Early identification of struggling students in large computer science courses: A replication study

Nils Rys-Recker
Western Washington University
Bellingham, Washington
rysrecn@wwu.edu

Qiang Hao
Western Washington University
Bellingham, Washington
qiang.hao@wwu.edu

***Abstract*—The identification of struggling students in a scalable manner in large computer science courses continues to attract researchers' attention because of the high failure and dropout rates in such courses. In the last two decades, studies on this topic made significant progress in taking advantage of dynamic data sources and employing machine learning techniques to identify struggling students. However, the success of these studies was still limited due to reasons such as oversimplification and utilizing exclusive tools. These limitations make it difficult to replicate the findings of prior research on this topic. To address these issues and explore the extent to which we can replicate studies on this topic, this study replicated a recent study that explored identifying struggling students at the topic level using context-agnostic features. Our results demonstrate the potential of context-agnostic features in identifying struggling students with varied success rates. Our findings shed light on the robustness and feasibility of using machine learning techniques to identify struggling students in large computer science courses. Finally, our discussion provides useful guidance on future studies and replications on this topic.**

I. Introduction

Computing science as a discipline continues to attract an increasing number of students in the last two decades. As the expansion continues, the high failure and dropout rate in computer science remains a challenge to be addressed by researchers and educators. As a result, computing education research witnessed increasing interest in how to help students in a scalable manner, starting with early identification of struggling students. By identifying struggling students early, instructors can provide guided help in a timely manner, and use the limited resources strategically to help those who need it the most, thus reducing the overall failure and dropout rates.

The approaches to identifying struggling students have evolved over the years. Early studies on this topic tend to focus more on static factors aiming at predicting student performance through quizzes and surveys conducted prior to learning [1]–[3]. These studies have yielded mixed results. Recent studies have shifted towards using dynamic factors to predict student performance [4]–[15]. Dynamic factors refer to those factors that can be observed and constantly measured as students engage with learning activities. The usage of dynamic factors contributes to a more in-depth understanding of students' day-to-day learning and potential struggles.

The collection of dynamic data heavily relies on different technologies because of the high data collection frequency and scale. The technologies range from clickers in classrooms to software that tracks student coding behaviors. Some approaches to dynamic data collection pose bigger barriers than others for researchers to replicate. The barriers can take the form of large initial costs, or software that is not public for others to access. Some studies took advantage of hardware such as clickers that might not be available elsewhere [4]–[6], while others use specialized software in the form of IDE integration [10], [16] or course management systems [7]–[9].

The use of version control systems has also been brought to the forefront by recent studies on this topic [15], [17], [18]. With the great importance of version control in industry and its extensive use in classrooms, version control systems have the potential to record a student's work at a fine-grained level by storing many snapshots of student code repositories. When combined with a continuous integration tool that supports automated testing on each of the stored snapshots, it has the potential to accurately measure student progress constantly and accurately. A notable study that utilized version control systems in tandem with continuous integration on this topic is the study of Arakawa et al. [19]. This study used the collected data to build and compare four different machine learning models for identifying struggling students at the topic level using only context-agnostic features.

To gain further insights into the feasibility of identifying struggling students at the topic level using context-agnostic features, we replicated the study conducted by Arakawa et al. We were guided by the following research questions:

1) To what extent can the results reported by Arakawa et al. be replicated?
2) To what extent can we identify student struggles in programming courses at the topic level using context-agnostic features?

Our first research question aims to explore and evaluate the robustness and generalizability of the reported approach of Arakawa et al. in identifying struggling students. Our second research question is the same as that of Arakawa et al. To answer this question, we adopted the same research design, utilized a system that combined the power of version control and continuous integration, and collected data from an entry-level programming course. Similarly, we trained and evaluated machine learning models that identify struggling students at the topic level using only context-agnostic features.

II. Background

A. Earlier Studies using static factors

Early studies in identifying struggling students focused more on static factors. Static factors refer to attributes that are unlikely to change during the period of studies, such as biographical information, socioeconomic status, or aptitude levels. More precisely, two approaches were used to study using static factors to identify struggling students.

A common approach was to use demographic information, socioeconomic status, or prior academic performance to predict student performance. An example of this approach is the study conducted by Newsted [1]. In this study, 472 students across three semesters in an introductory FORTRAN programming course were given 18-question questionnaires. Only 28% (131) valid questionnaires were returned. The generated regression models were able to explain 41% of the variance in a student's grade. Due to the limits of data collection and study design, the variance that can be explained by static factors was relatively low. The other approach was to use aptitude tests to measure student proficiency levels and use such data to identify struggling students. An example was the study of Kurtz [2]. In this study, an aptitude test measuring ten areas of formal reasoning such as correlational reasoning was distributed. Via using aptitude tests, Kurtz was able to explain 66% of the variance regarding letter grades. Barker and Unger [3] built upon the study of Kurtz and utilized an abridged version of the aptitude test. They examined 353 students across 10 instructors in 15 class sections with two different programming languages. Their findings were significantly different from Kurtz's. Barker and Unger achieved a correlation of 0.12 between ID-level and final grade in comparison to 0.80 from Kurtz's study.

While static factors showed some potential in predicting struggling students, they lack reliability and have relatively poor generalizability. As data on static factors is often collected only at the beginning of the learning, it often ignores students' day-to-day changes, progress, and struggles.

B. Recent Studies using dynamic factors

More recent studies have shifted from using static factors to using dynamic factors to identify struggling students. Dynamic factors refer to the factors that are dynamically changing constantly during the study time and thus must be collected as frequently as possible. Four common approaches were explored in using dynamic data to identify struggling students.

The first approach explored the usage of clickers in measuring student learning on a daily basis. Clickers are small handheld devices that Kenwright defines as allowing students to "answer questions in the form of quizzes or self-assessment, perform instructor or course evaluations, and/or record attendance" [20]. Porter et al. [4] utilized clicker quizzes in order to gather data for assessing student understanding and predicting student end-of-term outcomes. In this study, three different types of clicker questions were administered: Individual Votes, Group Votes, and Classwide Discussions. The findings of Porter et al. [4] showed that clicker performance correlated highly with final exam performance with a correlation of 0.64. Building upon this study Liao et al. [5] incorporated machine learning models based on clicker questions for predicting struggling students during separate terms. They found success in using a linear regression model which was able to accurately predict 70% of students as either struggling or not struggling with just the first three weeks as training data.

The second approach explored using the data on student interactions with course management systems to identify struggles. Zefra et al. [7] utilized this method in their study and collected data on student interaction activities, such as time spent on assignments. They achieved a 0.743 accuracy for predicting student final performance using a multi-instance grammar-guided genetic programming model. Fire et al. [8] tracked student social networks through individual and group assignments. By combining social network graph features with student grades the study produced a Linear Regression model with a variation of 17.4% and a Rotation Forest classifier with an AUC of 0.672 in determining which students were most likely to fail.

The third approach explored tracking student behavioral data via IDE plugins and specialized IDES, and used such data to identify struggling students. The data that was collected typically involved keystrokes and IDE compilation events. An example of this approach is the study conducted by Jadud [10] in which they used compilation behavior to measure error quotients, which denoted how much a student struggles with syntax errors. These error quotients, however, were found to have a low-quality correlation with student average grades, and student final exam scores. Petersen et al. [11], also found that the predictive power varied greatly across contexts using error quotients, with in some cases being unable to effectively predict performance at all. To address this issue, Becker [16] built upon error quotients with a new metric, repeated error density, which measured repeated errors. By applying both metrics to a dataset containing 29,019 error events, Becker argued that repeated errors were a strong indicator of struggling students and that repeated error density was promising as a less context-dependent metric.

The fourth approach explored the use of version control systems in combination with other dynamic data and used combined data to identify struggling students. Guerrero-Higueras et al. [17] explored tracking student interactions with a different version control system Git, such as the number of commits. They used the collected data to develop and evaluate different machine learning models, and found proof that "commits, additions, days, and commits/day are the most discriminant" factors in predicting student performance [17]. Sprint and Conci [18] studied using GitHub Classroom to collect data on student commit behaviors, and the extent to which the behaviors can predict student performance. They found a weak correlation between commit behavior and grades at a group level, but a strong correlation at the individual level. Arakawa et al. also utilized Git in tandem with the continuous integration tool Travis-CI. They focused on collecting only context-agnostic features to identify struggling students at the topic level. By doing so, they attempted to address two main concerns: (a) Student learning process is dynamic and full of changes, in that students may struggle on multiple learning topics. and (b) A reduced reliance on classroom-specific features helps increase the replication possibilities. Arakawa et al. were able to collect in-depth information, while not being intrusive in the students' programming experience. They tested and compared multiple machine learning models, and achieved an AUC value of over 80% using only the first two weeks' data of a semester.

C. Why Replication?

This study aims to replicate the study conducted by Arakawa et al. Replication studies play an important role in ensuring the reliability and generalizability of published studies [21]. Although most scientists understand the importance of replications, replication studies constitute a very minor portion of computing education research. Hao et al. [21] found that only 2.12% of publications from 2009 to 2018 were replication studies in the field of computing education. Particularly, replications on the topic of identifying struggling students are extremely rare. The difficulties in data collection, heavy reliance on context-dependent features, and lack of access to systems used in prior research all contribute to the lack of replications on this topic.

As a replication study, our study can shed light on the extent to which prior findings on this topic can be replicated. We chose to replicate the study of Arakawa et al. for the easiness of replicating their process of data collection and analysis, and the implication for practitioners. Many studies on this topic relied on instruments that were only accessible to the researchers, such as course management systems that were still being tested, or tools with limited accessibility, such as clickers. In contrast, the study of Arakawa et al. only relied on two pieces of tools that are available to any researchers and practitioners, Git as the version control system and Travis-CI as the continuous integration tool. If the identification of struggling students can be context-agnostic and efficient, it will have direct implications for research on this topic.

III. Research Design

Our research design followed the same design philosophy as Arakawa et al. We collected data from students at the topic level to avoid oversimplifying the definition of struggling students. To achieve this, we collected data for each programming assignment in an entry-level programming course with 41 students at a large undergraduate institution in the northwestern United States. There were three different programming assignments, and each assignment covered a different topic, ranging from *sorting*, *searching*, to *graph*.

Same as the original study, we adopted version control and continuous integration tools to track student learning behaviors and performance. Same as in the original study, we used Git and GitHub as the version control system. Different from the original study, we used GitHub Actions instead of Travis-CI as the continuous integration tool, considering that GitHub has a wider adoption by practitioners in computing education, runs faster, and is free to use by educators. Both GitHub Actions and Travis-CI serve the same purpose and don't affect the research design. When a student commits and pushes his or her code to GitHub, GitHub Actions will be triggered to automatically test the student's code against a set of prepared test cases. When testing is completed, GitHub Actions will notify students of the testing results, which contain detailed feedback corresponding to each test case. Both GitHub and GitHub Actions provided standardized RESTful APIs that allow the authorized parties to collect data on each commit and push, as well as the results of each triggered test. Through the APIs of GitHub and GitHub Actions, we were able to track fine-grained student coding behaviors and the testing results of every commit and push in their coding repositories.

In this study, we collected 1,134 commits across 120 student submissions for three assignments. Each student submission is represented by a distinct student GitHub repository, from which each commit and push triggered the automated testing powered by GitHub Action. Same as the original study, we collected data on the following context-agnostic features:

- *Normalized timestamp*: The timestamp of when a push was done, in which a 0 denotes the assignment start date, and a 1 denotes the assignment due date
- *Additions*: The number of lines of code added to the previous push
- *Deletions*: the number of lines of code removed from the previous push
- *Test ratio*: The ratio between the amount of test cases passed to the number of total test cases.
- *Error ratio*: The ratio between the number of previous pushes that failed to compile versus the number of previous pushes
- *First commit timestamp*: The normalized timestamp of the first commit.

TABLE I
THE CUMULATIVE PERFORMANCE RESULTS SHOWN AS AUROC SCORES OF OUR MODELS WITH DIFFERENT TRAINING AND TESTING DATA

Rounds	Training Data Set	Testing Data Set	RNN	LSTM	Wide RNN	Wide LSTM
	Replication Target	Replication Target	0.792 ± 0.035	0.771 ± 0.038	0.776 ± 0.030	0.784 ± 0.023
Round 1	Replication Target	New Data	0.673 ± 0.118	0.677 ± 0.151	0.684 ± 0.101	0.680 ± 0.116
Round 2	Replication Target & New Data	New Data	0.668 ± 0.095	0.754 ± 0.075	0.730 ± 0.101	0.767 ± 0.089
Round 3	New Data	New Data	0.662 ± 0.054	0.687 ± 0.090	0.636 ± 0.109	0.672 ± 0.100

Note: Replication target refers to the data from the study of Arakawa et al. New Data refers to the data set collected in this study.

- *Number of pushes with no progress*: The number of pushes from the student in which the test ratio has not increased
- *Highest test ratio*: the maximum test ratio achieved thus far by the student

We defined struggling students the same as in the study of Arakawa et al. - By the time an assignment is due, if a student fails at least one test case, the student is considered struggling. The conventional approach in defining struggling students is typically classifying students into two groups, struggling and not struggling, based on their overall performance or performance in the final exam. The conventional approach fails to account for the nuances during the student's learning process. Most importantly, if the ultimate goal is to provide just-in-time help to those who struggle, it is pressing to know which student is struggling on what topic in time. The definition of struggling students from Arakawa et al., therefore, enables us to provide in-time feedback.

We conducted the same data analysis of Arakawa et al., applying four machine learning models to explore if the context-agnostic features can identify struggling students. The four models include recurrent neural network (RNN), long short-term memory network (LSTM), Wide RNN, and Wide LSTM. To answer the two research questions, we conducted three rounds of analysis using each machine learning model:

- *Round 1*: We used the dataset from Arakawa et al. to train machine learning models, and test the models against the new data.
- *Round 2*: We used the dataset from Arakawa et al. in combination with the new data to train machine learning models, and test the models against the new data.
- *Round 3*: We used the new data to train machine learning models, and test the models against the new data.

To make our results comparable to the original study, we used an area under the receiver operating characteristic (AUROC) for the evaluation metric and utilized 5-fold stratified cross-validation to mitigate heterogeneity in the dataset.

IV. Results

Same as the original study, we evaluated each model's cumulative performance through making predictions on every commit sequence in the training dataset. The predictions were used to calculate the ROC, in order to understand how well the model performs in terms of making both early and late predictions. The results of our four models are summarized in Table 1. We can see that on average the best-performing training data variation was from *Round 2*, followed by *Round 1*, then *Round 3*; The best-performing training data set was the combination of the newly collected data and the data from the original study. The highest AUROC for identifying struggling students in the new data set was Wide LSTM with an AUROC of 0.768. The variations for identifying struggling students in the new data are found to be larger than in the original study, with an average variation of 0.099 in comparison to 0.032 in the replication models.

To compare how early the mean AUC can be achieved in comparison to the original study, we summarized the time into the assignment in order to achieve a mean AUC of at least 0.7 in Table 2. Considering that the duration of assignments were different in the original study from this one, we computed the percentage of time instead of days. Different from the original study, none of our models achieved a mean AUC greater than 0.8. However, all of our replication models were able to achieve a mean AUC greater than or equal to 0.7. The model that achieved a mean AUC of 0.7 the earliest was the wide LSTM, using 66.14% of the assignment time. For example, if an assignment lasts for 14 days, the wide LSTM can achieve a mean AUC of 0.7 in 9.26 days.

The comparisons between our results and the original study are presented in Tables 3 and 4. Table 3 compares the cumulative performance of each model type, with the performance measured by the AUROC score. The replication models performed worse on average by 0.134. Table 4 compares the real-time efficiency of each model type. The replication models were all slower, requiring on average 43% longer into the assignment to achieve the same milestone of a 0.7 AUC mean.

TABLE II
THE EARLIEST A MODEL CAN MAKE A PREDICTION INTO AN ASSIGNMENT IN ACHIEVING A DECENT AUC

Model Type	Mean AUC $\geq$ 0.7	AUC lower bound $\geq$ 0.7
RNN	75.59%	97.64%
LSTM	91.34%	–
Wide RNN	93.70%	–
Wide LSTM	66.14%	97.64%

Note: The percentage numbers represent the percentage of time into an assignment. For example, if an assignment lasts for 14 days, 66.14% of that duration is about 9 days.

TABLE III
THE CUMULATIVE PERFORMANCE COMPARISON

Model Type	Replication Results	Arakawa et al.'s Results
RNN	0.673 ± 0.118	0.821 ± 0.011
LSTM	0.754 ± 0.075	0.910 ± 0.0021
Wide RNN	0.730 ± 0.101	0.910 ± 0.0034
Wide LSTM	0.767 ± 0.089	0.922 ± 0.0019

Note: The comparisons are measured by AUROC scores

V. Discussion

A. To what extent can the results reported by Arakawa et al. be replicated?

We confirmed the streamlined process of data collection in the study of Arakawa et al. Arakawa et al. emphasized the importance of minimizing the complexity and overhead of data collection for future replications. Version control and continuous integration systems are generally available to practitioners of computing education, and their implementations (e.g., GitHub) typically provide a set of well-documented APIs for educators and researchers to collect data on their students, which makes it not only possible but also easy to conduct a replication study. More importantly, using such

TABLE IV
THE COMPARISON ON HOW EARLY A MODEL CAN IDENTIFY STRUGGLING STUDENTS

	Replication Results		Arakawa et al.'s Results	
Model Type	Mean AUC $\geq$ 0.7	AUC lower bound $\geq$ 0.7	Mean AUC $\geq$ 0.7	AUC lower bound $\geq$ 0.7
RNN	75.59%	97.64%	55.14%	100%
LSTM	91.34%	-	28.35%	44.85%
Wide RNN	93.70%	-	27.57%	30.71%
Wide LSTM	66.14%	97.64%	17.35%	17.35%

Note: The comparisons are measured by the percentage of time into an assignment

non-obtrusive approaches to collect data from students has an advantage over other reported approaches - students are less likely to be unrepresented in the collected data. Unlike surveys in which not every student would answer [1], or clicker quizzes which are unable to collect data on students who miss a day [22], version control systems are baked into the assignment submission process. Continuous integration also provides a large benefit by motivating a consistent stream of student submissions through in-time feedback and therefore allowing for more consistent data collection. We found that the overhead in data collection is greatly minimized because data is automatically generated and can be automatically collected, therefore not requiring us to perform extra tasks. The approach of data collection reported by Arakawa et al. has the potential to enable more replication studies on this topic. More importantly, it sheds light on how the research on this topic may have a real-world impact by minimizing the barrier of entry to data collection and analytics for practitioners in computing education.

We reached similar findings as Arakawa et al. However, we did not achieve training models as powerful as the original study in comparison. The replication models trained in this study only achieved reasonably good results in terms of identifying struggling students at the topic level using context-agnostic features. The cumulative performance of our replication models underperformed those reported by Arakawa et al. Measured by AUROC, our models underperformed on average by 0.134. In addition to that, it also takes longer for our models to identify struggling students on a topic. On average, it requires 45% longer than time into an assignment for our replication models to achieve the same milestone (0.7 AUC) than the counterparts reported in the original study.

The differences between this and the original study are likely due to the differences in sample sizes. When we trained the models only using the new data or the data from the original study, the models did not perform as well as the ones trained using a combination of new data and the data from the original study. LSTM, Wide RNN, and Wide LSTM all achieved reasonably high AUROC values. Additionally, it is worth noting that although our replication models underperformed their counterparts in the original study, our models had the same rankings. For example, RNN underperformed models trained using LSTM, Wide RNN, and Wide LSTM in both the original study and this study. For another example, Wide LSTM is the best-performing model in both the original study and this study. Given all these similarities in our findings, the reduced effects in our study are likely due to the comparatively limited testing data size. That said, although this study verifies that struggling students can be identified in a relatively small or medium-sized computer science course, it may come with a varied degree of success.

B. To what extent can we identify student struggles in programming courses at the topic level using context-agnostic features?

Our replications verify that it is possible to identify struggling students at the topic level. Most studies on this topic chose to identify struggling students on their overall performance or performance on the final exam. The study of Arakawa et al. hypothesized that struggling students could be identified along their learning progress on each individual topic, and explored its potential. Our replications further verify the potential to identify struggling students at the topic level. By doing so, we are able to catch two types of struggles that prior studies would miss when the overall performance is used as the target to evaluate struggling students. The first type is when a student passes the course while actually having struggled a lot during the learning; The second type is a student that a model determines will pass, while in reality they struggle on all topics and will fail the course. Although the first type still has the student finding success overall, they still may face very real struggles with specific topics that will go unidentified. By focusing on the topic level these students have the chance of being detected and therefore provided with the assistance they need. The second case is much more severe in scale. When focusing on the overall performance a misclassification means the student will be completely missed, essentially abandoned. By decreasing the scope to the topic level, missing a student once still allows struggles to be identified on different topics, therefore minimizing the potential damage of making a single misclassification.

Our replications also verify that it is possible to identify struggling students using context-agnostic features. The study of Arakawa et al. proposed and explored the potential of identifying struggling students using context-agnostic features. Our replications achieved better or similar results using small-sized testing data in comparison to the results from other studies on the same topic. For example, our best-performing model, Wide LSTM, achieved an AUROC score better than the model trained on clicker data reported by Liao et al. [6], which achieved an AUROC value of 0.76. For another example, our Wide LSTM model performed better than the model trained on student social networks produced by Fire et al. [8], which achieved an AUROC value of 0.672. Some prior studies used accuracy, instead of AUROC, to measure their model performance. Although accuracy is not the best measurement for extremely unbalanced classification tasks, such as struggle identification, our models still performed on par with the reported results using such a measurement. Zefra et al.'s [7] multi-instance grammar-guided genetic programming models achieved an accuracy of 0.743, and the models produced by Ahadi et al. [12] also achieved similar results, with their models predicting performance across semesters achieving an accuracy from 71% to 80%. Only the model reported by Castro-Wunsch et al. [13] outperformed our replication models by achieving an accuracy of 85.29%. Based on the comparison between our results and other studies, we believe that context-agnostic features are capable of training machine learning models for identifying struggling students at the topic level in programming courses. Considering that data collection on such features can be easily automated, future studies are encouraged to explore this direction further.

VI. Limitations

Our greatest limitation is the size of the testing dataset. In order to gain an understanding of how well a model is able to generalize on a bigger scale, collecting data on varying topics from different computer science courses at different institutions would lead to more robust results. Future studies or future replications on this topic may consider increasing the data size of testing data when possible.

VII. Conclusion

In this study, we replicated the study conducted by Arakawa et al. to answer two research questions, including "*To what extent can the results reported by Arakawa et al. be replicated?*" and "*To what extent can we identify student struggles in programming courses at the topic level using context-agnostic features?*" We replicated the results reported by Arakawa et al. with a varied degree of success. Our results demonstrated the potential of context-agnostic features in identifying struggling students. Our findings shed light on the robustness and feasibility of using machine learning techniques to identify struggling students in large computer science courses. Our discussion provided useful guidance on future studies and replications on this topic.

References

[1] P. R. Newsted, "Grade and ability predictions in an introductory programming course," *ACM SIGCSE Bulletin*, vol. 7, no. 2, pp. 87–91, 1975.

[2] B. L. Kurtz, "Investigating the relationship between the development of abstract reasoning and performance in an introductory programming class," in *Proceedings of the eleventh SIGCSE technical symposium on Computer science education*, 1980, pp. 110–117.

[3] R. J. Barker and E. A. Unger, "A predictor for success in an introductory programming class based upon abstract reasoning development," *ACM SIGCSE Bulletin*, vol. 15, no. 1, pp. 154–158, 1983.

[4] L. Porter, D. Bouvier, Q. Cutts, S. Grissom, C. Lee, R. McCartney, D. Zingaro, and B. Simon, "A multi-institutional study of peer instruction in introductory computing," in *Proceedings of the 47th ACM Technical Symposium on Computing Science Education*, 2016, pp. 358–363.

[5] S. N. Liao, D. Zingaro, M. A. Laurenzano, W. G. Griswold, and L. Porter, "Lightweight, early identification of at-risk cs1 students," in *Proceedings of the 2016 acm conference on international computing education research*, 2016, pp. 123–131.

[6] 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*, 2019, pp. 112–118.

[7] A. Zafra and S. Ventura, "Multi-instance genetic programming for predicting student performance in web based educational environments," *Applied Soft Computing*, vol. 12, no. 8, pp. 2693–2706, 2012.

[8] M. Fire, G. Katz, Y. Elovici, B. Shapira, and L. Rokach, "Predicting student exam's scores by analyzing social network data," in *Active Media Technology: 8th International Conference, AMT 2012, Macau, China, December 4-7, 2012. Proceedings 8*. Springer, 2012, pp. 584–595.

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

[10] M. C. Jadud, "Methods and tools for exploring novice compilation behaviour," in *Proceedings of the second international workshop on Computing education research*, 2006, pp. 73–84.

[11] A. Petersen, J. Spacco, and A. Vihavainen, "An exploration of error quotient in multiple contexts," in *Proceedings of the 15th Koli Calling Conference on Computing Education Research*, 2015, pp. 77–86.

[12] 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*, 2015, pp. 121–130.

[13] 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*, 2017, pp. 111–116.

[14] Z. Pullar-Strecker, F. D. Pereira, P. Denny, A. Luxton-Reilly, and J. Leinonen, "G is for generalisation: Predicting student success from keystrokes," in *Proceedings of the 54th ACM Technical Symposium on Computer Science Education V. 1*, 2023, pp. 1028–1034.

[15] K. Mierle, K. Laven, S. Roweis, and G. Wilson, "Mining student cvs repositories for performance indicators," *ACM SIGSOFT Software Engineering Notes*, vol. 30, no. 4, pp. 1–5, 2005.

[16] B. A. Becker, "A new metric to quantify repeated compiler errors for novice programmers," in *Proceedings of the 2016 ACM Conference on Innovation and Technology in Computer Science Education*, 2016, pp. 296–301.

[17] Á. M. Guerrero-Higueras, N. DeCastro-García, V. Matellán, and M. Á. Conde, "Predictive models of academic success: a case study with version control systems," in *Proceedings of the Sixth International Conference on Technological Ecosystems for Enhancing Multiculturality*, 2018, pp. 306–312.

[18] G. Sprint and J. Conci, "Mining github classroom commit behavior in elective and introductory computer science courses," *The Journal of Computing Sciences in Colleges*, vol. 35, no. 1, 2019.

[19] K. Arakawa, Q. Hao, W. Deneke, I. Cowan, S. Wolfman, and A. Peterson, "Early identification of student struggles at the topic level using context-agnostic features," in *Proceedings of the 53rd ACM Technical Symposium on Computer Science Education-Volume 1*, 2022, pp. 147–153.

[20] K. Kenwright, "Clickers in the classroom," *TechTrends*, vol. 53, no. 1, pp. 74–77, 2009.

[21] Q. Hao, D. H. Smith IV, N. Iriumi, M. Tsikerdekis, and A. J. Ko, "A systematic investigation of replications in computing education research," *ACM Transactions on Computing Education (TOCE)*, vol. 19, no. 4, pp. 1–18, 2019.

[22] L. Porter, D. Zingaro, and R. Lister, "Predicting student success using fine grain clicker data," in *Proceedings of the tenth annual conference on International computing education research*, 2014, pp. 51–58.