

From Partial Correctness to Completion: Predicting Learners' Early Success in Programming Practice

Md. Shahajada Mia

shahajadabrur@gmail.com

University of Aizu

Yutaka Watanobe

University of Aizu

Md. Mostafizer Rahman

University of Notre Dame

Md Faizul Ibne Amin

University of Aizu

Daniel M. Muepu

University of Aizu

Article

Keywords: Programming education, online judge systems, early success prediction, partial correctness, machine learning, learning analytics

Posted Date: July 21st, 2026

DOI: <https://doi.org/10.21203/rs.3.rs-9831273/v1>

License: © ⓘ This work is licensed under a Creative Commons Attribution 4.0 International License.

[Read Full License](#)

Additional Declarations: No competing interests reported.

From Partial Correctness to Completion: Predicting Learners' Early Success in Programming Practice

Md. Shahajada Mia^{1,2,*}, Yutaka Watanobe¹, Md. Mostafizer Rahman³, Md Faizul Ibne Amin¹, and Daniel M. Muepu¹

¹Graduate Department of Computer and Information Systems, The University of Aizu, Aizuwakamatsu, Fukushima 965-8580, Japan

²Department of Statistics, Pabna University of Science and Technology, Pabna-6600, Bangladesh

³Lucy Family Institute for Data & Society, University of Notre Dame, IN, USA

*shahajadabrur@gmail.com

ABSTRACT

Programming practice is a progressive learning process in which learners improve their skills through repeated submissions and gradual progress toward correctness. In online judge systems, early prediction of problem completion is important for understanding learner progress during problem solving. However, many existing approaches rely mainly on final binary outcomes, which may not adequately capture intermediate progress and learners' evolving capability. To address this gap, this study proposes an early success prediction framework that evaluates three formulations: binary outcome, ordinal correctness, and a proposed partial-correctness and progress-aware formulation. Using Aizu Online Judge (AOJ) submission logs, the prediction task determines whether a learner will solve a problem within the next K attempts based on early submission behavior. Experimental results demonstrate that the proposed formulation consistently outperforms binary-outcome and ordinal-correctness formulations across six machine learning models and evaluation metrics. On the Algorithm and Data Structures I (ALDS1) test set, random forest achieved the best performance under this formulation, showing relative improvements in PR-AUC and accuracy by 6.20% and 6.63%, respectively, over the binary-outcome formulation, and by 5.87% and 6.78%, respectively, over the ordinal-correctness formulation. Additional AOJ course evaluation and K sensitivity analysis further supported the stability of the formulation. Risk-based analysis also showed clearer separation among the three risk groups than the other two formulations, with low-risk groups showing more favorable attempt-wise success patterns than high-risk groups. Overall, the proposed formulation more effectively captures learning progression and provides an interpretable basis for early identification and instructional support in programming problem-solving environments.

Keywords: Programming education; online judge systems; early success prediction; partial correctness; machine learning; learning analytics

Introduction

Programming skills are increasingly important in industry, education, and research in the modern digital era. To support the development of these skills, programming education now relies heavily on practice-oriented learning environments, in which students improve through iterative problem-solving rather than solely through traditional classroom-based instruction^{1,2}. In such environments, learners repeatedly submit code, receive feedback, revise their solutions, and gradually improve their performance on programming tasks¹. Online judge (OJ) systems are particularly well-suited to support this process because they automatically compile and execute submitted code while recording detailed histories of verdicts, attempts, and problem-solving activities³. Unlike conventional educational settings that often emphasize only final outcomes, OJ systems capture rich behavioral data throughout the problem-solving process, making them valuable for analyzing learning progress and supporting data-driven research and early intervention in programming education⁴⁻⁶.

In this context, OJ systems contain problems of varying difficulty, and learners' attempt patterns and solving behavior often differ according to their proficiency and problem complexity^{4,7}. This issue is particularly important for novice programmers, who may struggle with complex problems within a limited number of attempts because of limited understanding of programming syntax, logic, and problem-solving strategies. Prior research has reported low success rates and weaker performance in introductory programming courses and Massive Open Online Courses (MOOCs)^{8,9}. Therefore, early prediction of learner success is relevant for supporting learning progress and providing timely, and effective assistance to struggling learners during the problem-solving process. In OJ systems, this need is especially important because learners progress gradually toward correctness through repeated submissions, automatic feedback, and successive revisions^{1,4}.

In recent years, educational data mining (EDM) and learning analytics (LA) techniques have been widely applied to predict student performance across diverse educational settings^{8,10–14}. Within such settings, EDM and LA support the extraction of meaningful patterns and predictive insights from educational data generated in digital learning platforms, such as OJ systems, MOOCs, and learning management systems, thereby helping improve the learning process^{6,15}. Prior studies have used a range of machine learning (ML) models, such as logistic regression (LR), random forest (RF), decision tree (DT), multilayer perceptron (MLP), Naive Bayes (NB), and K-nearest neighbor (KNN), to model learner outcomes from behavioral and academic data^{8,11,13,14,16,17}. Most of these studies focused on broader educational targets such as success/failure, exam grades, student dropout risk, and academic performance^{8,9,11,12,16–19}. However, problem-level early success prediction in programming tasks remains comparatively less explored despite the availability of fine-grained interaction data in programming learning environments.

Despite the progress of existing prediction studies, representing learner performance in programming tasks remains challenging because problem-solving ability often develops through multiple interactions rather than through a single final outcome. This perspective is also reflected in knowledge tracing (KT) research, where learner interactions are used to understand evolving performance from a sequence of past responses over time^{20,21}. Nevertheless, many existing KT methods represent learner responses mainly using binary outcomes^{20,21}, although partial-credit information may provide a richer representation of learner states than binary outcomes alone²². In OJ-based programming environments, learners often progress through repeated submissions for each programming problem, and partial scores based on the number of passed test cases among total cases can provide indication of how close learners are to full acceptance across submissions^{4,23}. However, the use of partial correctness and progress-related signals for early prediction of learner problem-solving outcomes in OJ-based programming tasks remains underexplored.

Motivated by this gap, this study investigates how learner problem-solving outcomes within the next K attempts can be predicted more effectively in OJ systems, where timely prediction can support intervention during ongoing problem solving. Rather than focusing only on which classifier performs best, the study addresses a more fundamental question: how should the prediction task itself be formulated to represent programming behavior effectively? To answer this question, the study examines three prediction formulations using submission-behavior, temporal, and progress-aware features, together with their implications for interpreting learner progress in programming tasks. The main contributions of this study are summarized as follows:

- We propose an early-success prediction framework for programming problem-solving outcomes in OJ environments. The framework evaluates three learner-problem formulations including binary outcome, ordinal correctness and a proposed partial-correctness and progress-aware formulation.
- We examine how prediction formulations affect early-success prediction by evaluating them across six ML classifiers such as LR, KNN, MLP, DT, Extreme Gradient Boosting (XGB), and RF under the same experimental protocol.
- Experiments on the large-scale Algorithm and Data Structures I (ALDS1) dataset from the Aizu Online Judge (AOJ) system demonstrate that our proposed formulation consistently outperforms the other two formulations across all classifiers using multiple classification metrics. Validation on the Computational Geometry (CGL) further supports its applicability in OJ-based programming education.
- The framework also converts predicted success probabilities into learner-problem risk groups, providing an interpretable basis for analyzing near-term success patterns and attempt-wise problem-solving behavior that can support educational monitoring, feedback, and problem-set refinement.

The rest of this paper is organized as follows: [Related Work](#) section reviews related studies in EDM and LA techniques relevant to this work. [Proposed Approach](#) section presents the dataset, preprocessing procedures, task formulations, and predictive models used in this study. [Experimental Setup](#) section describes the experimental settings, including hyperparameter tuning for each classifier through five-fold cross-validation (CV), and evaluation metrics. The comparative results of three formulations across several ML models are reported, and related statistical analyses are presented in the [Experimental Results](#) section. [Discussion](#) section discusses the major findings, limitations of the study, and future research directions. Finally, [Conclusion](#) section concludes the paper by highlighting major outcomes.

Related Work

This section reviews existing studies on predictive models in programming education and related areas. In addition, different modeling perspectives, such as binary and continuous outcomes, are discussed to establish the background of this study and highlight the research gap.

Predictive Modeling Approaches

Early prediction of learner success is an important topic in EDM and LA because it can provide early indications of learner progress and difficulty before task completion. Accordingly, many existing studies have explored this task using a variety of data sources, features, and ML methods across educational environments^{8,11–13,16,18}. Most of these studies adopted ML methods for their ability to capture hidden and more complex patterns in large-scale educational data and support effective prediction. Several studies have also focused on the early identification of at-risk students to support student success^{24–26}, as timely identification can enable intervention, improve engagement in study, and reduce adverse academic performances²⁶. For example, study²⁴ applied ML models to identify at-risk students and showed that non-traditional students exhibited lower accuracy (Acc) than all students, while study²⁵ examined early-stage dropout prediction using SVM, RF, KNN, and stochastic gradient descent and reported that KNN achieved the best performance.

Beyond at-risk identification, prior work has examined educational outcome prediction tasks, including final grade estimation, academic performance modeling, and classification of dropout or academic success^{11–13,16,18}. For instance, final grades were predicted from previously obtained grades¹¹, and early-stage risk factors were combined with predictive models to estimate student performance using grades, admission test scores, and related attributes¹⁸. Similarly, Jang et al.²⁷ compared multiple ML methods for student performance prediction and reported that LR achieved the best performance, while also using explainable artificial intelligence (XAI) to interpret the classification results. Generally, grade-related features have been closely associated with academic outcomes, whereas risk-related factors are strongly linked to dropout prediction^{12,13}.

Comparisons between classical and ensemble ML methods further show that ensemble approaches, particularly RF, can provide stronger predictive performance in exam-related outcomes¹³. Likewise, study¹⁶ classified students as graduate, dropout, or enrolled and found that boosting algorithms such as CatBoost and XGBoost outperformed traditional models, including LR, RF, and SVM. Related predictive settings have also been explored in more specific contexts, such as final grade prediction in a gamified course using current semester performance²⁸, early prediction of at-risk students in a power electronics course²⁹, and gaming preference prediction³⁰. Therefore, these studies mainly focused on general academic outcomes rather than more specific learning progress during ongoing tasks.

Prediction in Programming Education

Predicting student performance has become an important topic in programming education, particularly in introductory programming courses where many learners struggle, and failure rates are often high. Therefore, several researchers experimented on this course, on how to support and reduce the failure rate using ML methods. Costa et al.³¹ investigated failure prediction in introductory programming courses using SVM, DT, neural network (NN), and NB models after preprocessing and hyperparameter-tuning, and reported that SVM achieved the highest performance. Study³² modeled student progress and dropout among upper primary and high school students, showing that DT outperformed SVM, RF, and LR. Another study used ensemble stacking to predict final grades for estimating student achievement from assignment submission data³³, and a systematic review highlighted the growing adoption of data-driven approaches for predicting student performance in higher education, including programming-related contexts³⁴. In addition, educational datasets often exhibit class imbalance and skewed feature distributions, which can reduce classifier effectiveness and bias predictive results. Addressing this issue, Llanos et al.⁸ applied preprocessing and data balancing techniques to a 16-week programming course dataset, and evaluated eight classifiers using *Acc*, *recall (Rec)*, *F1 score (F1)*, and *AUC*. Their results showed that gradient boosting achieved the strongest overall performance with an *F1* of 86%.

In OJ-based programming education, prior studies have used submission-level interaction data to analyze learner performance and programming-related outcomes^{4,6,7,15,35–37}. However, most of this work has focused on comprehensive analytical tasks such as problem-solving behavior analysis, error understanding, student profiling, and problem recommendation, rather than success prediction during ongoing problem solving. For instance, study³⁵ applied DT and SARIMA models to processed AOJ data with log transformation to predict student submission and outcome patterns. Similarly, study³⁷ used multi-instance learning to identify student profiles from OJ assignment data, compared the results with ML methods, and applied XAI to show interpretability. Pereira et al.³⁸ also investigated dropout prediction in programming courses in OJ to improve students' performance and reduce instructors' workload. Collectively, these studies highlight the usefulness of OJ submission data for examining submission patterns and performance in programming tasks.

Knowledge Tracing and Progress Modeling

KT has been widely used to model learners' evolving knowledge states from sequences of past interactions and to predict future performance^{21,39}. Corbett and Anderson³⁹ introduced this perspective in a programming-related setting by modeling procedural knowledge acquisition while students learned to write short programs with the ACT Programming Tutor. Shen et al.⁴⁰ surveyed KT models, variants, and applications, highlighting the role of KT in tracking student progress in learning environments. In addition, learner progression may be influenced by factors such as the ordering of conceptually related tasks and learner fatigue during the learning process⁴¹. Prior studies have applied KT in programming-related tasks and other educational settings to

model knowledge progression and support learner modeling^{20,21,39,40}. Study²² further extended KT beyond binary correctness by incorporating partial credit through continuous scores, thereby proving that students' performance may be represented more effectively as a continuum than correct-or-incorrect outcomes.

Beyond KT, study³² used prior module content to predict student progress on the final problem in large-scale online programming courses, showing that interaction features can be used to forecast subsequent learning progress. Similarly, Gao et al.⁴² analyzed semester-long scores, engagement, and code metrics in a programming course and found that engagement and score were strongly associated, while code complexity decreased over time. Study⁴³ further showed that students' algorithmic-thinking development can be characterized through growth trajectories across multiple time points. In OJ-based settings, study⁴ demonstrated that problem difficulty and ordering influence users' performance and highlights the relevance of sequencing in programming tasks. As a whole, these studies provide a conceptual foundation for modeling learning as a dynamic process across sequential interactions and repeated attempts.

Research Gap and Rationale

The studies reviewed in Sections [Predictive Modeling Approaches](#) to [Knowledge Tracing and Progress Modeling](#) show that various ML methods can be effective for programming-related prediction, although their performance often depends on data characteristics and prediction settings. However, most existing studies have focused on broader educational outcomes, while comparatively less attention has been given to attempt-level early problem-solving prediction in OJ environments. In particular, predicting whether a learner will solve a problem within the next K attempts by considering their progression within the same problem remains underexplored. These limitations motivate the present study, which examines three prediction formulations across multiple ML algorithms for modeling near-term problem-solving success over repeated attempts in an OJ-based programming task.

Proposed Approach

This section presents the proposed framework for predicting near-term problem-solving success in OJ systems. The overall workflow of the study contains four major phases and is summarized in Fig. 1. First, submission data are collected from two courses in the AOJ system, and prepared and preprocessed for analysis. Second, the prediction task is represented through three prediction formulations, followed by a stratified user-based train-test split. Third, six ML models are trained for each formulation, and five-fold CV on the training set is used to select the best hyperparameter settings before final evaluation on the held-out test set. Finally, the predictive outcomes are further analyzed through risk-group profiling and related analyses to derive educational insights into learner problem-solving patterns.

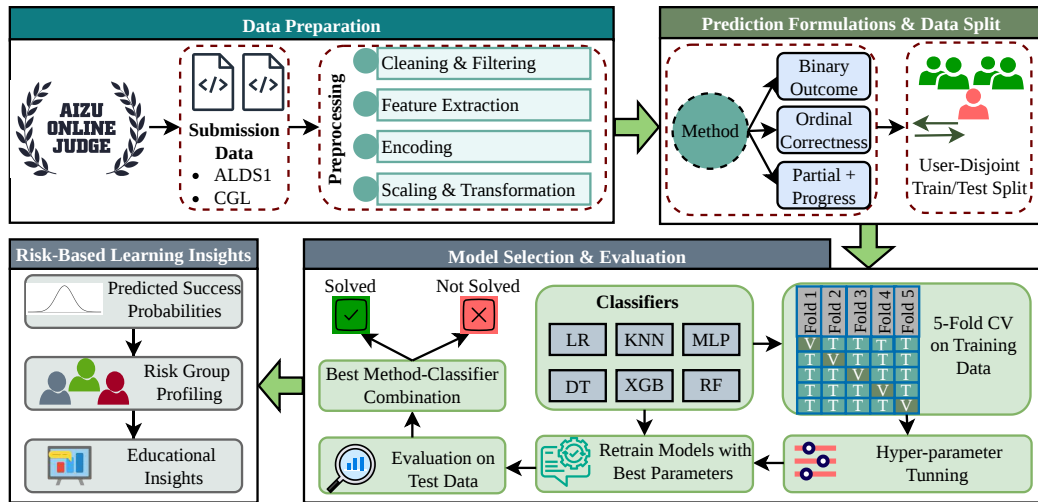


Figure 1. Overview of the proposed early-success prediction framework. In the 5-fold CV step, T and V denote the training and validation sets, respectively, for each fold.

Prediction Task Description

Let, $\mathcal{D}$ be the set of learner-problem-attempt interaction records obtained from the AOJ submission data. Each interaction point corresponds to a learner $U_j \in \mathcal{U} = \{U_1, U_2, \dots, U_M\}$, a problem $P_i \in \mathcal{P} = \{P_1, P_2, \dots, P_N\}$, and an attempt index t . Each point

is represented by a feature vector $\mathbf{x}_i = [x_{i,1}, x_{i,2}, \dots, x_{i,d}]$, and the feature matrix is represented as (1).

$$\mathbf{X} = \begin{bmatrix} x_{1,1} & x_{1,2} & \cdots & x_{1,d} \\ x_{2,1} & x_{2,2} & \cdots & x_{2,d} \\ \vdots & \vdots & \ddots & \vdots \\ x_{n,1} & x_{n,2} & \cdots & x_{n,d} \end{bmatrix} \quad (1)$$

where $\mathbf{X} \in \mathbb{R}^{n \times d}$, n , and d denote the number of prediction points and extracted features, respectively.

For each prediction point, only information available up to attempt t is used to construct $\mathbf{x}_{U_j, P_i, t}$, which prevents future-information leakage and reflects an accurately future prediction settings for a specific learner-problem pair. In this task, a learner may submit several incorrect or partially correct attempts without obtaining an accepted solution by attempt t . Therefore, the goal of this study is to predict whether the learner solve the same problem (P_i) within the next K attempts using only the interaction history available up to attempt t . The binary output of learner next K attempts success label is defined as:

$$y_{U_j, P_i, t}^{(K)} = \begin{cases} 1, & \text{if } U_j \text{ solves } P_i \text{ within attempts } t+1, \dots, t+K, \\ 0, & \text{otherwise.} \end{cases} \quad (2)$$

All three formulations used in this study share the same prediction output, but their main difference lies in the input representation used for outcome prediction.

Prediction Formulations

To compare different learner-state representations, three formulations are constructed for the shared near-term future success prediction task: binary outcome (M_1), ordinal correctness (M_2), and partial correctness and progress-aware (M_3) formulation. Table 1 summarizes the input feature groups, and purpose of each formulation. In M_1 , the model uses only base interaction features as a conventional baseline. These features describe general learner submission behavior, such as attempt index, time gap between two successive submissions, and submission status indicators including Wrong Answer (WA), Runtime Error (RE), Presentation Error (PE), Compilation Error (CE), Time Limit Exceeded (TLE), Memory Limit Exceeded (MLE), and Output Limit Exceeded (OLE).

For formulation involving correctness information, a submission-level correctness score (s) is computed from the proportion of passed test cases among total test cases for each submission, and formulated as follows:

$$s_{U_j, P_i, t} = \frac{N_{U_j, P_i, t}^{PC}}{N_{U_j, P_i, t}^{TC}} \quad (3)$$

where $N_{U_j, P_i, t}^{PC}$, and $N_{U_j, P_i, t}^{TC}$ denotes the number of passed test cases and the total number of test cases for the submission made by learner U_j for problem P_i at attempt t , respectively. The score ranges from 0 to 1, where 0 indicates that no test cases were passed, and values closer to 1 indicate higher partial correctness.

Formulation	Feature Groups	Features	Purpose
Binary outcome	Base interaction	Attempt index, time gap, submission-status indicators	Captures general learner-problem interaction patterns from submission behavior.
Ordinal correctness	Base interaction + Ordinal correctness	Attempt index, time gap, submission-status indicators, correctness level (0–5)	Represents learners' intermediate correctness states beyond binary success/failure.
Partial-correctness + Progress-aware	Base interaction + partial-correctness + progress-aware	Attempt index, time gap, submission-status indicators, partial-correctness score (current), correctness gain, peak correctness	Capture learners' partial solution quality in individual attempts and their progression across repeated attempts.

Table 1. Feature groups used in the three prediction formulations.

In M_2 , ordinal correctness information is integrated with base interaction features. For non-accepted prediction attempts, s is converted into ordered levels: $s = 0$ is assigned level 0, $0 < s \leq 0.20$ level 1, $0.20 < s \leq 0.40$ level 2, $0.40 < s \leq 0.60$ level 3, $0.60 < s \leq 0.80$ level 4, and $0.80 < s < 1.00$ level 5. This formulation allows the model to capture different degrees of correctness while preserving the same binary near-term success output.

In M_3 , s is used directly as a partial correctness value and combined with progress-aware features and base interaction indicators. These progress-aware features include the correctness gain between two consecutive attempts and the peak

correctness, based on the maximum correctness score observed up to attempt t across repeated submissions. The correctness gain is computed as follows:

$$\Delta s_{U_j, P_i, t} = s_{U_j, P_i, t} - s_{U_j, P_i, t-1} \quad (4)$$

A positive value of $\Delta s_{U_j, P_i, t}$ indicates improvement between consecutive attempts, whereas 0 or negative values indicate stable and reduced progress, respectively. Therefore, M_3 is designed to capture both the current partial solution quality and the direction of problem-solving progress across repeated attempts, providing richer representations for solving the problem within the next K attempts.

Predictive Models

After constructing the three prediction formulations, each formulation was systematically evaluated using the same set of ML classifiers to predict learner success within the next K attempts. Applying the same classifiers across formulations supports a fair controlled comparison of learner-problem representations and reduces potential model-selection bias. The selected classifiers represent complementary modeling perspectives: LR⁴⁴ for linear probabilistic prediction, KNN⁴⁵ for distance-based similarity, MLP⁴⁶ for nonlinear neural learning, DT⁴⁷ for interpretable tree-based prediction, RF⁴⁸ for bagging-based ensemble learning, and XGB⁴⁹ for gradient-boosting ensemble learning. This model diversity enables robust comparative evaluation of the prediction formulations under different learning mechanisms and model assumptions.

Experimental Setup

This section describes the experimental settings used in this study. It first describes the data source and preprocessing, followed by the model training process including CV and hyperparameter tuning. In addition, the evaluation metrics formulated to assess model performance are presented.

Dataset and Preprocessing

This study collected submission data from two courses in the AOJ system, namely ALDS1 and CGL^{23,50}. It used secondary data comprising fully anonymized educational records from the AOJ platform, with no direct participant interaction or access to personally identifiable information. Per the AOJ Terms of Use, submitted data may be used for research purposes (https://judge.u-aizu.ac.jp/onlinejudge/submission_note.jsp); ethical approval and informed consent were both exempted by the Research Ethics Committee, University of Aizu. All methods were carried out in accordance with the relevant guidelines and regulations. ALDS1 contains foundational algorithm and data-structure problems such as sorting, tree, and graph tasks⁵¹, whereas CGL focuses on computational geometry tasks, including projection, orthogonal relations, and intersection tasks⁵². The extracted AOJ system provides large-scale activity, including more than 10 million submissions from approximately 0.17 million users across over 3000 problems from programming courses and contests. Each record contains learner-problem interaction information, including user ID, Problem ID, Judge ID, verdict status, accuracy, code size, and programming languages such as C, C++, Java, and Python²³. Here, an attempt denotes a single submission for a learner-problem pair. Based on these records, learner-problem sequence data were constructed for the next K attempts prediction task, with $K = 5$ used as the primary setting; the data preparation workflow is summarized in Fig. 1.

First, the raw submission logs were cleaned by removing invalid records, converting timestamps, and arranging attempts sequentially within each learner-problem sequence. For both courses, submissions from January 2020 to October 2025 were retained, and users with at least 30 submissions were included to ensure sufficient interaction history for sequence-based prediction. The remaining data were then organized into valid learner-problem sequences with sufficient future attempts for the next- K -attempt prediction task. From these sequences, accuracy was used to derive partial-correctness information and attempt-level features were extracted including attempt index, time gap, peak correctness, correctness level and submission-status. These features are summarized for each prediction formulations in Table 1 and described in Prediction Formulations section. For each prediction point, the binary target outcome was assigned as 1 if an accepted submission occurred within the next K attempts for the same learner-problem pair and 0 otherwise. Current accepted submissions were excluded because the problem had already solved, and the final K -attempts of each sequence were removed due to insufficient future attempts for label construction. Then, categorical features were encoded. Finally, skewed numerical features were transformed and standardized before model training.

The summary statistics of the final prediction-instance datasets are shown in Table 2. ALDS1 contains 58 problems, 7255 users and 295047 prediction instance, whereas CGL contains 25 problems, 438 users and 10025 prediction instances. These statistics highlight that ALDS1 provides substantially larger interaction dataset, while CGL serves as smaller dataset for additional evaluation.

Characteristics	Type	ALDS1	CGL
Problem #	–	58	25
Users	–	7255	438
Submissions	–	295047	10025
Avg. submissions per user	Solved	17.91	13.33
	Not solved	25.82	14.60
Avg. submissions per problem	Solved	2203.98	228.16
	Not solved	2883.03	172.84

Table 2. Summary of the experimental dataset.

The continuous features were examined by descriptive statistics, including mean, standard deviation (SD), skewness and variance inflation factor (VIF), as reported in Table 3. Attempt index showed large variability ($SD = 172.58$) with an average of 59.21, indicating substantial differences in repeated-attempt behavior across learner-problem sequences. The skewness values further showed that attempt index and time gap were highly positive skewed, with values of 5.64 and 19.01, respectively, reflecting repeated and irregular submission behavior. Therefore, logarithmic transformation was applied to these skewed features to reduce the influence of extreme values and compress large ranges⁵³. The mean and SD also varied markedly across features, for example, correctness gain showed a relatively small scale of -0.08 ± 0.36 , whereas time gap showed a much larger scale of 343781.77 ± 3580221.35 . This variation highlights the requirement of preprocessing: after logarithmic transformation, z-score standardization was applied to numerical features to ensure a consistent feature scale across all models. Finally VIF was used to assess multicollinearity among numerical features, since highly collinear predictors may provide limited information during model training. Since all VIF values were below 2, well under the common threshold of 5, multicollinearity was minimal among the selected features, supporting their suitability for subsequent modeling⁵⁴.

Feature	Mean $\pm$ SD	Skewness	VIF
Attempt Index	59.21 $\pm$ 172.58	5.64	1.20
Correctness (Current)	0.16 $\pm$ 0.29	1.59	1.67
Correctness Gain	-0.08 $\pm$ 0.36	-0.98	1.49
Time Gap	343781.77 $\pm$ 3580221.35	19.01	1.03
Peak Correctness	0.49 $\pm$ 0.45	-0.01	1.71

Table 3. Descriptive statistics and VIF of the continuous feature set.

Validation Protocol and Hyperparameter Settings

To compare ML model performance across prediction formulations, a user-disjoint stratified data split was applied, with 80% of users assigned to the training set and 20% to the test set. This ensured that submissions from the same users did not appear in both sets, thereby reducing user-level information leakage. As shown in Fig. 2, the solved and not-solved proportions with submissions were broadly comparable across training, testing and full sets for both courses, although the outcome distributions differed slightly between ALDS1 and CGL. For example, in the training set, ALDS1 contained lower proportion of solved cases than not-solved; while CGL showed higher solved proportion.

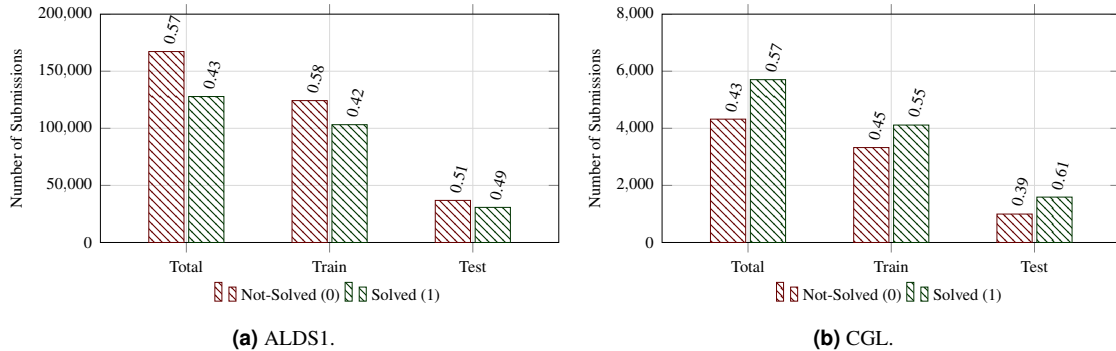


Figure 2. Submission distributions across user-disjoint splits. The proportion of labels is annotated on the bars.

Five-fold GroupKFold CV was then applied to the ALDS1 training set, using the same user-level grouping to avoid user overlap across validation folds⁵⁵. For each ML model and formulation method, hyperparameters were searched within the

candidate ranges reported in Table 4, while other implementation settings, such as solver, optimizer, maximum iterations, and random seed, were kept consistent according to the experimental protocol. The best configuration was selected based on the highest mean precision-recall area under the curve (PR-AUC) across the validation folds. The selected configurations were then retrained on the full ALDS1 training set and evaluated on the held-out ALDS1 test set using multiple evaluation metrics. Model-specific imbalance handling was applied only during training, where appropriate, while the validation and test sets were kept unchanged⁵⁶. The full CV method and classifier evaluation procedure are summarized in Algorithm 1. The selected hyperparameter settings from ALDS1 were also applied to CGL without additional tuning to examine performance consistency in an additional course dataset.

Model	Hyperparameter and search values
LR	<code>regularization (C) = {0.001, 0.01, 0.1, 1.0, 10, 100, 1000}</code>
KNN	<code>n_neighbors = {2, 3, ..., 15}, weights = {uniform, distance}, metric = {manhattan, euclidean}</code>
MLP	<code>hidden_layer_sizes = {(32,), (64,), (64, 32), (128, 64)}, alpha = {0.001, 0.0001, 0.00001}, learning_rate_init = {0.001, 0.0001, 0.0003}, activation = relu, batch_size = 256, solver = adam</code>
DT	<code>max_depth = {None, 2, 3, 5, 10}, min_samples_leaf = {1, 2, 5, 10}, min_samples_split = {2, 5, 10}</code>
XGB	<code>n_estimators = {300, 500, 1000}, max_depth = {2, 3, 5, 10}, min_child_weight = {1, 2, 5, 10}, learning_rate = {0.01, 0.03, 0.05, 0.10}</code>
RF	<code>n_estimators = {300, 500, 1000}, max_depth = {None, 2, 5, 10, 20}, min_samples_leaf = {1, 2, 5, 10}, min_samples_split = {2, 5, 10}</code>

Table 4. Hyperparameter settings for the classifiers.

Algorithm 1 Cross-Validated Formulation and Classifier Evaluation

Input: Preprocessed dataset $\mathcal{D}$, prediction formulations $M = \{M_1, M_2, M_3\}$, classifiers $C = \{\text{LR, KNN, MLP, DT, XGB, RF}\}$

Output: Final held-out test set performance for all formulation–classifier combinations

- 1: Split $\mathcal{D}$ into 80% training and 20% test sets using user-disjoint stratified sampling
 - 2: **for** each fomulation $m \in M$ **do**
 - 3: Construct formulation-specific train/test feature representations
 - 4: **for** each classifier $c \in C$ **do**
 - 5: Perform five-fold CV with hyperparameter tuning on formulation-specific training set
 - 6: Select the best hyperparameter setting based on mean *PR-AUC*
 - 7: Retrain c on the full training set under method m
 - 8: Evaluate on the test data through *Acc*, *Prec*, *Rec*, *F1*, *ROC-AUC*, and *PR-AUC* using Eqs. (5)–(10)
 - 9: **end for**
 - 10: **end for**
-

Evaluation Metrics

There are various measures for comparing the performance of classification models. Different metrics are commonly used because each metric captures different characteristics of model performance and provides a more comprehensive comparison⁵⁷. Most threshold-based measures are derived from confusion matrix; where column represent the instances of predicted classes and row represent the observed instances of actual classes. *Acc* is one of the most widely used classification evaluation metrics, as it measures the proportion of correctly predicted instances among all observations. However, *Acc* can be misleading when class distribution is uneven, because the performance on the majority class may dominate the overall results. Therefore, precision (*Prec*), *Rec*, *F1* were also used with *Acc* in this study to provide a more balanced evaluation of classification performance. The threshold-based metrics are formulated in Eqs. (5)–(8).

$$Acc = \frac{TP + TN}{TP + TN + FP + FN} \quad (5)$$

$$Prec = \frac{TP}{TP + FP} \quad (6)$$

$$Rec = \frac{TP}{TP + FN} \quad (7)$$

$$F1 = 2 \times \frac{Prec \times Rec}{Prec + Rec} \quad (8)$$

where TP, TN, FP , and FN denote the true positives, true negatives, false positives, and false negatives, respectively. In this study, the positive class represents successful problem solving within the next K attempts. In addition, threshold independent metrics, namely receiver operating characteristic area under the curve ($ROC-AUC$), and $PR-AUC$ were also used to evaluate the performance across decision thresholds. $ROC-AUC$ measures the area under the receiver operating characteristic curve based on true positive rate (TPR) and false positive rate (FPR), where TPR is equivalent to Rec and $FPR = \frac{FP}{FP+TN}$. $ROC-AUC$ is defined as:

$$ROC-AUC = \int_0^1 TPR(FPR)d(FPR) \quad (9)$$

$PR-AUC$ measures the area under the precision-recall curve and summarizes the trade-off between $Prec$ and Rec , and is defined as:

$$PR-AUC = \int_0^1 Prec(Rec)d(Rec) \quad (10)$$

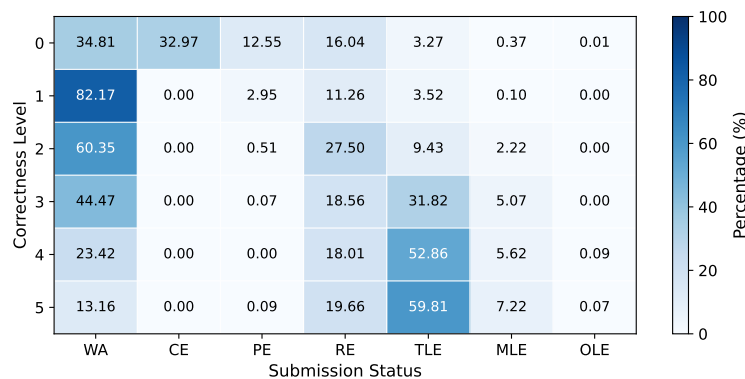
$PR-AUC$ was used as the primary metric because the prediction task in this study focuses on successful problem solving prediction, and positive-rate-based evaluation is informative when positive-class prediction quality is important⁵⁸. The other metrics were included for complementary evaluation.

Experimental Results

This section presents the submission distribution and learner behavior across the two courses. The comparative performance of the ML models is then reported across three formulations. Finally, risk-group based statistical analysis is presented to interpret the predictive outcomes.

Submission Distribution and Behavior

Submission patterns differed across the two courses in terms of solved and not-solved cases. As shown in Table 2, ALDS1 had higher average (avg.) submissions per user and per problem for not-solved cases, for example, the avg. submissions per user were 17.91 for solved cases and 25.82 for not-solved cases. This indicates that not-solved cases were associated with more repeated attempts before learners stopped or did not reach acceptance within the next five attempts. In contrast, CGL showed an opposite submission pattern, suggesting that learner interaction behavior varied across course contents. Fig. 2 further showed that the solved proportion was lower than the not-solved proportion in ALDS1 (solved vs not-solved: 0.43 vs 0.57), whereas CGL showed the opposite trend. These course-level differences indicate that ALDS1 and CGL provide contrasting submission-data contexts for evaluating the prediction formulations.



Chi-square Test: $\chi^2(30) = 155765.97$, $p = 0.000$, Cramér's $V = 0.32$; H_0 : submission status and correctness level are independent.

Figure 3. Submission behavior across correctness levels on ALDS1.

Fig. 3 presents the distribution of non-accepted submission statuses across correctness levels in ALDS1. At level 0, where no partial correctness was observed, most attempts received for WA, followed by CE, RE, and PE; whereas resource limitation errors (RLE) such as TLE, MLE, and OLE were negligible. For levels 1-5, a clear progression pattern was observed: the proportion of WA decreased substantially ($82.17\% \rightarrow 13.16\%$), whereas RLE increased at most cases, with TLE rising from 3.52% to 59.81% , and MLE from 0.10% to 7.22% . In addition, most WA submissions occurred at lower partial-correctness levels, particularly levels 1 and 2, whereas most TLE submissions appeared at higher levels, especially levels 4 and 5. Other statuses, such as RE, remained relatively stable, while CE, PE, and OLE accounted for only small proportions across levels 1–5. The Chi-square test further confirmed a significant association between submission status and correctness level ($p < 0.01$), with Cramér’s $V = 0.32$ indicating a moderate effect size. This suggests meaningful correctness-status variation across submissions and supports the inclusion of correctness information in prediction tasks.

Predictive Performance Comparison

Comprehensive experiments were conducted to identify the most effective predictive formulation across ML models. The CV-level validation results were obtained using the selected hyperparameter configurations for each classifier and formulation from the five-fold GroupKFold procedure described in [Validation Protocol and Hyperparameter Settings](#) section. Fig. 4 reports the mean $PR-AUC$ and SD across validation folds for each classifier under M_1 - M_3 on the ALDS1 training set. Overall, M_1 and M_2 showed comparable validation performance, while M_3 consistently achieved higher $PR-AUC$ values across classifiers. For example, for LR, the average $PR-AUC$ increased from 0.539 ± 0.123 (M_1) $\rightarrow$ 0.540 ± 0.123 (M_2) $\rightarrow$ 0.588 ± 0.096 (M_3), indicating improved validation performance in M_3 and lower fold-to-fold variability. Among the classifiers, RF achieved the highest mean $PR-AUC$ across all three formulations; outperforming the other models by relative improvements of approximately $0.18\% - 7.75\%$ under M_1 , $0.18\% - 7.69\%$ under M_2 , and $0.31\% - 9.35\%$ under M_3 . In contrast, LR and KNN showed comparatively weaker performance, whereas XGB and MLP showcased competitive performance.

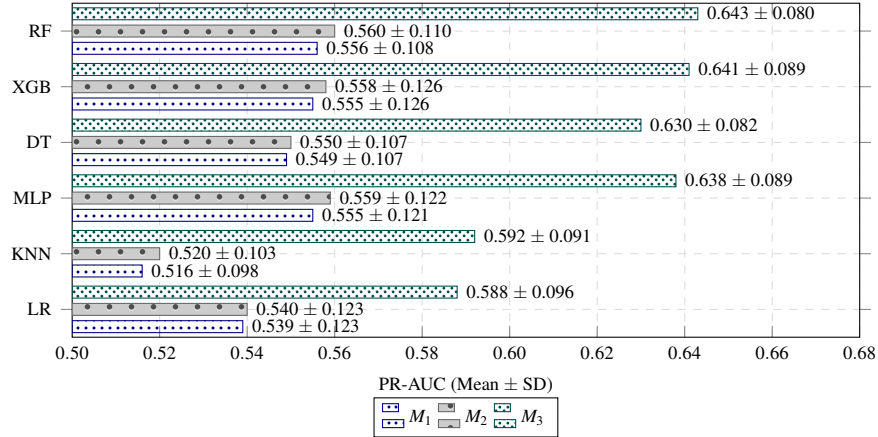


Figure 4. Five-fold CV performance across formulations on ALDS1.

Beyond the CV-level trends, Wilcoxon signed-rank tests⁵⁹ were applied to further examine pairwise differences among method-classifier combinations based on fold-wise $PR-AUC$ values. As shown in Fig. 5, the results generally align with the CV patterns, with M_3 -based models showing stronger validation performance than M_1 and M_2 , particularly when combined with RF. RF-based combinations produced the most consistent statistically favorable comparisons across formulations, whereas several other configurations’ comparisons were not statistically significant. In particular, comparison involving LR and KNN under M_1 , KNN under M_2 showed no significant differences from other method-classifier settings. These results provide additional fold-wise evidence for the advantage of the M_3 formulation.

Following CV-based parameter and model selections, Table 5 reports the held-out ALDS1 test performance of all method-classifier combinations using Acc , $Prec$, Rec , $F1$, and $PR-AUC$, while Fig. 6a visualizes the $PR-AUC$ (%) across M_1 - M_3 . Under M_1 , RF obtained the highest Acc (0.6604), $Prec$ (0.6999), and $PR-AUC$ (0.7099), whereas LR achieved the highest Rec (0.6163) and MLP achieved the highest $F1$ (0.6261). Under M_2 , MLP yielded the best Acc (0.6619), and $Prec$ (0.7026), while LR obtained the highest Rec (0.6168) and $F1$ (0.6205). Nevertheless, RF remained the stronger model in terms of $PR-AUC$ (0.7121). Under M_3 , RF achieved the highest Acc (0.7042), $Prec$ (0.7114), and $PR-AUC$ (0.7539), whereas the highest Rec (0.6905) and $F1$ (0.6924) were exhibited in MLP and XGB, respectively. Across classifiers, M_3 produced the top $PR-AUC$ values over M_1 and M_2 , whereas the difference between M_1 and M_2 was relatively small. This pattern suggests that incorporating partial-correctness and progress-aware information yields better predictions beyond binary and ordinal outcome representations.

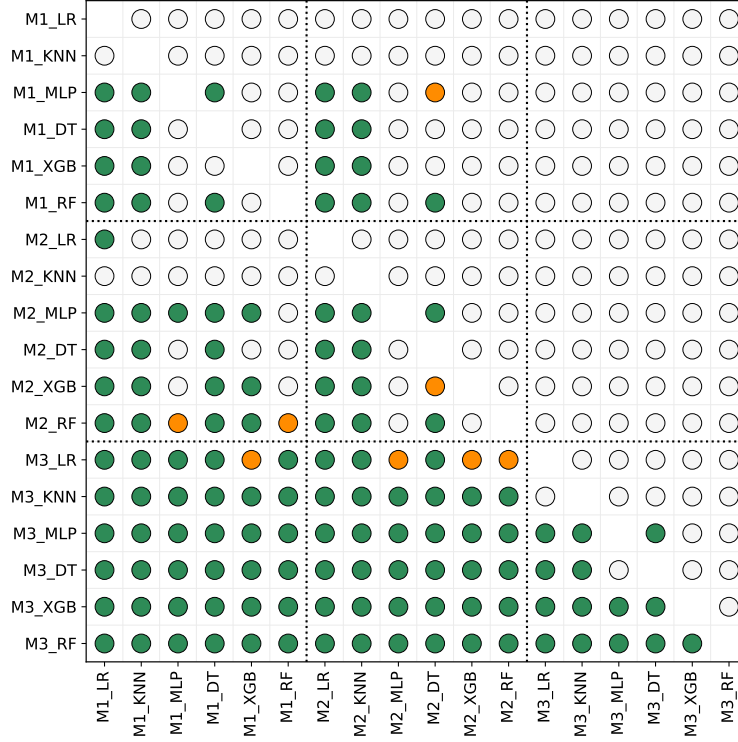


Figure 5. Wilcoxon signed-rank dot plot of five-fold CV PR-AUC across classifiers on ALDS1. Green and orange indicates the statistically significant improvements of the row model over the column model at $p < 0.05$ and $0.05 \leq p < 0.10$, respectively, whereas light gray indicates no significant difference ($p \geq 0.10$). Dotted lines separate M_1 - M_3 .

For example, XGB gradually improved from M_1 to M_3 (70.76% $\rightarrow$ 71.00% $\rightarrow$ 75.23%). The highest PR-AUC was obtained by RF under M_3 , reaching approximately 75.39%, and exceeding the other classifiers in M_3 by about 0.16-4.50 percentage-points. Therefore, the M_3 formulation also provided the strongest held-out test performance.

Classifier	Binary outcome					Ordinal correctness					Partial-correctness & progress-aware				
	Acc	Prec	Rec	F1	PR-AUC	Acc	Prec	Rec	F1	PR-AUC	Acc	Prec	Rec	F1	PR-AUC
LR	0.6301	0.6242	0.6163	0.6202	0.6937	0.6301	0.6242	0.6168	0.6205	0.6939	0.6677	0.6596	0.6654	0.6625	0.7089
KNN	0.6196	0.6180	0.5861	0.6017	0.6503	0.6252	0.6343	0.5557	0.5924	0.6603	0.6810	0.6868	0.6420	0.6636	0.7136
MLP	0.6492	0.6555	0.5992	0.6261	0.7056	0.6619	0.7026	0.5380	0.6094	0.7102	0.6990	0.6939	0.6905*	0.6922	0.7482
DT	0.6563	0.6900	0.5427	0.6076	0.6993	0.6526	0.6797	0.5508	0.6085	0.6969	0.6995	0.7020	0.6723	0.6868	0.7429
XGB	0.6578	0.6863	0.5561	0.6144	0.7076	0.6597	0.6889	0.5575	0.6162	0.7100	0.7007	0.6976	0.6873	0.6924*	0.7523
RF	0.6604	0.6999	0.5378	0.6082	0.7099	0.6595	0.6940	0.5463	0.6113	0.7121	0.7042*	0.7114*	0.6671	0.6885	0.7539*

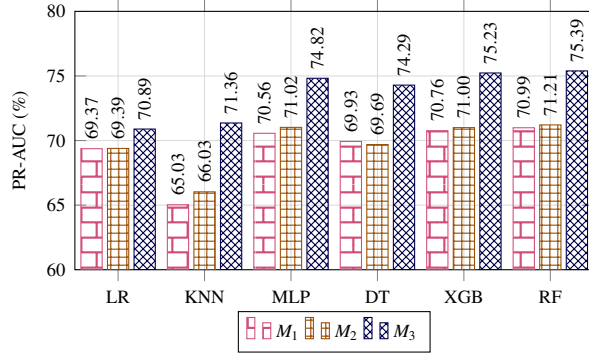
Bold values indicate the best result for each metric within each formulation, and * indicates the best results for each metric across all formulations.

Table 5. Performance comparison of classification algorithms on test data across formulations for the ALDS1.

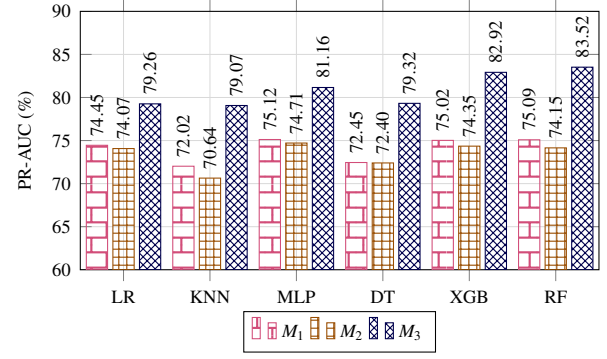
Fig. 7 presents the ROC curves with ROC-AUC values for all models across formulations. The ROC curves under M_3 were positioned closer to the upper left region of the plot than the others two, indicating a better balance between TP and FP rates. The ROC-AUC results followed a similar trend to PR-AUC, with M_3 generally producing higher values than M_1 and M_2 . For instance, MLP showed a clear increasing trend from M_1 to M_3 (0.6974 $\rightarrow$ 0.6988 $\rightarrow$ 0.7594). Among all combinations, RF under M_3 achieved the highest ROC-AUC value of 0.7628, while other classifiers ranged from 0.7164 to 0.7625. The confusion matrices in Fig. 8 further compare the RF models across formulations and show that M_3 identified solved cases more effectively than M_1 and M_2 . Overall, the held-out ALDS1 results support the effectiveness of the M_3 formulation, particularly when combined with RF, for predicting early success within the next K attempts.

Robustness Analysis

To examine the stability of the proposed method, additional experiments were conducted using cross-course evaluation on CGL and prediction-window (K) sensitivity analysis. For the cross-course evaluation, Table 6 shows that the CGL results were

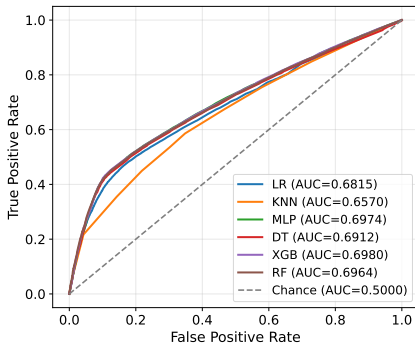


(a) ALDS1.

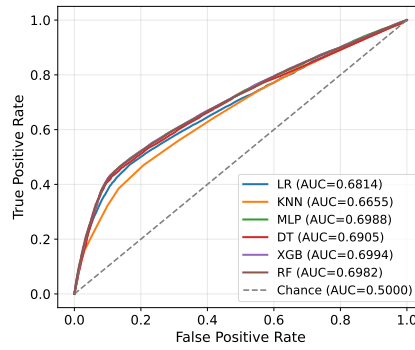


(b) CGL.

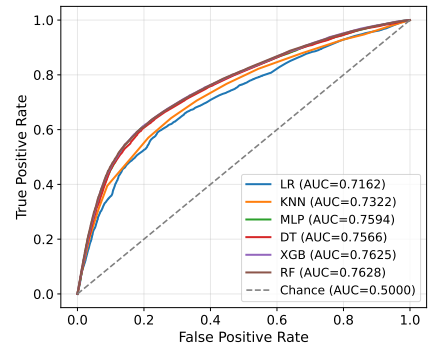
Figure 6. Comparison of PR-AUC values on the test data across classifiers and formulations.



(a) Binary outcome.

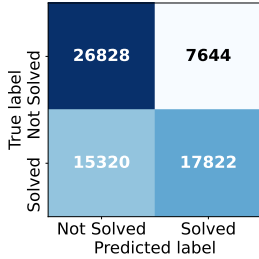


(b) Ordinal correctness.

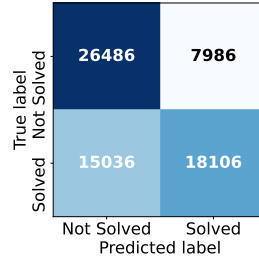


(c) Partial-correctness & progress-aware.

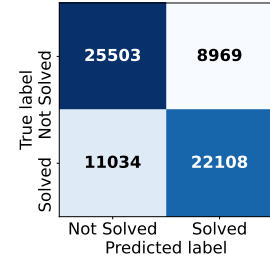
Figure 7. ROC curves on the held-out test sets for ALDS1 across formulations; AUC value is reported in the legend.



(a) M_1



(b) M_2



(c) M_3

Figure 8. Confusion matrices of RF for each formulation on the ALDS1 test set.

broadly consistent with the ALDS1 findings, with M_3 achieving stronger performance than M_1 and M_2 across the evaluation metrics. For instance, MLP demonstrated a progressive increase in *Acc* from M_1 to M_3 ($0.6056 \rightarrow 0.6148 \rightarrow 0.6868$). Although KNN and MLP showed comparatively better results under M_1 and M_2 . But under M_3 , RF produced the highest *PR-AUC*, *ROC-AUC*, and *Rec*, indicating consistently stronger performance, while XGB yielded more competitive results, achieving the highest *Acc*, *Prec*, and *F1*. Specifically, RF achieved the highest PR-AUC under M_3 , reaching 83.52% as compared with other M_3 classifiers, which ranged from 79.07% to 82.92% (Fig. 6b). These findings indicate that the advantage of the M_3 formulation was also observed on the additional CGL course, although leading classifier varied across evaluation metrics.

For the K sensitivity analysis, LR and RF were evaluated on ALDS1 test set using $K = 3, 5$, and 7 , where LR served as baseline and RF represented the best-performing model. The positive-rate baseline denotes the proportion of solved cases for each K setting. As shown in Fig. 9, RF remained clearly above the baseline across all K settings, whereas LR showed a larger gain at $K = 5$, followed by $K = 3$, and was close to baseline at $K = 7$. Across both classifiers, M_1 and M_2 produced

Classifier	Binary outcome						Ordinal correctness						Partial-correctness & progress-aware					
	Acc	Prec	Rec	F1	ROC-AUC	PR-AUC	Acc	Prec	Rec	F1	ROC-AUC	PR-AUC	Acc	Prec	Rec	F1	ROC-AUC	PR-AUC
LR	0.5522	0.7714	0.3862	0.5147	0.6388	0.7445	0.5673	0.7187	0.4868	0.5804	0.6367	0.7407	0.6539	0.7334	0.6868	0.7093	0.7108	0.7926
KNN	0.6121	0.6701	0.7270	0.6974	0.6271	0.7202	0.5889	0.6594	0.6855	0.6722	0.6021	0.7064	0.6763	0.7264	0.7597	0.7427	0.7268	0.7907
MLP	0.6056	0.7457	0.5440	0.6291	0.6636	0.7512	0.6148	0.7171	0.6170	0.6633	0.6543	0.7471	0.6868	0.7526	0.7308	0.7415	0.7459	0.8116
DT	0.5673	0.7175	0.4887	0.5814	0.6262	0.7245	0.5746	0.7015	0.5365	0.6080	0.6172	0.7240	0.6922	0.7768	0.7006	0.7368	0.7353	0.7932
XGB	0.5777	0.7456	0.4755	0.5806	0.6485	0.7502	0.5909	0.7206	0.5465	0.6216	0.6413	0.7435	0.7189*	0.7786*	0.7585	0.7684*	0.7725	0.8292
RF	0.5913	0.7234	0.5428	0.6202	0.6467	0.7509	0.6071	0.7129	0.6044	0.6542	0.6410	0.7415	0.7119	0.7649	0.7673*	0.7661	0.7761*	0.8352*

Bold values indicate the best result for each metric within each formulation, and * indicates the best results for each metric across all formulations.

Table 6. Performance comparison of classification algorithms on test data across formulations for the CGL.

nearly identical $PR-AUC$ values across K , whereas M_3 demonstrated a larger separation from the baseline, with the largest gain observed at $K = 5$, indicating its effectiveness across different prediction windows.

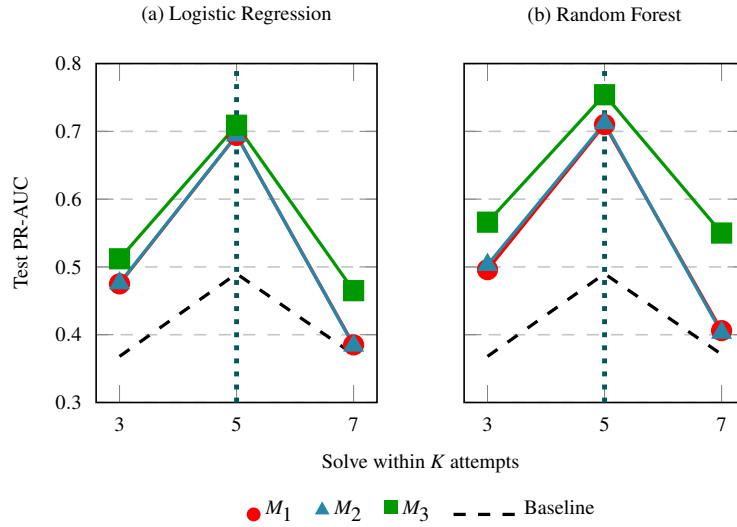


Figure 9. Stability of PR-AUC on test-set across K for LR and RF on ALDS1. The teal dotted vertical line marks the primary setting $K = 5$.

Feature-Level Statistical Validation

Logistic generalized linear model (GLM)⁶⁰ with binomial family and logit link was fitted to provide statistical validation of the M_3 formulation and to examine the association between its features and early success outcomes. The M_3 feature set served as predictors, while early success within the next five attempts was used as the binary outcome variable; continuous predictors follow the same transformations and standardization described in [Dataset and Preprocessing](#) section. Two-way clustered standard errors (SEs) at the user and problem levels were used to account for dependency among learner-problem observations⁶¹. Statistical significance was assessed using Wald z -tests based on the clustered SEs.

As reported in Table 7, several M_3 features showed statistically significant associations with early success at 1% level. Peak correctness showed the strongest positive association ($OR = 1.694$, 95% $CI : 1.595 - 1.799$, $p < 0.001$), indicating that a one-unit increase in peak correctness was associated with approximately 69.4% higher odds of solving the problem within the next five attempts, holding other predictors constant. The log-transformed time gap also showed a positive association ($OR = 1.191$, 95% $CI : 1.144 - 1.241$, $p < 0.001$) and exhibited higher odds for near-term success. Correctness (current) and correctness gain also exhibited significant associations. For submission-status indicators, PE showed higher odds than the reference category (WA) ($OR = 1.282$, $p = 0.003$), whereas CE, RE, TLE, and MLE showed lower odds; OLE was not statistically significant. Overall, GLM results provide statistical support for the association of the M_3 features with early success.

Risk-Group Analysis

To support interpretable use of the predicted outcomes, the predicted success probabilities were converted into three risk groups. The RF model first estimated the probability of solving a problem within the next K attempts (p_i), and tertile-based thresholds

Feature	Coef	OR	CI of OR	Wald Stat	P-value
Intercept	-0.121	0.886	0.754–1.040	-1.482	0.138
Log Attempt Index	0.055	1.057	0.929–1.203	0.838	0.402
Log Time Gap	0.175	1.191	1.144–1.241	8.375	0.000*
Correctness (Current)	-0.118	0.889	0.843–0.936	-4.445	0.000*
Correctness Gain	-0.103	0.902	0.879–0.926	-7.663	0.000*
Peak Correctness	0.527	1.694	1.595–1.799	17.148	0.000*
CE	-0.325	0.722	0.652–0.800	-6.200	0.000*
PE	0.248	1.282	1.090–1.506	3.011	0.003*
RE	-0.354	0.702	0.644–0.765	-8.025	0.000*
TLE	-0.353	0.703	0.613–0.805	-5.096	0.000*
MLE	-0.650	0.522	0.434–0.629	-6.868	0.000*
OLE	0.038	1.039	0.695–1.552	0.186	0.853

Coef = Coefficient (log-odds), OR = Odds ratio, CI = Confidence interval. * indicate predictor with statistically significant coefficients at the 1% level of significance. WA is the reference category for the submission-status indicators.

Table 7. Logistic GLM results with two-way clustering by user and problem on ALDS1.

were then computed from out-of-fold training predictions to assign each learner-problem instance into *high-risk*, *moderate-risk*, or *low-risk* groups. The first threshold (t_1) corresponds to the 33.3rd percentile, and the second threshold (t_2) corresponds to the 66.7th percentile of p_i . Instances with $p_i < t_1$ were categorized as *high-risk*, those with $t_1 \leq p_i < t_2$ as *moderate-risk*, and those with $p_i \geq t_2$ as *low-risk*.

Table 8 summarizes the mean observed and predicted success rates across risk groups for M_1 - M_3 , along with risk gaps among the groups. The mean observed and predicted success rates were closely aligned within each group, indicating that the estimated probabilities reflected the empirical group-level success patterns across M_1 - M_3 . Across all formulations, the expected monotonic pattern was observed: the *low-risk* group had the highest observed and predicted success rates, the *high-risk* group had the lowest, and the *moderate-risk* group remained between them. For example, under M_3 , the observed success rate decreased from 71.20% in the *low-risk* group to 36.25% in the *moderate-risk* group and 24.06% in the *high-risk* group. The predicted success rates showed a similar decreasing pattern from *low-risk* to *high-risk* groups (74.00 $\rightarrow$ 43.06 $\rightarrow$ 32.04). M_3 also produced the widest separation between *low-risk* and *high-risk* groups, with an observed risk gap of 47.14 percentage points and a 41.96-percentage-point predicted gap, compared with 33.51 and 33.83 percentage points observed gap and 29.88 and 29.86 percentage points predicted gap for M_1 and M_2 . These result indicate that M_3 produced clearer risk-group stratification than M_1 and M_2 , further supporting the classification results through better separation of learner-problem instances with different near-term success likelihoods.

Method	Observed Success				Predicted Success			
	Low	Moderate	High	Risk Gap	Low	Moderate	High	Risk Gap
M_1	66.87	38.39	33.36	33.51	69.11	45.51	39.23	29.88
M_2	66.84	38.77	33.01	33.83	69.06	45.64	39.20	29.86
M_3	71.20	36.25	24.06	47.14	74.00	43.06	32.04	41.96

Risk Gap is computed as the difference between the low-risk and high-risk values of each method.

Highlighted values indicate the strongest risk-group separation across methods.

Table 8. Summary statistics of risk groups across three formulations .

Fig. 10 further illustrates the distribution and attempt-wise behavior of the three risk groups. In Fig. 10a, submission-level risk was obtained directly from the instance-level predictions. For the user- and problem-level summaries, each user or problem was represented by the risk group that appeared most frequently among its associated submissions. For example, if a problem such as *ALDS1_2_D* contained the largest proportion of *high-risk* instances (55.18%), followed by *moderate-risk* (25.26%) and *low-risk* (19.56%), it was summarized as a *high-risk* problem. The submission distribution was relatively balanced across the groups, with the *low-risk* group accounting for a slightly larger share (36.57%). At the user level, the *moderate-risk* group contained the largest proportion, accounting for more than half of the users, followed by the *high-risk* (24.80%) and *low-risk* (17.41%) groups. In contrast, more than half of the problems were categorized as *low-risk*, followed by *high-risk* (22.41%) and *moderate-risk* (18.97%).

Fig. 10b further compares submission volume and success rate across attempt intervals. Across three risk groups, most submissions were concentrated in the early attempt ranges, particularly between 1 and 10 attempts. The *high-risk* group

showed consistently lower success rates across most attempt intervals despite substantial submission volumes, especially in the early to middle attempt intervals (1-15). Its success rate also tended to decline in later intervals, indicating that many *high-risk* learner-problem instances involved repeated attempts with limited near-term success. In contrast, the *low-risk* group maintained higher success rates across most intervals and showed generally a favorable success pattern as attempts increased. The *moderate-risk* groups showed intermediate but less stable behavior, with most submissions received in the early ranges. Overall, these patterns suggest that the risk groups captured distinct attempt-wise learner-problem-solving profiles.

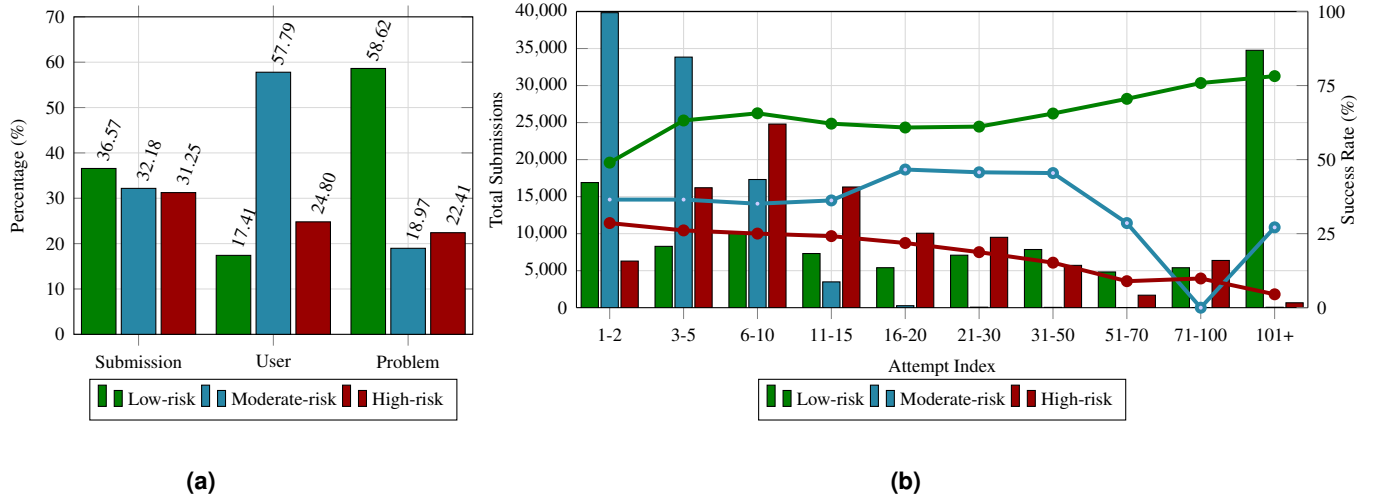


Figure 10. Risk-group distribution and attempt-wise behavior on ALDS1: (a) risk-group proportions across different characteristics, and (b) comparison of submissions volume (bars) and success proportion (lines) over attempt numbers by risk groups.

Discussion

This section discusses the major findings in relation to predictive performance, statistical validation, and supporting behavioral analyses. It further highlights the educational implications of the proposed approach and outlines the major limitations of the study.

Performance Analysis

This paper investigates how different behavioral representations affect early success prediction by evaluating three task formulations using AOJ submission features across various ML models. The contrasting submission and outcome patterns observed in ALDS1 and CGL (Fig. 2), together with the required preprocessing steps, provided the empirical context for evaluating the prediction formulations. Overall, M_3 formulation offered a more effective representation, achieving stronger CV, and held-out ALDS1 performance across all ML models than M_1 and M_2 (Fig. 4, and Table 5). M_3 also showed lower fold-to-fold variability, with SD ranges of M_3 : 0.080 – 0.096 compared with 0.098-0.126 for M_1 and 0.103-0.126 for M_2 , indicating more stable performance across user-disjoint validation folds. These findings suggest that partial-correctness and progress-aware features better represent learners’ gradual progress in OJ-based programming tasks, where learners often improve through repeated submissions before reaching the final accepted solution.

At the classifier level, RF produced the most consistent predictive results, particularly under M_3 . As reported in Table 5, and Fig. 7, within each formulation, RF obtained the highest *PR-AUC* on the held-out ALDS1 test set, and under M_3 , it also produced the best *Acc*, *Prec*, and *ROC-AUC* among ML models. Among all formulation-classifier combinations, RF- M_3 , achieved the highest *Acc*, *Prec*, *ROC-AUC* and *PR-AUC* values. For example, RF- M_3 obtained a *PR-AUC* of 75.39% compared with 70.89%-75.23% for other classifiers, and a *Prec* of 71.14% compared with 65.96%-70.20%. Therefore, RF may have benefited from the richer learner-problem representation in M_3 , which combines attempt behavior, time gaps, submission outcomes, and partial-score progression, allowing the model to capture feature interactions more effectively than simpler linear or instance-based classifiers. This observation is consistent with prior educational prediction studies in which RF showed strong outcome in student outcome prediction tasks^{13, 14}.

The statistical and risk-based analyses further strengthen the interpretation of the comparative results. The fold-wise statistical comparison showed that M_3 generally produced more favorable validation outcomes than the M_1 and M_2 , while RF

showed stronger classifier-level performance (Fig. 5). As shown in Table 7, the logistic GLM also revealed that progress-aware features, such as peak correctness and temporal behavior were meaningfully associated with early success, supporting the relevance of the M_3 feature design. In addition, the VIF results did not indicate severe multicollinearity among the numerical predictors (Table 3), supporting their distinct contribution to modeling and in the M_3 . The risk-group analysis revealed better distinction among the risk groups under M_3 , with a larger *low-to-high* risk gap (Table 8), suggesting that RF- M_3 predicted probabilities were useful not only for classification but also for distinguishing different levels of near-term success likelihood. Overall, the analysis suggests that the improvement was achieved by combining the partial-correctness and progress-aware representation with an effective learning model.

Educational Implications

The proposed formulation has practical implications for OJ-based programming education because it represents learners' intermediate progress through partial correctness, temporal behavior, and repeated-attempt progression. Its stronger predictive performance and statistical validation indicate that these signals can offer useful insights into learners' evolving problem-solving strategies, capturing not only final success but also how individuals progress across attempts toward near-term success or continued struggle. Repeated submissions often reflect debugging, error correction, and gradual problem-solving refinement. The risk-group-based analysis provides an interpretable link between model outputs and educational decision-making. The clearer *low*-, *moderate*-, and *high-risk* separation under M_3 , together with larger *low-high* risk gap reported in Table 8, indicates that M_3 can distinguish learner-problem instances with different levels of near-term success likelihood. Since learners and problems varied across risk groups (Fig. 10a), and risk groups are assigned at the learner-problem-attempt level, rather than being fixed labels for individual learners or problems, a learner may require support for one problem while progressing well on another, making the proposed formulation suitable for fine-grained monitoring during problem solving.

From an instructional perspective, the lower success pattern observed for *high-risk* instances across attempt intervals, compared with the more favorable trajectories of *low-risk* instances in Fig. 10b, suggests that the model may help identify cases where learner may need support before accumulating many unsuccessful submissions. The *high-risk* group may be prioritized for timely hints, easier related problems, or instructor feedback. The *moderate-risk* group may represent learners who are progressing but still need guidance and continued practice, while the *low-risk* group may be encouraged to attempt more challenging tasks. From an instructor-support perspective, aggregated risk-group profiles may help instructors identify learners who need timely guidance and monitor where support should be prioritized during programming practice. At the problem-set level, problems that repeatedly appear in *high-risk* profiles may indicate that the need for clearer statements, prerequisite review, additional examples, or revised instructional sequencing. Therefore, the proposed approach can transform OJ submission data into actionable information for learner support, and instructional monitoring and problem-set refinement.

Scalability and Generalizability

The evaluation of CGL alongside ALDS1 provides evidence of course-level robustness under different data conditions; ALDS1 represents a large dataset with 58 problems, while CGL is relatively smaller with 25 problems and different solved/not-solved submission patterns. Despite these differences, M_3 broadly maintained stronger performance than M_1 and M_2 (Table 6, and Fig. 6b), suggesting that the advantages of modeling partial-correctness and progression was not limited to the main ALDS1 dataset. In addition, the stable results of M_3 , especially when combined with RF, across $K = 3, 5$, and 7 , highlights its potential for both shorter and longer prediction windows (Fig. 9). This is important because learner may require different number of attempts before reaching a correct solution depending on their proficiency level, prior progress and problem complexity across datasets.

The M_3 formulation relies on interaction-based submission records including verdicts, attempt histories, time gaps and partial-correctness information. Since these types of sequential learner-task interaction data are commonly available in programming education environments, especially platforms that record submission outcomes and test-case-level progress. Therefore, the formulation may be adapted to other OJ platforms, programming MOOCs, automated assessment systems, and academic learning analytics environments without substantial structural changes. It may also support in early-warning systems, progress tracking, and personalized problem recommendation. More broadly, the same partial-correctness and progress-aware concepts may be applicable to other prediction tasks where sequential behavior and repeated-attempt patterns are available such as text-response prediction, dropout-risk prediction, and academic performance prediction.

Limitations and Future Work

Although the proposed framework provided promising results for predicting early success and generating useful educational insights, its performance may vary across educational datasets with different characteristics. This study still has some limitations, as follows:

- The dataset was limited to two AOJ courses, where CGL was utilized to examine course-wise robustness. Additional validation using other OJ platforms, programming courses, and educational datasets would further support broader generalizability.

- Learner behavior in AOJ may differ from other programming learning environments due to diverse problem sets, course design, problem complexity, and learner populations.
- This study mainly focused on next-five-attempts prediction, although sensitivity analysis was conducted across three K values; the optimal prediction window may vary depending on dataset structure and learning context.
- The risk analysis used tertile-based predicted probabilities; therefore, the groups should be treated as decision-support indicators rather than fixed learner/problem labels, and their practical value requires intervention-based validation.
- This study used various ML models based on tabular behavioral and progress-aware features, which may not fully capture long-term sequence learning patterns.

In future work, we plan to validate the performance of the proposed framework using other OJ platforms, programming courses, or educational datasets. Future work may also incorporate additional sequence-based features with behavioral features, apply sequence-aware models and evaluate the risk-based feedback in real educational settings.

Conclusion

This paper presents an early-success prediction framework for predicting learners' near-term success within the next K attempts in an online programming environment. The framework uses AOJ submission logs to construct learner-problem prediction formulations and compares proposed partial-correctness and progress-aware formulation with binary-outcome and ordinal-correctness formulations across multiple ML models. The proposed formulations consistently showed stronger performance across all ML models than the other two, achieving higher evaluation and validation results. In particular, RF with the proposed formulation achieved a $PR-AUC$ of 75.39% and $ROC-AUC$ of 76.28%, compared with 70.99% and 69.64% for the binary-outcome formulation, and 71.21% and 69.82% for ordinal-correctness formulation, respectively. The additional course-based evaluation showed consistent performance trends and further supported the course-wise robustness of the formulation under different submission data patterns, despite the contrasting positive-rate distributions, especially in test set between ALDS1 and CGL.

In addition, the logistic GLM-based feature interpretation showed that progress-aware and partial-correctness variables were meaningfully associated with early success outcomes, supporting their inclusion in the proposed formulation. Beyond predictive performance, the framework provides decision-support insights for OJ-based programming education. The risk-profile analysis demonstrated that *high*-, *moderate*-, and *low-risk* groups reflected different levels of near-term success likelihood and problem-solving progression, with *high-risk* groups showing lower success rates and weaker attempt-based progression, thereby supporting interpretable monitoring of learner-problem interactions. Overall, the proposed framework offers an effective approach for early success prediction and provides actionable insights for supporting learners and instructors in OJ-based programming environments, with potential applicability to related programming learning contexts.

References

1. Messer, M., Brown, N. C., Kölling, M. & Shi, M. Automated grading and feedback tools for programming education: A systematic review. *ACM Transactions on Comput. Educ.* **24**, 1–43, DOI: <https://doi.org/10.1145/3636515> (2024).
2. Scherer, R., Siddiq, F. & Viveros, B. S. A meta-analysis of teaching and learning computer programming: Effective instructional approaches and conditions. *Comput. Hum. Behav.* **109**, 106349, DOI: <https://doi.org/10.1016/j.chb.2020.106349> (2020).
3. Watanobe, Y., Rahman, M. M., Matsumoto, T., Rage, U. K. & Ravikumar, P. Online judge system: Requirements, architecture, and experiences. *Int. J. Softw. Eng. Knowl. Eng.* **32**, 917–946, DOI: <https://doi.org/10.1142/S0218194022500346> (2022).
4. Wang, J., Lin, P., Tang, Z. & Chen, S. How problem difficulty and order influence programming education outcomes in online judge systems. *Heliyon* **9**, DOI: <https://doi.org/10.1016/j.heliyon.2023.e20947> (2023).
5. Chen, K. *et al.* Accoding: A graph-based dataset for online judge programming. *Sci. Data* **11**, 548, DOI: <https://doi.org/10.1038/s41597-024-03392-z> (2024).
6. Rahman, M. M., Watanobe, Y., Kiran, R. U., Thang, T. C. & Paik, I. Impact of practical skills on academic performance: A data-driven analysis. *IEEE Access* **9**, 139975–139993, DOI: [10.1109/ACCESS.2021.3119145](https://doi.org/10.1109/ACCESS.2021.3119145) (2021).
7. Mia, M. S., Watanobe, Y., Rahman, M. M., Amin, M. F. I. & Muepu, D. M. Attention-driven clustering of programming problem difficulty using statistical and machine learning techniques. *IEEE Access* **13**, 155425–155446, DOI: [10.1109/ACCESS.2025.3606125](https://doi.org/10.1109/ACCESS.2025.3606125) (2025).

8. Llanos, J., Bucheli, V. A. & Restrepo-Calle, F. Early prediction of student performance in cs1 programming courses. *PeerJ Comput. Sci.* **9**, e1655, DOI: <http://dx.doi.org/10.7717/peerj-cs.1655> (2023).
9. Gardner, J. & Brooks, C. Student success prediction in moocs. *User Model. User-Adapted Interact.* **28**, 127–203, DOI: <https://doi.org/10.1007/s11257-018-9203-z> (2018).
10. López-Zambrano, J., Torralbo, J. A. L. & Romero, C. Early prediction of student learning performance through data mining: A systematic review. *Psicothema* **33**, 456, DOI: [10.7334/psicothema2021.62](https://doi.org/10.7334/psicothema2021.62) (2021).
11. Yağcı, M. Educational data mining: prediction of students' academic performance using machine learning algorithms. *Smart Learn. Environ.* **9**, 11, DOI: <https://doi.org/10.1186/s40561-022-00192-z> (2022).
12. Jokhan, A., Sharma, B. & Singh, S. Early warning system as a predictor for student performance in higher education blended courses. *Stud. High. Educ.* **44**, 1900–1911, DOI: <https://doi.org/10.1080/03075079.2018.1466872> (2019).
13. Mastour, H., Dehghani, T., Moradi, E. & Eslami, S. Early prediction of medical students' performance in high-stakes examinations using machine learning approaches. *Heliyon* **9**, DOI: <https://doi.org/10.1016/j.heliyon.2023.e18248> (2023).
14. Alkan, B. B., Kuzucuk, S., Alkan, N. & Sinan, A. Using machine learning to predict student outcomes for early intervention and formative assessment. *Sci. Reports* **15**, 39797, DOI: <https://doi.org/10.1038/s41598-025-23409-w> (2025).
15. Rahman, M. M., Watanobe, Y., Matsumoto, T., Kiran, R. U. & Nakamura, K. Educational data mining to support programming learning using problem-solving data. *IEEE Access* **10**, 26186–26202, DOI: [10.1109/ACCESS.2022.3157288](https://doi.org/10.1109/ACCESS.2022.3157288) (2022).
16. Villar, A. & de Andrade, C. R. V. Supervised machine learning algorithms for predicting student dropout and academic success: a comparative study. *Discov. Artif. Intell.* **4**, 2, DOI: <https://doi.org/10.1007/s44163-023-00079-z> (2024).
17. Marbouti, F., Diefes-Dux, H. A. & Madhavan, K. Models for early prediction of at-risk students in a course using standards-based grading. *Comput. & Educ.* **103**, 1–15, DOI: [http://dx.doi.org/10.1016/j.compedu.2016.09.005](https://doi.org/10.1016/j.compedu.2016.09.005) (2016).
18. Alhazmi, E. & Sheneamer, A. Early predicting of students performance in higher education. *IEEE Access* **11**, 27579–27589, DOI: <https://doi.org/10.1109/ACCESS.2023.3250702> (2023).
19. Kala, A., Torkul, O., Yildiz, T. T. & Selvi, I. H. Early prediction of student performance in face-to-face education environments: a hybrid deep learning approach with xai techniques. *IEEE Access* **12**, 191635–191649, DOI: [10.1109/ACCESS.2024.3516816](https://doi.org/10.1109/ACCESS.2024.3516816) (2024).
20. Shaka, M., Carraro, D. & Brown, K. Error tracing in programming: A path to personalised feedback. In *Proceedings of the 19th Workshop on Innovative Use of NLP for Building Educational Applications (BEA 2024)*, 330–342 (2024).
21. Liu, N., Wang, Z., Baraniuk, R. & Lan, A. Open-ended knowledge tracing for computer science education. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, 3849–3862, DOI: <https://doi.org/10.18653/v1/2022.emnlp-main.254> (2022).
22. Wang, Y. & Heffernan, N. Extending knowledge tracing to allow partial credit: Using continuous versus binary nodes. In *International conference on artificial intelligence in education*, 181–188, DOI: https://doi.org/10.1007/978-3-642-39112-5_19 (Springer, 2013).
23. Aizu Online Judge. Aizu online judge api. <https://judge.u-aizu.ac.jp/onlinejudge/api.jsp> (2026). Accessed: 2026-04-09.
24. Herrmann, L. & Weigert, J. Ai-based prediction of academic success: Support for many, disadvantage for some? *Comput. Educ. Artif. Intell.* **7**, 100303, DOI: <https://doi.org/10.1016/j.caeai.2024.100303> (2024).
25. Skittou, M., Merrouchi, M. & Gadi, T. Development of an early warning system to support educational planning process by identifying at-risk students. *IEEE access* **12**, 2260–2271, DOI: [10.1109/ACCESS.2023.3348091](https://doi.org/10.1109/ACCESS.2023.3348091) (2023).
26. Adnan, M. *et al.* Predicting at-risk students at different percentages of course length for early intervention using machine learning models. *IEEE Access* **9**, 7519–7539, DOI: [10.1109/ACCESS.2021.3049446](https://doi.org/10.1109/ACCESS.2021.3049446) (2021).
27. Jang, Y., Choi, S., Jung, H. & Kim, H. Practical early prediction of students' performance using machine learning and explainable ai. *Educ. information technologies* **27**, 12855–12889, DOI: <https://doi.org/10.1007/s10639-022-11120-6> (2022).
28. Nabizadeh, A. H., Goncalves, D., Gama, S. & Jorge, J. Early prediction of students' final grades in a gamified course. *IEEE Transactions on Learn. Technol.* **15**, 311–325, DOI: [10.1109/TLT.2022.3170494](https://doi.org/10.1109/TLT.2022.3170494) (2022).
29. Alcaraz, R., Martínez-Rodrigo, A., Zangróniz, R. & Rieta, J. J. Early prediction of students at risk of failing a face-to-face course in power electronic systems. *IEEE Transactions on Learn. Technol.* **14**, 590–603, DOI: [10.1109/TLT.2021.3118279](https://doi.org/10.1109/TLT.2021.3118279) (2021).

30. Barata, G., Gama, S., Jorge, J. & Gonçalves, D. Early prediction of student profiles based on performance and gaming preferences. *IEEE Transactions on Learn. Technol.* **9**, 272–284, DOI: [10.1109/TLT.2016.2541664](https://doi.org/10.1109/TLT.2016.2541664) (2016).
31. Costa, E. B., Fonseca, B., Santana, M. A., de Araújo, F. F. & Rego, J. Evaluating the effectiveness of educational data mining techniques for early prediction of students' academic failure in introductory programming courses. *Comput. human behavior* **73**, 247–256, DOI: <https://doi.org/10.1016/j.chb.2017.01.047> (2017).
32. Zhang, V., Jeffries, B. & Koprinska, I. A machine learning approach for predicting student progress in online programming education. *Int. J. Artif. Intell. Educ.* **35**, 3614–3644, DOI: <https://doi.org/10.1007/s40593-025-00510-9> (2025).
33. Hoq, M., Brusilovsky, P. & Akram, B. Explaining explainability: Early performance prediction with student programming pattern profiling. *J. Educ. Data Min.* **16**, DOI: <https://doi.org/10.5281/zenodo.14246435> (2024).
34. Pires, J. P., Brito Correia, F., Gomes, A., Borges, A. R. & Bernardino, J. Predicting student performance in introductory programming courses. *Computers* **13**, 219, DOI: <https://doi.org/10.3390/computers13090219> (2024).
35. Mia, M. S., Watanobe, Y., Rahman, M. M., Amin, M. F. I. & Muepu, D. M. Predictive modeling of submissions and learning outcomes in online judge systems. In *2025 IEEE 18th International Symposium on Embedded Multicore/Many-core Systems-on-Chip (MCSoc)*, 483–489, DOI: [10.1109/MCSoc67473.2025.00081](https://doi.org/10.1109/MCSoc67473.2025.00081) (IEEE, 2025).
36. Amin, M. F. I., Watanobe, Y., Rahman, M. M., Muepu, D. M. & Mia, M. S. Error understanding in program code with llm-dl for multi-label classification. *arXiv preprint arXiv:2603.25005* DOI: <https://doi.org/10.48550/arXiv.2603.25005> (2026).
37. Rico-Juan, J. R., Sánchez-Cartagena, V. M., Valero-Mas, J. J. & Gallego, A. J. Identifying student profiles within online judge systems using explainable artificial intelligence. *IEEE Transactions on Learn. Technol.* **16**, 955–969, DOI: [10.1109/TLT.2023.3239110](https://doi.org/10.1109/TLT.2023.3239110) (2023).
38. Pereira, F. D. *et al.* Early dropout prediction for programming courses supported by online judges. In *International conference on artificial intelligence in education*, 67–72, DOI: https://doi.org/10.1007/978-3-030-23207-8_13 (Springer, 2019).
39. Corbett, A. T. & Anderson, J. R. Knowledge tracing: Modeling the acquisition of procedural knowledge. *User modeling user-adapted interaction* **4**, 253–278, DOI: <https://doi.org/10.1007/BF01099821> (1994).
40. Shen, S. *et al.* A survey of knowledge tracing: Models, variants, and applications. *IEEE Transactions on Learn. Technol.* **17**, 1858–1879, DOI: [10.1109/TLT.2024.3383325](https://doi.org/10.1109/TLT.2024.3383325) (2024).
41. Wang, H., Wu, Q., Bao, C., Ji, W. & Zhou, G. Research on knowledge tracing based on learner fatigue state. *Complex & Intell. Syst.* **11**, 226, DOI: <https://doi.org/10.1007/s40747-025-01831-x> (2025).
42. Gao, Z. *et al.* Tracing distinct learning trajectories in introductory programming course: a sequence analysis of score, engagement, and code metrics for novice computer science vs. math cohorts. *Int. J. STEM Educ.* **12**, 27, DOI: <https://doi.org/10.1186/s40594-025-00546-2> (2025).
43. Yusuf, A. & Noor, N. M. Modeling students' algorithmic thinking growth trajectories in different programming environments: an experimental test of the matthew and compensatory hypothesis. *Smart Learn. Environ.* **11**, 38, DOI: <https://doi.org/10.1186/s40561-024-00324-7> (2024).
44. Cox, D. R. The regression analysis of binary sequences. *J. Royal Stat. Soc. Ser. B: Stat. Methodol.* **20**, 215–232 (1958).
45. Cover, T. & Hart, P. Nearest neighbor pattern classification. *IEEE transactions on information theory* **13**, 21–27, DOI: [10.1109/TIT.1967.1053964](https://doi.org/10.1109/TIT.1967.1053964) (1967).
46. Rumelhart, D. E., Hinton, G. E. & Williams, R. J. Learning representations by back-propagating errors. *nature* **323**, 533–536, DOI: <https://doi.org/10.1038/323533a0> (1986).
47. Quinlan, J. R. Induction of decision trees. *Mach. learning* **1**, 81–106, DOI: <https://doi.org/10.1007/BF00116251> (1986).
48. Breiman, L. Random forests. *Mach. learning* **45**, 5–32, DOI: <https://doi.org/10.1023/A:1010933404324> (2001).
49. Chen, T. & Guestrin, C. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, 785–794, DOI: <https://doi.org/10.1145/2939672.2939785> (2016).
50. University of Aizu. Aizu Online Judge Developers. <http://developers.u-aizu.ac.jp/>. Accessed: 2026-04-29.
51. Aizu Online Judge. Algorithms and Data Structures I (ALDS1). <https://onlinejudge.u-aizu.ac.jp/courses/lesson/1/ALDS1/1>. Accessed: 2026-05-19.
52. Aizu Online Judge. Computational Geometry (CGL). <https://onlinejudge.u-aizu.ac.jp/courses/library/4/CGL/all>. Accessed: 2026-05-19.

53. West, R. M. Best practice in statistics: The use of log transformation. *Annals clinical biochemistry* **59**, 162–165, DOI: <https://doi.org/10.1177/00045632211050531> (2022).
54. Kim, J. H. Multicollinearity and misleading statistical results. *Korean journal anesthesiology* **72**, 558–569, DOI: <https://doi.org/10.4097/kja.19087> (2019).
55. Allgaier, J. & Pryss, R. Practical approaches in evaluating validation and biases of machine learning applied to mobile health studies. *Commun. Medicine* **4**, 76, DOI: <https://doi.org/10.1038/s43856-024-00468-0> (2024).
56. He, H. & Garcia, E. A. Learning from imbalanced data. *IEEE Transactions on knowledge data engineering* **21**, 1263–1284, DOI: [10.1109/TKDE.2008.239](https://doi.org/10.1109/TKDE.2008.239) (2009).
57. Sokolova, M. & Lapalme, G. A systematic analysis of performance measures for classification tasks. *Inf. processing & management* **45**, 427–437 (2009).
58. Saito, T. & Rehmsmeier, M. The precision-recall plot is more informative than the roc plot when evaluating binary classifiers on imbalanced datasets. *PloS one* **10**, e0118432, DOI: <https://doi.org/10.1371/journal.pone.0118432> (2015).
59. Wilcoxon, F. Individual comparisons by ranking methods. *Biom. bulletin* **1**, 80–83, DOI: <https://doi.org/10.2307/3001968> (1945).
60. Nelder, J. A. & Wedderburn, R. W. Generalized linear models. *J. Royal Stat. Soc. Ser. A: Stat. Soc.* **135**, 370–384 (1972).
61. Cameron, A. C., Gelbach, J. B. & Miller, D. L. Robust inference with multiway clustering. *J. Bus. & Econ. Stat.* **29**, 238–249, DOI: <https://doi.org/10.1198/jbes.2010.07136> (2011).

Author contributions statement

M.S.M.: Conceptualization, Methodology, Data Curation, Formal Analysis, Visualizations, Writing-original draft. Y.W.: Resources, Supervision, Project Administrator, Writing: review & editing. M.M.R.: Software, Validation, Writing-review & editing. M.F.I.A.: Software, Validation, Writing-review & editing. D.M.M.: Validation, Data Curation, Writing: review & editing.

Data availability statement

The data used in this study were obtained from the Aizu Online Judge system and are not publicly available; however, requests for access may be made through the developer portal of the AOJ system (<http://developers.u-aizu.ac.jp/index>).

Competing interests

The authors declare no competing interests.

Funding

This work was supported by Japan Society for the Promotion of Science (JSPS) KAKENHI under Grant 23H03508.