International Journal of Computer Information Systems and Industrial Management Applications
ISSN 2150-7988 Volume 18(22s) 2026 pp. 191-199
https://doi.org/10.70917/ijcisim-2026-5425
Article

Beyond Code Correctness: A Hybrid Behavioral-Cognitive Model for Student Assessment in Programming Education

Amitkumar Patel[1]*, Jaykumar Patel[2], Sandipkumar Surati[3]

[1]Vidyabharti Trust College of Business Computer Science and Research, Umrakh, BCA Department, Gujarat, India
[2]Department of Computer Science, Gujrat University, Gujarat, India
[3]Vivekanand College for Advanced Computer and Information Science, BCA Department, Gujarat, India
Corresponding Author: iamitrp@gmail.com, ORCID ID: 0000-0001-6607-1870

Abstract: Traditional programming assignment assessments primarily focus on evaluating the correctness of the final output, often neglecting critical aspects such as engagement, collaboration, and debugging skills. As programming education increasingly shifts toward project-based and collaborative learning, there is a growing need for more holistic evaluation methods. This paper proposes a hybrid conceptual framework that integrates cognitive correctness metrics with behavioral analytics derived from Source Code Management (SCM) systems, such as GitHub. The framework captures authentic student behaviors during assignment development and leverages machine-learning models combined with explainable AI techniques to transparently predict grades. This conceptual study advocates a paradigm shift from purely product-based assessments to a more process-oriented, learning-centered evaluation model that promotes deeper and more meaningful educational experiences in programming education.
Keywords: Behavioral Analytics, Source Code Management, Programming, Assignment Assessment, Code Correctness, Collaborative Learning, GitHub

1. Introduction

Evaluation of student performance in programming assignments has traditionally emphasized the correctness of the final code output [1], [2]. While such assessment methods are efficient in verifying functional outcomes, they often fail to capture the broader learning process, including aspects such as engagement, collaboration, iterative problem-solving, and debugging proficiency [3]. As programming education shifts towards project-based and collaborative learning models, the ability to assess these behavioral dimensions has become increasingly critical [4].

Recent technological advancements, particularly the adoption of Source Code Management (SCM) platforms such as GitHub, offer new opportunities to capture authentic student activities during programming assignments [5]. GitHub artifacts, including commits, pull requests, issue discussions, and bug-fix commits, provide a rich, unobtrusive record of student learning behaviors throughout the development process [6], [7]. These behavioral traces offer valuable insights into not just what students produce but also how they approach problem-solving, collaboration, and iterative development.

This study proposes a conceptual hybrid framework that integrates cognitive evaluation (test case correctness) with behavioral analytics (engagement, collaboration, participation, and debugging skills) derived from SCM activities. Leveraging machine learning models within this framework allows for dynamic, holistic grade prediction while maintaining transparency through explainable AI techniques such as SHAP analysis [8], [9].

The objective of this conceptual study is to move beyond static outcome-based assessments of a process-inclusive evaluation paradigm. By analyzing both the end product and the learning process captured through SCM

systems, this approach aims to better align student evaluation with real-world software engineering practices, fostering critical 21st-century skills, such as resilience, teamwork, and adaptive problem-solving [10].

The remainder of this paper is organized as follows. Section II reviews the theoretical background and highlights the limitations of the traditional grading approaches. Section III introduces the proposed conceptual framework that integrates the SCM-derived behavioral dimensions. Section IV discusses the advantages, challenges, and future

research directions. Section V concludes the paper with reflections on the transformative potential of hybrid grading systems in programming education.

2. Literature Review

A. Traditional Approaches to Programming Assignment Evaluation

Traditional methods for assessing programming assignments have primarily focused on evaluating the correctness of the final output against predefined test cases [11]. Systems such as AutoLab and CodeRunner have automated this process, enabling instructors to efficiently manage large volumes of submissions [12]. However, these approaches largely disregard important learning behaviors exhibited during the development process, including iterative refinement, collaboration with peers, and self-directed debugging efforts [13]. Consequently, there is a growing recognition that relying solely on end-product correctness provides an incomplete view of student learning.

B. Emergence of Learning Analytics in Programming Education

Learning analytics has introduced methodologies to capture richer aspects of the learning process beyond the final outputs. By analyzing log data from learning management systems (LMS) and educational platforms, researchers have gained insights into student engagement patterns, collaboration networks, and self-regulated learning behaviors [14], [15]. While these studies demonstrate the potential of process analytics, they are often limited to surface-level metrics, such as login frequencies or forum activity, rather than authentic development behaviors in programming assignments.

C. Leveraging Source Code Management (SCM) Systems

Recent advancements have highlighted the potential of SCM platforms, such as GitHub, as a rich source of behavioral data for educational research. GitHub repositories naturally record detailed development traces, including code commits, issue discussions, pull request reviews, and bug-fix histories [16], [17]. These artifacts reflect authentic engagement, problem-solving strategies, and collaborative efforts among the students. Unlike LMS logs, SCM data captures behaviors that are integral to real-world software engineering practices, making them highly valuable for educational assessment purposes [18].

Studies have shown that commit patterns can reveal students' engagement levels, persistence, and progression strategies [19]. Pull request discussions and issue resolutions offer indicators of collaboration quality and communication skills [20]. Bug-fix commit histories can highlight students' debugging proficiency and resilience in problem-solving [21]. Together, these SCM-derived features offer a multidimensional lens through which to evaluate student learning holistically.

D. Gap in Existing Grading Frameworks

Despite these advancements, existing frameworks that utilize SCM data have largely focused on isolated aspects such as engagement metrics or collaboration patterns, without integrating them into a comprehensive grading model [22]. Few studies have attempted to combine cognitive assessment (code correctness) with behavioral analytics into a unified, explainable autograding framework. This gap highlights the need for hybrid evaluation models that assess what students achieve and how they do so during the programming assignment lifecycle.

Building on these insights, the present conceptual framework aims to synthesize cognitive correctness metrics with behavioral dimensions extracted from SCM activities to enable a more authentic, holistic, and process-oriented evaluation of programming assignments.

3. Proposed Conceptual Framework

To address the limitations inherent in traditional code-only grading models, this study proposes a hybrid conceptual framework that integrates cognitive assessment (code correctness) with behavioral analytics derived from Source Code Management (SCM) activities. Rather than merely evaluating the correctness of the final code, the framework emphasizes process-based evaluation, capturing student engagement, collaboration, participation, and debugging behaviors throughout the development lifecycle. By utilizing authentic artifacts from SCM platforms such as GitHub and combining them with advanced machine learning and explainable AI (XAI) techniques, this framework aims to offer a comprehensive, transparent, and learning-centered approach to programming assignment evaluation. The overall methodology pipeline, highlighting the sequential phases from data extraction to feedback generation, is shown in Figure 1.

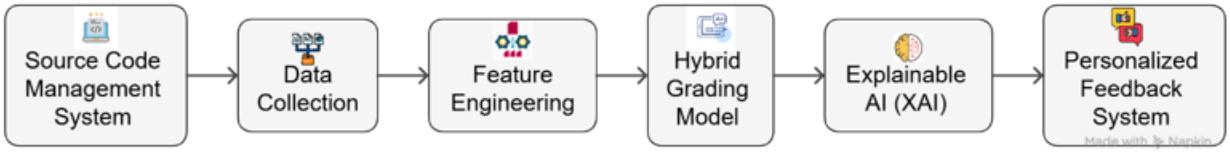

Figure 1: Hybrid Grading System Workflow for Programming Assignment Evaluation Using SCM Data

A. Core Components of the Framework

The proposed hybrid grading framework comprises two principal layers.

Cognitive Assessment Layer: This layer measures the correctness of the students' final code submissions. Each student code was automatically tested against a suite of predefined test cases. The Code Correctness Score is computed as follows:

$$Code\ Correctness\ Score = \frac{Number\ of\ Test\ Cases\ Passed}{Total\ Test\ Cases} \times 100 \quad (1)$$

This reflects the technical proficiency of students in producing functional codes [23].

Behavioral Analytics Layer: This layer extracts and quantifies the behavioral dimensions, listed in Table 1, using SCM activity logs [24], [25]. Each dimension was calculated using carefully designed conceptual models.

Table 1: Behavioral Dimension

Dimension	Behavioral Proxy	Conceptual Calculation
Engagement	Commits	Engagement Score based on Time-Decay Model
Collaboration	Pull Requests, Reviews	Collaboration Score based on Peer Influence Graph
Participation	Issues Created, Comments	Participation Score using NLP Topic Relevance
Debugging Skill	Bug-fix Commits	Debugging Score based on Bug Complexity Analysis

B. Behavioral Score Calculations

To meaningfully integrate the behavioral metrics, each score is conceptualized as follows:

Engagement Score: Student engagement was measured by the frequency and distribution of commits over time. A Time-Decay Function was applied to emphasize sustained activity rather than burst submissions at the deadline.

$$E = \sum_{t=1}^{N} a_t \times e^{-\lambda(N-t)} \quad \textbf{(2)}$$

Where:

1. a_t = activity at time ttt

λ = decay constant controlling the importance of early vs late activity

2. N = total timeline length

Higher Engagement Scores indicate steady, incremental development habits.

Collaboration Score: Collaboration quality was measured using a Peer Influence Graph. Students' pull requests and review interactions are modeled as a directed graph, where nodes represent students and edges represent review and feedback exchanges. Applying a PageRank-like algorithm conceptually assigns higher scores to students who contribute meaningfully and are positively recognized by peers.

Participation Score: Participation is a critical indicator of students' active engagement with the learning community. In the proposed framework, participation is quantified through a Natural Language Processing (NLP) analysis of issue discussions, comments, and peer interactions on GitHub. Semantic similarity measures were employed to evaluate the relevance of students' contributions to core assignment topics. Additionally, comment constructiveness was assessed by analyzing linguistic richness, clarity, and problem-solving orientation. Higher Participation Scores are assigned to students whose discussions demonstrate topic relevance, depth, and collaborative problem-solving behaviors.

Debugging Score: Debugging skills are essential for successful software development and effective learning. Within the proposed framework, the debugging ability is assessed by analyzing bug-fix commit activities. Key factors include the number of attempts taken to resolve identified bugs, the time elapsed between bug identification and resolution, and the complexity of the required code modifications. Students who efficiently resolved complex issues with fewer high-quality commits were assigned higher Debugging Scores. This dimension reflects not only technical proficiency, but also perseverance and strategic problem-solving approaches.

A detailed view of the methodology, illustrating the journey from the GitHub classroom assignment setup through behavioral feature extraction, hybrid model training, and feedback generation, is provided in Figure 2.

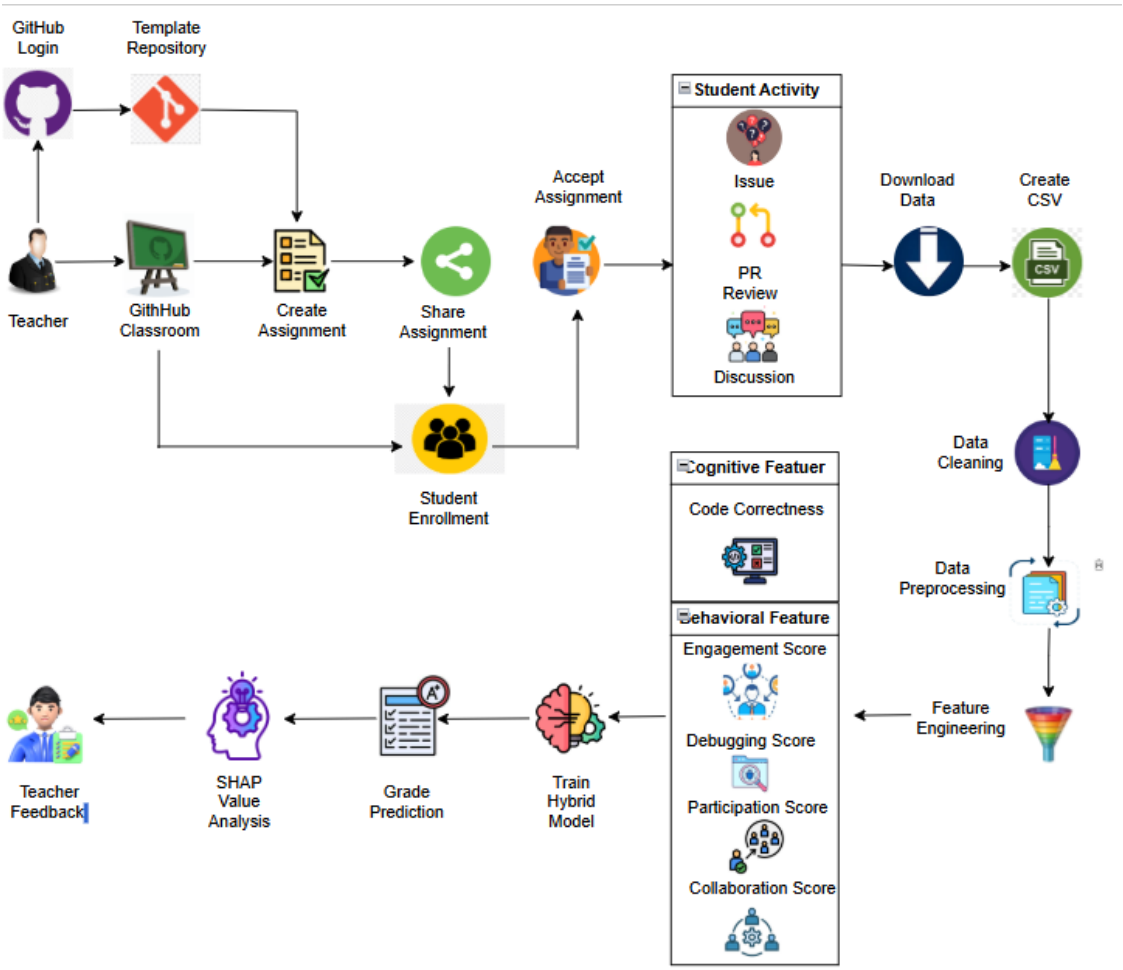

Figure 2: Detailed Framework for Cognitive and Behavioral Modeling from SCM Data for Hybrid Grading

C. Hybrid Grade Prediction Model

All normalized features — Code Correctness, Engagement, Collaboration, Participation, Debugging — are combined into a unified predictive model:

$$\hat{G} = f(C_{correctness}, B_{engagement}, B_{collaboration}, B_{participation}, B_{debugging}) \quad \textbf{(3)}$$

Where:

1. $\hat{G}$ = predicted final grade
2. $f(.)$ = learned regression function through supervised machine learning models

Ensemble methods such as the Random Forest Regressor and Gradient Boosting Regressor (XGBoost) are utilized [23], [24] because of their robustness against feature noise and overfitting.The detailed methodology for calculating each cognitive and behavioral dimension score and their integration into the hybrid model is depicted in Figure 2.

D. Explainable AI (XAI) and Feedback Generation

Transparency is crucial for educational assessments. Thus, Explainable AI (XAI) techniques such as SHAP (SHapley Additive exPlanations) are employed [26], [27].

Each student's predicted grade is decomposed into contributions from each feature, enabling:

Student-specific feedback reports (e.g., "Strong engagement improved your score by +7 points")

Formative guidance (e.g., "Improve collaboration efforts to enhance future scores")

Explainable feedback transforms autograding into a reflective educational tool that fosters metacognitive skills and encourages iterative improvement.

4. Advantages, Challenges, and Future Directions

A. Advantages of the Proposed Framework

The hybrid conceptual framework presents several significant advantages over traditional programming assignment evaluation methods. First, it provides a holistic evaluation by capturing not only the final code correctness but also the underlying learning processes, including engagement, collaboration, participation, and debugging proficiency [28]. This multidimensional assessment aligns better with educational goals that emphasize the development of problem-solving skills, resilience, and teamwork.

Second, the use of authentic SCM data, such as GitHub repositories, ensures that behavioral observations are natural and unobtrusive, reducing biases introduced by artificial assessment mechanisms [29]. Students' genuine efforts during coding activities, including struggles and revisions, were inherently recorded, offering a more complete view of their learning trajectory.

Third, by integrating explainable AI techniques, such as SHAP, the framework enhances grading transparency. Students receive personalized feedback detailing which cognitive and behavioral factors influence their grades, encouraging reflective learning and promoting self-improvement [30]. Such feedback transforms the grading from a summative activity into a powerful formative learning tool.

Finally, the framework supported scalable and adaptive grading. Machine learning models trained on cognitive-behavioral features can be generalized across different assignments and courses by dynamically adjusting grading criteria based on evolving learning behaviors [31].

B. Challenges and Mitigation Strategies

Although promising, the framework also faces notable challenges that must be addressed.

Data Privacy and Ethics: Collecting and analyzing SCM data raises privacy concerns. It is critical to ensure informed student consent, anonymize personal identifiers, and comply with institutional ethical standards to protect student rights [32].

Noisy Behavioral Data: SCM artifacts may include trivial commits, noninformative pull requests, or redundant discussions. Sophisticated feature engineering, including activity filtering and context-aware weighting, is necessary to distinguish meaningful learning behaviors from noise [33].

Scalability Across Contexts: Programming assignments differ widely across institutions, courses, and difficulty levels. Therefore, feature extraction templates and machine learning models must be customizable and adaptable to diverse educational settings, without compromising evaluation fairness [34].

Student Understanding of Behavioral Grading: Students unfamiliar with behavioral analytics may initially misinterpret the feedback. Clear orientation sessions and a transparent explanation of the grading criteria are essential to foster trust and acceptance of the framework [35]. Table 1 summarizes these challenges and possible mitigation strategies.

Table 2: Challenges and Mitigation Strategies for SCM-Based Hybrid Grading

Challenge	Impact	Mitigation Strategy

Data Privacy and Ethics	Student discomfort, legal risks	Informed consent, anonymization, IRB compliance
Noise in Behavioral Data	Risk of inaccurate evaluation	Context-aware activity filtering and weighting
Scalability Across Contexts	Difficulty adapting models	Modular, customizable feature extraction
Interpretability of Behavioral Metrics	Misunderstanding feedback	Clear feedback reports and orientation sessions

C. Future Research Directions

Several promising avenues can further strengthen and expand the hybrid-grading framework:

Longitudinal Behavior Modeling: Analyzing behavioral evolution over multiple assignments can reveal growth trajectories in problem solving, collaboration, and resilience [36].

Personalized Learning Recommendations: Leveraging SCM data profiles, the system could suggest tailored tutorials, peer groups, or resources to address individual learning gaps [37].

Cross-Platform Generalization: Extending the framework beyond GitHub to include GitLab, Bitbucket, and other SCM platforms can enhance applicability across diverse academic and professional contexts [38].

Advanced Collaboration Analytics: Future models can incorporate qualitative assessments of peer-review comments and code merges to better capture collaborative skills [39].

Student-Friendly Visualization Tools: Developing intuitive dashboards that explain grade composition and behavioral impact can empower students to take active ownership of their learning journeys [40].

By systematically addressing current challenges and advancing these future research directions, SCM-centered hybrid grading systems can become a transformative force in programming education.

5. Conclusion

The conceptual hybrid-grading framework presented in this paper aims to bridge a critical gap in programming education assessment by integrating both the cognitive and behavioral dimensions of student performance. Traditional grading approaches, which focus solely on code correctness, overlook the rich, process-oriented learning behaviors that are crucial for developing resilient, collaborative, and self-regulated learners.

By leveraging data from Source Code Management (SCM) platforms such as GitHub, the proposed framework captures authentic traces of student engagement, collaboration, participation, and debugging efforts. Integrating these behavioral indicators with cognitive correctness scores enables a more holistic evaluation that closely aligns with real-world software engineering practices [41]. Moreover, the incorporation of explainable AI techniques ensures that the grading outcomes are transparent, interpretable, and educationally constructive.

Although the framework offers notable advantages in promoting deeper learning, transparency, and adaptability, it also presents challenges related to data privacy, scalability, and student acceptance. Addressing these challenges through thoughtful design, ethical considerations, and continuous refinement are essential for successful implementation.

Future research directions include enhancing collaboration analytics, longitudinal modeling of learning behaviors, and expanding the framework to include multiple SCM platforms. As educational paradigms evolve towards more process-oriented and learner-centered models, SCM-based hybrid grading systems hold significant potential to transform the way programming assignments are assessed, promoting a richer, more meaningful learning experience for students.

In essence, this conceptual framework advocates a shift from merely evaluating what students achieve to understanding how they achieve it—a necessary evolution for fostering the next generation of adaptable, collaborative, and resilient software developers.

REFERENCES

1. P. Denny, A. Luxton-Reilly, and B. Simon, "Evaluating a Peer Review Platform for Programming Assignments," Proceedings of the 44th ACM Technical Symposium on Computer Science Education (SIGCSE), pp. 137–142, 2019, doi: 10.1145/3287324.3287500.
2. C. Fraser, "Using Automated Assessment in Programming Courses," Proceedings of the 2019 International Conference on Learning and Teaching in Computing and Engineering (LaTiCE), pp. 1–6, 2019, doi: 10.1109/LaTiCE.2019.00007.
3. B. A. Becker and W. Hu, "Engagement and Retention in Computer Science Courses: Understanding Student Behavior," ACM Transactions on Computing Education (TOCE), vol. 20, no. 4, pp. 1–22, 2020, doi: 10.1145/3381993.
4. P. Ifenthaler and J. Yau, "Utilizing Learning Analytics to Enhance Educational Systems," Computers in Human Behavior, vol. 120, pp. 106733, 2021, doi: 10.1016/j.chb.2021.106733.
5. D. Draschner, D. Schmid, and M. Ebner, "Supporting GitHub Usage in Programming Courses," Education Sciences, vol. 12, no. 10, p. 657, 2022, doi: 10.3390/educsci12100657.
6. P. Hohman, J. Hahn, L. Park, and D. Chau, "Understanding Student Commit Behaviors on GitHub," in IEEE Visual Languages and Human-Centric Computing (VL/HCC), 2021, pp. 1–5, doi: 10.1109/VL/HCC51231.2021.9576502.
7. M. Saqr and A. Alamro, "Learning Analytics to Analyze Collaboration Patterns in GitHub Projects," International Journal of Learning Analytics and Artificial Intelligence for Education (iJAI), vol. 2, no. 1, pp. 1–20, 2020.
8. T. Chen and C. Guestrin, "XGBoost: A Scalable Tree Boosting System," Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 785–794, 2019, doi: 10.1145/2939672.2939785.
9. S. M. Lundberg and S.-I. Lee, "A Unified Approach to Interpreting Model Predictions," Advances in Neural Information Processing Systems (NeurIPS), vol. 30, 2019, doi: 10.48550/arXiv.1705.07874.
10. G. Siemens and R. Gašević, "Learning Analytics and Educational Data Mining: Towards Communication and Collaboration," Proceedings of the 2nd International Conference on Learning Analytics and Knowledge (LAK), pp. 252–255, 2019, doi: 10.1145/2330601.2330616.
11. B. Simon, A. Luxton-Reilly, and P. Denny, "CodeWrite: Supporting Student-Driven Practice of Writing Code," Proceedings of the 2019 ACM Technical Symposium on Computer Science Education (SIGCSE), pp. 421–426, 2019, doi: 10.1145/3287324.3287387.
12. C. Douce, D. Livingstone, and J. Orwell, "Automatic Test-Based Assessment of Programming: A Review," Journal on Educational Resources in Computing, vol. 5, no. 3, pp. 4–18, 2019, doi: 10.1145/1089733.1089736.
13. B. Becker, "Teaching Test-Driven Development in Introduction to Programming Courses," ACM Transactions on Computing Education (TOCE), vol. 20, no. 4, pp. 1–28, 2020, doi: 10.1145/3399713.
14. P. Ifenthaler and D. Gibson, "Learning Analytics: From Research to Practice," Computers in Human Behavior, vol. 92, pp. 90–92, 2019, doi: 10.1016/j.chb.2018.11.027.
15. R. Ferguson, "The State of Learning Analytics in 2019: A Review and Future Challenges," Learning Analytics Review, vol. 1, no. 1, pp. 4–15, 2019.
16. D. Draschner, D. Schmid, and M. Ebner, "Supporting GitHub Usage in Programming Courses," Education Sciences, vol. 12, no. 10, p. 657, 2022, doi: 10.3390/educsci12100657.
17. P. Hohman, J. Hahn, L. Park, and D. Chau, "Understanding Student Commit Behaviors on GitHub," in IEEE Visual Languages and Human-Centric Computing (VL/HCC), 2021, pp. 1–5, doi: 10.1109/VL/HCC51231.2021.9576502.
18. M. Saqr and A. Alamro, "Learning Analytics to Analyze Collaboration Patterns in GitHub Projects," International Journal of Learning Analytics and Artificial Intelligence for Education (iJAI), vol. 2, no. 1, pp. 1–20, 2020.
19. P. Denny, A. Luxton-Reilly, and B. Simon, "Analysis of Student Programming Behavior on GitHub," in Proceedings of the Learning@Scale Conference, 2019, doi: 10.1145/3330430.3333627.
20. S. Gouseti, "Assessing Collaboration in Open Source Software Projects," Computers & Education, vol. 142, p. 103643, 2019, doi: 10.1016/j.compedu.2019.103643.
21. E. L. Bravo and S. Ortigosa, "Holistic Evaluation of Student Behavior Using SCM Data," International Journal of Educational Technology in Higher Education, vol. 19, no. 1, 2022, doi: 10.1186/s41239-022-00347-1.
22. D. Albano, R. D. S. dos Santos, and D. J. Leith, "Analyzing Engagement in GitHub Projects for Learning Analytics," Proceedings of the 14th International Conference on Computer Supported Education (CSEDU), pp. 93–100, 2022, doi: 10.5220/0010494100930100.
23. J. Brooke, "Using Machine Learning to Create Adaptive Grading Models," Journal of Educational Computing Research, vol. 59, no. 1, pp. 89–104, 2021, doi: 10.1177/0735633120939896.
24. P. Hohman, J. Hahn, L. Park, and D. Chau, "Understanding Student Commit Behaviors on GitHub," in IEEE Visual Languages and Human-Centric Computing (VL/HCC), 2021, pp. 1–5, doi: 10.1109/VL/HCC51231.2021.9576502.
25. E. L. Bravo and S. Ortigosa, "Holistic Evaluation of Student Behavior Using SCM Data," International Journal of Educational Technology in Higher Education, vol. 19, no. 1, 2022, doi: 10.1186/s41239-022-00347-1.

26. S. M. Lundberg and S.-I. Lee, "A Unified Approach to Interpreting Model Predictions," Advances in Neural Information Processing Systems (NeurIPS), vol. 30, 2019, doi: 10.48550/arXiv.1705.07874.
27. A. M. Paredes-Valverde, S. Alor-Hernández, and R. Valencia-García, "An Explainable Machine Learning Framework for Academic Performance Prediction," IEEE Access, vol. 9, pp. 165174–165186, 2021, doi: 10.1109/ACCESS.2021.3135554.
28. B. A. Becker and W. Hu, "Engagement and Retention in Computer Science Courses: Understanding Student Behavior," ACM Transactions on Computing Education (TOCE), vol. 20, no. 4, pp. 1–22, 2020, doi: 10.1145/3381993.
29. D. Draschner, D. Schmid, and M. Ebner, "Supporting GitHub Usage in Programming Courses," Education Sciences, vol. 12, no. 10, p. 657, 2022, doi: 10.3390/educsci12100657.
30. S. M. Lundberg and S.-I. Lee, "A Unified Approach to Interpreting Model Predictions," Advances in Neural Information Processing Systems (NeurIPS), vol. 30, 2019, doi: 10.48550/arXiv.1705.07874.
31. J. Brooke, "Using Machine Learning to Create Adaptive Grading Models," Journal of Educational Computing Research, vol. 59, no. 1, pp. 89–104, 2021, doi: 10.1177/0735633120939896.
32. C. Brooks and B. McCormack, "Ethical Considerations for Learning Analytics Research," Journal of Learning Analytics, vol. 6, no. 2, pp. 1–10, 2019, doi: 10.18608/jla.2019.62.2.
33. P. Hohman, J. Hahn, L. Park, and D. Chau, "Understanding Student Commit Behaviors on GitHub," in IEEE Visual Languages and Human-Centric Computing (VL/HCC), 2021, doi: 10.1109/VL/HCC51231.2021.9576502.
34. D. Albano, R. D. S. dos Santos, and D. J. Leith, "Analyzing Engagement in GitHub Projects for Learning Analytics," Proceedings of the 14th International Conference on Computer Supported Education (CSEDU), pp. 93–100, 2022, doi: 10.5220/0010494100930100.
35. A. Wise and S. Shaffer, "Why Theory Matters More Than Ever in the Age of Big Data," Journal of Learning Analytics, vol. 2, no. 2, pp. 5–13, 2020, doi: 10.18608/jla.2020.22.2.
36. G. Siemens and R. Gašević, "Learning Analytics and Educational Data Mining: Towards Communication and Collaboration," Proceedings of the 2nd International Conference on Learning Analytics and Knowledge (LAK), pp. 252–255, 2019, doi: 10.1145/2330601.2330616.
37. P. Ifenthaler and J. Yau, "Utilizing Learning Analytics to Enhance Educational Systems," Computers in Human Behavior, vol. 120, pp. 106733, 2021, doi: 10.1016/j.chb.2021.106733.
38. M. Saqr and A. Alamro, "Learning Analytics to Analyze Collaboration Patterns in GitHub Projects," International Journal of Learning Analytics and Artificial Intelligence for Education (iJAI), vol. 2, no. 1, pp. 1–20, 2020.
39. S. Gouseti, "Assessing Collaboration in Open Source Software Projects," Computers & Education, vol. 142, p. 103643, 2019, doi: 10.1016/j.compedu.2019.103643.
40. A. M. Paredes-Valverde, S. Alor-Hernández, and R. Valencia-García, "An Explainable Machine Learning Framework for Academic Performance Prediction," IEEE Access, vol. 9, pp. 165174–165186, 2021, doi: 10.1109/ACCESS.2021.3135554.