01/24/technology/computer-science-courses-college.html

Watson, A. & McCabe, T. (1996) Structured testing: a testing methodology using the cyclomatic complexity metric. *NIST Special Publication 500-235*

Whitney, T & Taylor. V (2018) Increasing women and underrepresented minorities in computing: the landscape and what you can do. *Computer*, vol. 51, no. 10, pp. 24-31 doi:10.1109/MC.2018.3971359.