Proceedings of CONF-MLA 2026 Symposium: Intelligent Systems and Automation: AI Models, IoT, and Robotic Algorithms
DOI: 10.54254/2755-2721/2026.LD38058

Applications of Educational Data Mining in Identifying Learning Difficulties among Novice Programmers

Chuyang Meng

School of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu, China
mcy.325@outlook.com

Abstract. This narrative review examines how educational data mining (EDM) can inform instructional decisions about learning difficulties among novice programmers. This paper draws on a focused set of publications, including empirical and application-oriented studies. The analysis focuses on target alignment, timing, and granularity. Submission histories and compiler feedback can reveal learning processes that are not reflected in final grades. Repeated attempts may indicate either progress or ongoing difficulties. Grouping learners and sequence analysis help identify behavioral patterns. Supervised prediction is used to estimate outcomes, while cognitive diagnosis assesses knowledge mastery. Early records may help prioritize students for review. Concept-focused support requires more detailed evidence and checks against actual assignments. Differences in targets, observation windows, and validation designs limit comparisons across studies. Results based on complete learning histories do not establish early prediction capability. Model explanations also do not establish intervention effects. The proposed framework remains empirically unvalidated. Evaluation should address measurement quality, generalizability, predictive error, privacy, and fairness. The effects of subsequent instructional support on learning require separate assessment.

Keywords: student performance prediction, learning difficulty, programming education, educational data mining, cognitive diagnosis

1. Introduction

Programming difficulties are not only about syntax: prior knowledge, problem-solving habits, and instructional design also affect learning progress [1, 2]. In a multinational assessment, many first-year students were still unable to write a working program after the end of an introductory course [3]. In this article, the term "learning difficulties" refers to observed obstacles in the learning process, such as repeated errors, ineffective debugging, unsuccessful submissions, or low participation, rather than to a clinical diagnosis.

Educational Data Mining (EDM) supports student modeling and instructional decisions [4]. Common prediction methods are decision trees, naive Bayes, random forest (RF) models, support vector machines, and boosting methods [5]. Risk prediction does not explain causes; intervention

© 2026 The Authors. This is an open access article distributed under the terms of the Creative Commons Attribution License 4.0 (https://creativecommons.org/licenses/by/4.0/).

DOI: 10.54254/2755-2721/2026.LD38058

effects need to be evaluated separately [6]. The use of demographic features as predictors needs to be justified and assessed in terms of fairness [7].

Applications of EDM in programming education extend from predicting course outcomes to examining how students approach and revise their work. These applications raise a practical question: what can teachers do with the resulting evidence? A prediction may identify a student for review without explaining the difficulty or indicating an appropriate response. This review therefore focuses on actionable instructional decisions. It considers how the target, timing, and detail of an analytical output affect its usefulness for selecting students for review or planning concept-focused support. The aim is to clarify what different methods can contribute to these decisions and where additional evidence from assignments or learning records is still needed.

Section 2 discusses the studies included in the review. Sections 3 and 4 consider the suitability of analytical results for particular instructional decisions in terms of target, timing, and granularity.

Four questions frame this review:

RQ1: Which forms of static and process data can serve as indicators of learning difficulty among novice programmers?

RQ2: How are learner grouping, sequence mining, supervised prediction, and cognitive diagnosis applied to identify programming difficulty?

RQ3: Where do these methods diverge in predictive performance, interpretability, and application scenarios?

RQ4: What limitations persist regarding generalizability, instructional application, privacy, and fairness?

These questions are addressed through a structured synthesis of empirical and conceptual work.

2. Literature review

2.1. Review methodology

This paper takes a focused narrative review approach with a structured synthesis. Most of the papers reviewed were published between January 1, 2020 and August 24, 2026, with thirteen publications falling into this period. Two earlier studies from 2001 and 2003 were kept as key background sources on novice programming. The review materials were collected in stages: ten full-text articles were chosen first, and then five more were added to strengthen the coverage of foundational, methodological, and recent work. Papers were included based on their relevance to programming education, their contribution to EDM-related analysis or the review's conceptual basis, and whether there was enough information to assess their methods or findings. By study type, ten publications are empirical or application-oriented studies and five are reviews or conceptual discussions. The empirical or application studies cover programming performance, learning behavior, knowledge diagnosis, and instructional support. The other publications provide background and methodological context. The bibliographic information and reported findings of all fifteen publications have been checked against the original texts as well as publisher or DOI records. The early programming assessments provide background on novice learning difficulties rather than EDM prediction results.

For each study, the context, sample, input data, analytical methods, and assessed outcomes are compared. Observation windows, validation designs, and reported metrics are also considered. These details determine how the findings can be interpreted in the classroom context. They also form the basis for the methodological recommendations in Section 3, although these recommendations have not yet been tested as an intervention framework.

DOI: 10.54254/2755-2721/2026.LD38058

This review uses a descriptive synthesis without statistical pooling. The exact dates and complete result counts of the original searches were not retained. A complete search and screening log is unavailable. The review therefore does not claim exhaustive coverage, and its selection process cannot be fully reproduced.

2.2. Data and analytical approaches

Earlier grades, demographic information, and prior learning experience describe students' starting points. Submission records, errors, timestamps, numbers of attempts, code revisions, and resource usage describe their subsequent learning. Numerical data have to be interpreted in conjunction with correctness, error types, and code changes. Higher-performing groups made more frequent use of practice-related materials [8]. Submission metadata allowed the researchers to predict the outcome, even without access to students' source code or demographic information [9]. A follow-up course analysis found that higher-performing students generally took longer to submit their programming assignments and made more attempts on average [10]. Incorrect submissions could sometimes be accompanied by active code improvements [11].

The four approaches address different questions and can complement each other; their accuracy metrics cannot be used for a unified ranking. Learner grouping does not require learning difficulty labels because it can identify, for instance, behavioral similarities. Network analysis and community detection were used to identify learner groups [8]. Mixture model profiles help explain the predictions [11]. Action-sequence patterns were extracted from the Snap logs of 106 students for AdaBoost classification [12]. Students were divided into higher-performing and lower-performing groups based on the median student grade. These labels may reflect students' support needs, but they do not define learning difficulty. Supervised prediction concerns a defined outcome. Temporal validation and privacy-aware federated prediction address different deployment needs [13, 14]. P-FuzzyCDF was evaluated for concept mastery on four real programming datasets [15].

3. Methodological framework

Evidence needs are related to data and method choice. No new model was trained. Teaching context shapes both the decisions available to a teacher and the evidence required for those decisions. This review draws on this relationship.

3.1. Teaching context and feasible decisions

Feasible responses and deadlines are determined by course stage, support resources, platform records, assessment schedules, and student experience. If support resources are limited, a teacher may have to decide whom to review first; reviewing a concept requires finer-grained evidence. Collaborative projects may call for group feedback; decisions at the end of the course concern achieved outcomes. Each setting places different demands on the target, time window, and level of detail.

3.2. Evidence matched to the decision

Three criteria are used to judge whether an analytical output is suitable for the intended teaching decision. The first is target alignment, that is, whether the output matches the teacher's decision question; high accuracy on a different target does not establish suitability.

DOI: 10.54254/2755-2721/2026.LD38058

For timing alignment, both inputs and outputs have to be available before an action is needed. Information that is available only at the end of the course cannot support earlier decision-making. Early records can support predictions of final grades before instructional action is required.

Granularity matching concerns a seemingly simple question: how much detail is needed for effective action? Student-level risk assessment may help identify which students should be prioritized for review, but identifying which concepts should be revisited would require more fine-grained evidence. More detail is not always better. Even when considered together, these three criteria, which have not yet been validated, cannot replace evaluations of measurement quality, transferability, predictive error, or intervention effects.

3.3. Data adequacy and method selection

Timing constrains data selection. Final grades and complete learning histories summarize outcomes but come too late for early decisions. Early submission records may help prioritize review [10, 11]. Support decisions also have to take current learning performance into account. Sequence analysis, code changes, and error information may provide insight into learning processes [8-12]. These interpretations have to be checked against representative code samples or learning records. Method selection depends on the goal of the analysis and the level of detail required. Early risk prediction uses early performance data. Cognitive diagnosis uses assessment feedback and problem-to-concept mappings. Conclusions about specific knowledge points need to be double-checked against actual assignments. Just having suitable data is not enough—quality checks should also look at relationships between records, duplicates, identifiers, missing values, and event sequences. The normalization criteria should also be clear, including whether the number of tasks or active weeks is used. Preprocessing parameters have to be learned only from the training data and applied to the validation and test data.

Grouping analysis involves several considerations from the outset. Group size, correlations, stability, and how interpretable the results are—all of these matter. Silhouette scores sometimes help decide on cluster numbers, though not always conclusively. Interpreting group labels often depends on learning outcomes or what students actually produced.

For sequence analysis, reports need to clarify the types of events analyzed. Session boundaries—where they start and end—should be described. The minimum support thresholds used for pattern detection are important to state. Handling of redundant patterns (if any) requires explanation. Reports should also note which code snapshots, if any, were used to give additional context. If categories are drawn too broadly, they may mask meaningful differences between students. If drawn too narrowly, they may produce sparse patterns. Higher-performing students were more likely to show behavior patterns that were interpreted as top-down design [12]. Lower-performing students tended to repeatedly add, edit, and run code. They also often tested code blocks before inserting them. Such interpretations depend on both the type of task and the block-based programming environment.

Global feature importance summarizes the overall contribution of features to model predictions, and individual feature contributions explain a single prediction. Shapley additive explanations (SHAP) establish neither concept deficits nor intervention effects [11]. If the available data are insufficient, the intended decision cannot be supported. This limitation is shown in Figure 1.

DOI: 10.54254/2755-2721/2026.LD38058

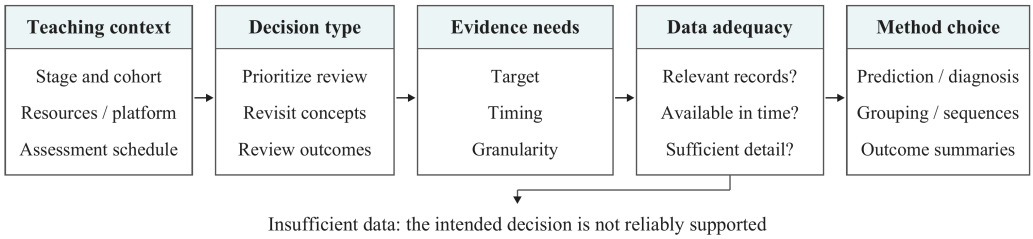

Figure 1. From teaching context to evidence and method choice

The framework proposed in this section is a synthesis of the reviewed literature, not the result of an empirical validation. Its suitability for teaching decisions has not been tested.

4. Comparative evaluation and discussion

Table 1 summarizes the contexts, samples, methods, and reported results of the ten empirical or application studies. The reported values do not support a unified performance ranking. The evaluation conditions and limitations are discussed below.

Table 1. Overview of the ten empirical or application studies included in the review

Study	Context and data	Method and purpose	Reported result
[3]	216 first-year CS students from four universities	Multinational programming-skill assessment	Mean score: 22.89/110
[8]	About 400 students; four cohorts in two programming courses	Community detection and pass/fail prediction from learning logs	Cleaned gradient boosting: accuracy .79, F1 .785, ROC-AUC .81
[12]	106 students; five Snap assignments	Differential sequence mining (CDSM-G) plus AdaBoost	First assignment: accuracy .792, precision .833, recall .769
[9]	2,080 students; two courses, three annual editions each	Privacy-friendly pass/fail prediction from submission metadata	Balanced accuracy about 60%-70% initially and 75%-80% after the first evaluation
[10]	468 students from 12 CS1/CS2 course offerings	Early multiclass prediction using grades, delivery time, and attempts; prediction windows at weeks 3, 5, and 7	Reported GBC F1: 86%, 94%, and 96% at weeks 3, 5, and 7, respectively
[11]	772 students; two semesters and five assignments	Stacked regression with SHAP and mixture-model profiles; 10-fold cross-validation repeated ten times	Final exam grades on a 0–1 scale. All five assignments: RMSE .151, R^2 .55; first assignment only: RMSE .201, R^2 .35

DOI: 10.54254/2755-2721/2026.LD38058

Table 1. (continued)

[14]	493 students; 2,452 Python-course test records	Entropy-adaptive differentially private federated MLP; 80:20 training/test split at the test-record level	Reported test accuracy: EADP-FedAvg 92.7%, DP-FedAvg 88.6%. Reported "macro average score": .921; metric definition unclear
[13]	650 students across 36 class sections in Fall 2024 and Winter 2025	Six models using multimodal data and temporal validation	Reported "accuracy": Fall XGBoost 91.24%; Winter RF 84.28%, XGBoost 83.77%
[6]	253 students in a 16-week Java course	At-risk identification and 12 evaluated interventions	Students showing improvement in attendance or assignment practice: attendance 83% (15/18); assignment practice 55% (36/65)
[15]	531 students, 206 problems, and four programming courses	P-FuzzyCDF for score prediction and knowledge diagnosis	Programming Fundamentals: MAE .07, MSE .09, RMSE .01 (see Note)

Note. Values are reproduced as reported in the original studies. The definition of "macro average score" is unclear in the federated learning study [14]. The MAE, MSE, and RMSE values for P-FuzzyCDF are reproduced from Table 6 of the original article [15]. Its reported MSE and RMSE values do not satisfy the stated relationship, RMSE = √MSE. These error values should be interpreted with caution. Differences in tasks, samples, labels, observation windows, validation designs, and outcome scales limit direct comparison.

4.1. Performance and instructional use

Performance is assessed separately for different task types: classification tasks use precision, recall, F1 score, area under the receiver operating characteristic curve (ROC-AUC), and balanced accuracy; continuous outcomes use mean absolute error (MAE), root mean squared error (RMSE), or R-squared [5, 10, 11]. Comparisons with baseline models help assess whether added complexity improves predictive performance. Class-specific precision and recall can show errors that might be hidden by overall accuracy.

F1 scores of 86%, 94%, and 96% were reported in weeks 3, 5, and 7, respectively [10]. An accuracy of 0.792 was reported using first-assignment data [12]. Groups were defined by a median grade split. The target was relative performance rather than the direct identification of learning difficulty.

An accuracy of 91.24% was reported for the Fall XGBoost model, 84.28% for the Winter RF model, and 83.77% for the Winter XGBoost model [13]. The reported regression "accuracy" is not clearly defined and therefore cannot be interpreted as standard classification accuracy. The authors also do not describe the feature window clearly enough to explain how the claimed one-month prediction lead is achieved. It is therefore unclear whether the timing of this prediction is appropriate.

Timing determines when teachers can intervene. The results from week 3 allow more time for instructional support than those from week 7 [10]. Using assignment completion dates instead of calendar dates delays the assessment of late submissions [11]. Studies should state their objectives, deadlines, and evaluation units, as well as whether results are based on random splits or a new group of students. The programming assessment and the intervention outcomes address questions that differ from simply assessing prediction accuracy [3, 6].

DOI: 10.54254/2755-2721/2026.LD38058

In the simulated federated setting, test accuracy was 92.7% with EADP-FedAvg, 88.6% with DP-FedAvg, and 96.9% with centralized learning [14]. The input features included score-derived features, and the data were split at the individual test-record level. Because the future prediction targets and student-level separation between the training and test sets are not clearly specified, it is unclear whether these methods are suitable for prospective evaluation involving previously unseen students. Moreover, when training noise is adjusted using test-set prediction entropy, there are concerns about the independence of the evaluation.

The RMSE and R-squared values were 0.151 and 0.55, respectively, when data from all five assignments were used, compared with 0.201 and 0.35 when only first-assignment data were used [11]. Early warning capability would be assessed using the first-assignment window; results based on the complete set do not establish this capability. The concept estimates are appropriate for concept-focused practice only if there are good problem-to-concept mappings and checks against students' work [15]. The reported MSE and RMSE values are inconsistent with each other (see the note to Table 1), which limits the interpretation of these error metrics [15].

Grouping analysis highlights variation among students. Sequence analysis gets at how learning unfolds step by step. Risk prediction helps set priorities for review, but actually pinpointing who needs support requires more granular, case-by-case information. The reported percentages reflect the proportions of intervention recipients whose attendance or engagement in practice improved—these numbers do not speak to grade changes or prove the intervention caused improvement [6].

When the system signals an alert, teachers might look over a student's recent assignments for recurring mistakes or shifts in engagement. Responses could range from immediate feedback or individual tutoring to suggesting more practice or just keeping an eye on progress. Setting the threshold for intervention involves weighing the risk of missing students in need against the limits of available resources. Afterwards, it is necessary to check whether the support measures have made a difference in learning.

4.2. Limitations and future directions

The supporting evidence derives from a narrow range of courses, institutions, platforms, and programming environments [8-12]. More recent studies also examine specific educational settings [13, 14]. There are differences in course level and task design as well [10, 15]. Results observed in these contexts may not apply to all novice programmers. Validation processes ought to involve independent datasets and defined observation windows. Testing across multiple courses, institutions, and experience levels is needed to address generalizability. Federated learning does not transfer raw data, but its predictive accuracy must be evaluated separately for each context. It is important to exclude direct personal identifiers from predictive models. Where demographic variables are used, a clear rationale is required, and group-level prediction errors must be examined [7]. Researchers should document the deployment context, sources of uncertainty, privacy measures, and any instructional outcomes. This review is based on fifteen publications. There is no full record of the search and screening process, which restricts reproducibility and breadth. The framework proposed here still lacks empirical validation. Further work should evaluate both its impact on teaching decisions and the learning outcomes following intervention.

5. Conclusion

The present literature review examines how EDM evidence can inform instructional decision-making. The merit of such evidence rests on three core elements: target alignment, timing, and the

DOI: 10.54254/2755-2721/2026.LD38058

granularity of information. Research outputs ought to correspond to the specific instructional decisions under consideration and must be obtainable before the rollout of any instructional intervention. The level of granularity ought to fit the required operational action. Risk scores may indicate students in need of review, but the pinpointing of gaps within particular knowledge domains requires checks against students' actual assignments. The appraisal of such systems requires the consideration of measurement quality, generalizability, predictive error, model stability, privacy, and fairness. Reporting should explicitly define prediction targets, observation windows, and evaluation units. The conceptual framework delineated in this work has not been empirically validated. Subsequent research endeavors will need to explore the way it shapes instructional decisions and exerts effects on the learning outcomes of students after the implementation of supportive measures.

Future research should prioritize evaluating the framework in teaching practice. Such evaluation needs to examine whether the framework helps teachers interpret analytical findings and select suitable support. It should also assess whether the resulting interventions improve student learning. Predictive performance alone cannot answer these questions. Testing across institutions is necessary because course requirements, assessment practices, and student experience may affect both model performance and the usefulness of its outputs. Evidence from one setting offers limited grounds for adoption elsewhere. Researchers should specify when predictions become available, how teachers use them, and which learning outcomes are subsequently assessed. This would help identify the conditions under which the framework is useful and where it requires revision. Until such evidence is available, its value for instructional decision-making remains a question for empirical investigation.

References

[1] Cheah, C. S. (2020). Factors contributing to the difficulties in teaching and learning of computer programming: A literature review. *Contemporary Educational Technology, 12*(2), Article ep272. https://doi.org/10.30935/cedtech/8247

[2] Robins, A., Rountree, J., & Rountree, N. (2003). Learning and teaching programming: A review and discussion. *Computer Science Education, 13*(2), 137-172. https://doi.org/10.1076/csed.13.2.137.14200

[3] McCracken, M., Almstrum, V., Diaz, D., Guzdial, M., Hagan, D., Ben-David Kolikant, Y., Laxer, C., Thomas, L., Utting, I., & Wilusz, T. (2001). A multi-national, multi-institutional study of assessment of programming skills of first-year CS students. In *Working Group Reports from ITiCSE on Innovation and Technology in Computer Science Education* (pp. 125-140). Association for Computing Machinery. https://doi.org/10.1145/572133.572137

[4] Romero, C., & Ventura, S. (2020). Educational data mining and learning analytics: An updated survey. *WIREs Data Mining and Knowledge Discovery, 10*(3), Article e1355. https://doi.org/10.1002/widm.1355

[5] Pires, J. P. J., Brito Correia, F., Gomes, A., Borges, A. R., & Bernardino, J. (2024). Predicting student performance in introductory programming courses. *Computers, 13*(9), Article 219. https://doi.org/10.3390/computers13090219

[6] Utamachant, P., Anutariya, C., & Pongnumkul, S. (2023). i-Ntervene: Applying an evidence-based learning analytics intervention to support computer programming instruction. *Smart Learning Environments, 10*, Article 37. https://doi.org/10.1186/s40561-023-00257-7

[7] Baker, R. S., Esbenshade, L., Vitale, J., & Karumbaiah, S. (2023). Using demographic data as predictor variables: A questionable choice. *Journal of Educational Data Mining, 15*(2), 22-52. https://doi.org/10.5281/zenodo.7702628

[8] Mai, T. T., Bezbradica, M., & Crane, M. (2022). Learning behaviours data in programming education: Community analysis and outcome prediction with cleaned data. *Future Generation Computer Systems, 127*, 42-55. https://doi.org/10.1016/j.future.2021.08.026

[9] Van Petegem, C., Deconinck, L., Mourisse, D., Maertens, R., Strijbol, N., Dhoedt, B., De Wever, B., Dawyndt, P., & Mesuere, B. (2023). Pass/fail prediction in programming courses. *Journal of Educational Computing Research, 61*(1), 68-95. https://doi.org/10.1177/07356331221085595

[10] Llanos, J., Bucheli, V. A., & Restrepo-Calle, F. (2023). Early prediction of student performance in CS1 programming courses. *PeerJ Computer Science, 9*, Article e1655. https://doi.org/10.7717/peerj-cs.1655

DOI: 10.54254/2755-2721/2026.LD38058

[11] Hoq, M., Brusilovsky, P., & Akram, B. (2024). Explaining explainability: Early performance prediction with student programming pattern profiling. *Journal of Educational Data Mining, 16*(2), 115-148. https://doi.org/10.5281/zenodo.14246435

[12] Gao, G., Marwan, S., & Price, T. W. (2021). Early performance prediction using interpretable patterns in programming process data. In *Proceedings of the 52nd ACM Technical Symposium on Computer Science Education* (pp. 342-348). Association for Computing Machinery. https://doi.org/10.1145/3408877.3432439

[13] Bouallegue, S., Omri, A., & Al-Naemi, S. (2026). Machine learning approaches for early student performance prediction in programming education. *Information, 17*(1), Article 60. https://doi.org/10.3390/info17010060

[14] Chen, S., & Qi, X. (2025). Entropy-adaptive differential privacy federated learning for student performance prediction and privacy protection: A case study in Python programming. *Frontiers in Artificial Intelligence, 8*, Article 1653437. https://doi.org/10.3389/frai.2025.1653437

[15] Zhang, Y. T., Li, Z., Liu, Y., & Wu, Y. H. (2023). Student performance prediction based on cognitive diagnosis model. *Computer Systems & Applications, 32*(9), 239-247. [in Chinese] https://doi.org/10.15888/j.cnki.csa.009261