Asian And Pacific Economic Review
ISSN: 1000-6052

IMPACT OF AI-BASED AUTO-GRADING SYSTEMS ON STUDENT LEARNING CONSISTENCY IN PROGRAMMING COURSES

Mr. Ajay Kumar Gupta

Assistant Professor, Department of Computer Science, Lucknow Public College of Professional Studies, Lucknow, Uttar Pradesh

https://doie.org/10.65985/APER.2026805538

ABSTRACT

The integration of AI-based auto-grading systems in programming education has significantly transformed assessment practices in higher education. While prior research emphasizes efficiency gains and instructor workload reduction, limited empirical evidence evaluates how auto-grading affects student learning dynamics across time. This study examines whether AI-based auto-grading systems improve learning consistency, submission punctuality, and performance stability among undergraduate programming students. Using longitudinal data from 240 students enrolled in two parallel introductory programming course sections, one utilizing AI-based auto-grading and the other relying on traditional manual grading, the study analyzes semester-wide grade variance, assignment punctuality, and performance trajectory trends across eight structured programming assignments. Results indicate that students exposed to AI-based auto-grading demonstrate significantly lower grade volatility and higher punctual submission rates. Moreover, longitudinal regression modeling shows smoother upward learning trajectories among students receiving immediate AI feedback. However, excessive resubmission frequency slightly moderates trajectory improvement, suggesting potential optimization behavior rather than reflective learning in some cases. The findings suggest that AI-based auto-grading systems function primarily as feedback stabilization mechanisms that enhance learning consistency rather than merely increasing average grades. The study contributes to understanding AI-enabled assessment systems in computer science education and offers policy insights for balanced AI integration in formative evaluation practices.

INTRODUCTION

Assessment remains one of the most influential mechanisms shaping student learning behavior in programming education. Unlike theoretical disciplines, programming courses rely heavily on iterative practice, debugging cycles, and feedback-driven correction. Traditionally, assignment evaluation has depended on manual grading processes, which often involve delayed feedback and subjective variation in assessment standards. As enrollment sizes increase and instructional resources remain limited, many institutions have adopted AI-based auto-grading systems to streamline evaluation.

AI-based auto-grading systems automatically compile code, test submissions against predefined cases, detect logical errors, and generate immediate performance feedback. These systems promise scalability, objectivity, and rapid turnaround. However, the pedagogical implications of such automation remain insufficiently examined. While efficiency and standardization are clear benefits, less is known about how continuous automated feedback influences student learning patterns across time. Most existing studies focus on average grade improvements or student

satisfaction. Such measures capture static outcomes rather than dynamic learning behavior. Programming mastery develops cumulatively across assignments, making performance stability and trajectory progression more meaningful indicators of learning quality. Large fluctuations in assignment scores may signal unstable understanding, while smoother progression suggests incremental skill consolidation.

This study introduces the concept of **learning consistency** within programming education. Learning consistency refers to reduced grade volatility, timely engagement with assignments, and progressive improvement trends across the semester. Rather than asking whether AI-based auto-grading increases final grades, the research evaluates whether it stabilizes learning processes. The central argument is that immediate automated feedback may function as a corrective reinforcement loop. By allowing students to detect and fix errors quickly, auto-grading systems may prevent the accumulation of misconceptions that often occur under delayed manual grading systems. At the same time, unlimited resubmission features may encourage strategic optimization behavior focused on passing test cases rather than strengthening conceptual reasoning.

Using longitudinal assignment data from two parallel programming sections, this study empirically examines whether AI-based auto-grading reduces grade variance, increases submission punctuality, and promotes smoother learning trajectories. By shifting focus from static achievement to performance stability, the study provides a more nuanced understanding of AI-enabled assessment systems in computer science education.

LITERATURE REVIEW

Feedback is widely recognized as one of the most powerful determinants of learning effectiveness. Educational psychology literature consistently emphasizes that timely, specific, and actionable feedback enhances student performance and reduces error persistence. In programming education, feedback plays an especially critical role because learning often occurs through iterative debugging and refinement. Delayed correction may allow misconceptions to solidify, whereas immediate responses may facilitate rapid conceptual adjustment. Auto-grading systems emerged initially to address scalability challenges in large programming courses. Early systems focused primarily on automated test-case execution and syntactic validation. More advanced AI-based auto-grading tools now incorporate logical analysis, plagiarism detection, performance metrics, and structured feedback generation.

Research on these systems has predominantly examined instructor benefits, including reduced grading workload, improved evaluation consistency, and faster turnaround times. From a student-centered perspective, formative assessment theory suggests that continuous feedback cycles improve learning outcomes by promoting self-regulation and reflective correction. Immediate automated evaluation may shorten the feedback loop, enabling students to adjust their solutions during active cognitive engagement. Such reinforcement aligns with behaviorist learning principles, where rapid feedback strengthens correct responses and discourages repeated mistakes.

However, automation also raises concerns. When students are allowed multiple resubmissions, some may adopt a trial-and-error approach, iteratively modifying code until it passes test cases

without fully understanding the underlying logic. This phenomenon resembles optimization behavior observed in standardized testing contexts, where learners focus on performance metrics rather than conceptual mastery.

Thus, while auto-grading may improve efficiency, its effect on deeper learning processes remains ambiguous. Existing empirical studies often measure average assignment scores or course completion rates. These static indicators provide limited insight into how performance evolves across time. Programming competence develops sequentially, and instability in assignment outcomes may indicate inconsistent conceptual grasp. Therefore, evaluating grade variance and trajectory progression offers a more dynamic perspective on educational impact. Despite widespread institutional adoption of AI-based grading tools, limited research has examined longitudinal performance stability under automated assessment conditions. This study extends prior work by shifting analytical focus toward learning consistency, submission behavior, and performance volatility rather than isolated achievement metrics.

HYPOTHESES DEVELOPMENT

The theoretical foundation of this study rests on feedback reinforcement theory and self-regulated learning frameworks. Immediate feedback reduces uncertainty and allows learners to adjust strategies in real time. In programming contexts, where error correction is central to skill development, timely system-generated feedback may prevent accumulation of misunderstandings.

Under manual grading systems, delays between submission and feedback can create discontinuity in learning. Students may proceed to subsequent assignments without fully correcting prior conceptual errors. AI-based auto-grading compresses this delay, potentially stabilizing learning progression.

H1: Students exposed to AI-based auto-grading will exhibit significantly lower grade variance across assignments compared to students under manual grading systems.

Submission punctuality reflects engagement and behavioral consistency. Automated platforms often provide structured deadlines, instant confirmation, and iterative testing environments, which may encourage timely submission.

H2: AI-based auto-grading is positively associated with higher assignment submission punctuality.

Learning progression can be modeled through performance trajectory slopes across sequential assignments. Immediate correction may promote smoother incremental improvement rather than erratic fluctuations.

H3: Students using AI-based auto-grading will demonstrate more stable upward performance trajectories across the semester.

However, unlimited resubmission features may alter student behavior. High-frequency resubmission may reflect optimization for test-case success rather than reflective understanding.

H4: Excessive resubmission attempts moderate the positive relationship between auto-grading and learning consistency.

These hypotheses collectively examine whether AI-based auto-grading systems function as stabilizing educational mechanisms or merely as performance accelerators.

METHODOLOGY

Research Design and Sample

The study adopts a longitudinal quasi-experimental design using semester-level data from two parallel sections of an introductory programming course at a university. A total of 240 undergraduate computer science students participated. Section A (n = 120) used an AI-based auto-grading system integrated into the programming platform, while Section B (n = 120) followed a traditional manual grading model with instructor-evaluated feedback provided within one week of submission.

Both sections were taught using identical syllabi, instructional materials, assignment structures, and evaluation weightage. The only structural difference was the grading mechanism. The semester included eight programming assignments of increasing complexity.

Variables

Independent Variable

Auto-Grading Usage (binary):
1 = AI-based auto-grading
0 = Manual grading

Dependent Variables

1. **Grade Variance**
   Standard deviation of each student's assignment scores across eight submissions. Lower values indicate greater learning stability.

2. **Submission Punctuality**
   Percentage of assignments submitted before the official deadline.

3. **Learning Trajectory Slope**
   Linear regression slope calculated using assignment scores over time for each student. Higher positive slope indicates consistent improvement.

Moderator

Average Resubmission Attempts per Assignment (continuous variable).

Analytical Approach

Independent sample ***t-tests*** were used to compare mean differences between groups. Multiple regression analysis was conducted to examine the relationship between auto-grading and dependent variables while incorporating the resubmission moderator. All tests were performed at a 5% significance level. Effect sizes were calculated to evaluate practical significance.

RESULTS

Descriptive Findings

Students in the auto-grading section demonstrated lower average grade variance (Mean SD = 6.3) compared to the manual grading section (Mean SD = 9.1). This indicates reduced performance fluctuation across assignments.

Submission punctuality was higher among auto-grading students (Mean = 91.5%) relative to manual grading students (Mean = 82.4%). Additionally, average resubmission attempts per assignment were 2.4 in the auto-grading section compared to 1.1 in the manual section.

Learning trajectory analysis revealed that auto-grading students exhibited a stronger positive slope (Mean slope = +2.1) compared to manual grading students (Mean slope = +1.2).

Hypothesis Testing

H1: Grade Variance

Independent t-test results indicate a statistically significant reduction in grade variance under auto-grading conditions ($t = -4.87$, $p < 0.001$). Effect size (Cohen's $d = 0.61$) suggests a moderate impact. H1 is supported.

H2: Submission Punctuality

The difference in punctuality rates is statistically significant ($t = 5.12$, $p < 0.001$), with a moderate effect size ($d = 0.66$). H2 is supported.

H3: Learning Trajectory

Regression analysis indicates that auto-grading positively predicts trajectory slope ($\beta = 0.84$, $p = 0.002$). The model explains 28 percent of variance ($R^2 = 0.28$). H3 is supported.

H4: Moderation Effect

Interaction analysis between auto-grading and resubmission attempts reveals a small but significant moderation effect ($\beta = -0.27$, $p = 0.038$). High resubmission frequency slightly weakens trajectory smoothness. H4 is partially supported.

The findings indicate that AI-based auto-grading systems reduce performance volatility, increase timely engagement, and promote smoother improvement trends. However, excessive iterative resubmission may introduce strategic optimization behavior that moderates learning progression.

DISCUSSION

This study examined whether AI-based auto-grading systems enhance learning consistency in undergraduate programming education. The findings provide structured evidence that automated grading does more than accelerate evaluation processes; it appears to stabilize learning trajectories across time. The significant reduction in grade variance among students exposed to auto-grading suggests improved performance stability. Lower volatility implies fewer extreme score fluctuations across assignments, indicating that students were able to correct mistakes earlier and avoid repeated conceptual errors. This supports the theoretical argument that immediate feedback functions as a reinforcement mechanism that strengthens incremental learning. Submission punctuality results further reinforce this interpretation. Students using auto-grading systems submitted assignments more consistently before deadlines. Immediate compilation checks and structured submission interfaces may reduce procrastination and uncertainty.

In contrast, manual grading systems often create feedback gaps that delay corrective engagement. The trajectory analysis offers additional insight. Students in the auto-grading group demonstrated smoother upward improvement patterns across assignments. Rather than oscillating between high and low performance, these students progressed more gradually. This pattern aligns with formative assessment theory, which emphasizes continuous feedback cycles as drivers of steady skill acquisition. However, the moderation effect of resubmission frequency introduces nuance. While iterative correction is pedagogically beneficial, excessive resubmission may signal optimization behavior focused on satisfying automated test cases rather than internalizing conceptual understanding. The negative moderation effect, although small, suggests that unlimited iteration without reflection may slightly reduce learning depth gains.

Importantly, the results do not indicate that auto-grading inflates grades artificially. Instead, the evidence suggests stabilization and structured progression. The impact is moderate rather than extreme, reinforcing the credibility of the findings. Overall, the study positions AI-based auto-grading as a feedback stabilization tool that enhances learning consistency while requiring balanced resubmission policies to prevent strategic over-optimization. The findings have direct implications for academic institutions integrating AI-based auto-grading systems into programming curricula.

First, auto-grading systems should be adopted primarily as formative learning tools rather than purely administrative grading mechanisms. Their greatest benefit appears in reducing volatility and promoting stable skill progression. Second, institutions should implement structured resubmission limits. While allowing iterative correction encourages engagement, unlimited attempts without reflective prompts may encourage superficial optimization behavior. A capped resubmission policy or mandatory reflection after a certain number of attempts may preserve learning depth. Third, auto-grading platforms should incorporate explanatory feedback rather than binary pass/fail outputs. Structured hints and reasoning prompts may enhance conceptual reinforcement and mitigate overreliance on trial-and-error strategies. Fourth, hybrid grading models may offer optimal balance. Combining AI-based preliminary grading with instructor commentary for selected assignments can integrate scalability with conceptual depth.

Finally, institutional AI governance policies should focus on transparency and pedagogical alignment. Rather than framing automation as a threat to academic integrity, policies should emphasize structured use aligned with learning objectives. The results suggest that AI-based auto-grading systems, when properly designed and regulated, can enhance learning stability without compromising educational rigor.

CONCLUSION

The rapid expansion of AI-based auto-grading systems in programming education has shifted assessment practices toward automation, scalability, and immediacy. While institutional adoption continues to accelerate, empirical clarity regarding their pedagogical consequences remains limited. This study contributes to that gap by examining how AI-enabled grading influences learning consistency across time rather than merely affecting static achievement metrics. Using longitudinal assignment data from two parallel course sections, the findings demonstrate that AI-based auto-grading significantly reduces grade volatility, increases submission punctuality, and promotes smoother upward learning trajectories. These results suggest that immediate, structured feedback operates as a stabilizing mechanism in programming education. By compressing feedback loops, auto-grading systems appear to prevent the accumulation of conceptual errors that often emerge under delayed manual evaluation systems.

Importantly, the results indicate moderate effects rather than dramatic transformations. Auto-grading does not artificially inflate performance, nor does it fundamentally alter overall achievement distribution. Instead, it enhances stability and structured progression. This distinction is crucial. Stability reflects consolidation of skill acquisition rather than superficial grade enhancement. The moderation analysis provides further nuance. While iterative resubmission supports corrective engagement, excessive resubmission frequency slightly weakens trajectory improvement. This suggests that unregulated iteration may encourage test-case optimization behavior rather than reflective conceptual processing. However, the moderation effect remains modest, indicating that appropriate policy design can mitigate such risks.

The study advances the literature by introducing learning consistency as a meaningful evaluation metric in programming education research. Traditional research often focuses on mean score differences or completion rates. Yet programming mastery develops cumulatively. Volatility and instability in performance may reveal deeper learning gaps that average grades conceal. By analyzing grade variance and trajectory slopes, this study provides a more dynamic understanding of educational impact.

Several limitations warrant consideration. The study was conducted within a single course context. Broader replication across institutions, programming levels, and curricular structures would strengthen generalizability. Additionally, the analysis did not directly measure conceptual depth through qualitative assessments, which may provide complementary insight. Future research may explore hybrid grading models, adaptive AI feedback designs, and long-term retention outcomes. Investigating how AI-generated feedback quality influences learning consistency would further refine understanding of effective implementation strategies.

In conclusion, AI-based auto-grading systems appear to enhance learning stability in programming courses by reinforcing timely correction and structured engagement. Their effectiveness depends not solely on technological capability but on thoughtful pedagogical alignment. When integrated with balanced resubmission policies and reflective prompts, auto-grading systems can function as constructive reinforcement mechanisms within modern computer science education.

REFERENCES

- Anderson, J. R. (1996). ACT: A simple theory of complex cognition. *American Psychologist, 51*(4), 355–365.
- Biggs, J. (1998). Assessment and classroom learning. *Assessment in Education, 5*(1), 103–110.
- Black, P., & Wiliam, D. (1998). Assessment and classroom learning. *Assessment in Education, 5*(1), 7–74.
- Boud, D., & Falchikov, N. (2007). *Rethinking assessment in higher education*. Routledge.
- Carless, D. (2006). Differing perceptions in feedback process. *Studies in Higher Education, 31*(2), 219–233.
- Davis, F. D. (1989). Perceived usefulness and user acceptance. *MIS Quarterly, 13*(3), 319–340.
- Ericsson, K. A. (2006). The influence of experience and deliberate practice. *Cambridge Handbook of Expertise*.
- Gikandi, J., Morrow, D., & Davis, N. (2011). Online formative assessment. *Computers & Education, 57*(4), 2333–2351.
- Guzdial, M. (2015). *Learner-centered design of computing education*. Morgan & Claypool.
- Hattie, J., & Timperley, H. (2007). The power of feedback. *Review of Educational Research, 77*(1), 81–112.
- Ihantola, P., et al. (2010). Review of automatic assessment systems. *Proceedings of Koli Calling*.
- Jordan, S., & Mitchell, T. (2009). e-Assessment for learning? *British Journal of Educational Technology, 40*(2), 371–389.
- Keuning, H., Jeuring, J., & Heeren, B. (2018). Automated feedback in programming education. *ACM Transactions on Computing Education, 18*(3), 1–24.
- Kirschner, P., Sweller, J., & Clark, R. (2006). Why minimal guidance does not work. *Educational Psychologist, 41*(2), 75–86.

- Kulhavy, R. (1977). Feedback in written instruction. *Review of Educational Research, 47*(1), 211–232.
- Linn, M., & Clancy, M. (1992). The case for case studies in programming instruction. *Communications of the ACM, 35*(3), 121–132.
- Nicol, D., & Macfarlane-Dick, D. (2006). Formative assessment and self-regulated learning. *Studies in Higher Education, 31*(2), 199–218.
- Pachler, N., Daly, C., Mor, Y., & Mellar, H. (2010). Formative assessment in digital environments. *Computers & Education, 54*(3), 715–724.
- Pane, J. F., & Myers, B. A. (1996). Usability issues in programming systems. *IBM Systems Journal, 35*(3), 509–532.
- Rogers, E. M. (2003). *Diffusion of innovations* (5th ed.). Free Press.
- Shute, V. (2008). Focus on formative feedback. *Review of Educational Research, 78*(1), 153–189.
- Tempelaar, D., et al. (2012). Formative assessment analytics. *Computers & Education, 58*(1), 263–272.
- Topping, K. (1998). Peer assessment between students. *Review of Educational Research, 68*(3), 249–276.
- Venkatesh, V., Morris, M., Davis, G., & Davis, F. (2003). User acceptance of IT. *MIS Quarterly, 27*(3), 425–478.
- Wang, Y., Li, X., & Su, H. (2020). AI in educational assessment. *IEEE Access, 8*, 156-168.
- Wiliam, D. (2011). *Embedded formative assessment*. Solution Tree Press.
- Wing, J. M. (2006). Computational thinking. *Communications of the ACM, 49*(3), 33–35.
- Zhang, P., & Li, N. (2005). Affective quality and cognitive absorption. *Communications of the ACM, 48*(9), 105–109.

APPENDIX

Appendix A: Descriptive Statistics

Variable	Auto-Grading (n=120)	Manual (n=120)
Grade Variance (Mean SD)	6.3	9.1
Submission Punctuality (%)	91.5	82.4
Learning Slope	2.1	1.2
Avg. Resubmissions	2.4	1.1

Appendix B: t-test Results

Variable	t-value	p-value	Cohen's d
Grade Variance	-4.87	<0.001	0.61
Punctuality	5.12	<0.001	0.66
Trajectory Slope	3.41	0.001	0.44

Appendix C: Regression Model

Dependent Variable: Learning Trajectory Slope

$R^2 = 0.28$

Predictor	Beta	p-value
Auto-Grading	0.84	0.002
Resubmissions	-0.31	0.041
Auto × Resub	-0.27	0.038