

Cross-Group Aligned Problem Difficulty Clustering Using Attention-Weighted Feature Learning

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

Fang Liu

University of Notre Dame

Article

Keywords: Programming learning, online judge systems, educational data mining, clustering algorithms, feature attention mechanism, global difficulty alignment, Non-Monotonic Ordering

Posted Date: August 26th, 2026

DOI: <https://doi.org/10.21203/rs.3.rs-9899773/v1>

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

[Read Full License](#)

Additional Declarations: No competing interests reported.

Cross-Group Aligned Problem Difficulty Clustering Using Attention-Weighted Feature Learning

Md. Shahajada Mia^{1,2,*}, Yutaka Watanobe¹, Md. Mostafizer Rahman³, Md Faizul Ibne Amin¹, Daniel M. Muepu¹, and Fang Liu^{3,4}

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

⁴Department of Applied and Computational Mathematics and Statistics, University of Notre Dame, IN, USA

*shahajadabrur@gmail.com

ABSTRACT

Developing programming skills is central to computer science education, and Online Judge (OJ) platforms have become a widely used infrastructure for large-scale practice beyond classroom-based learning. However, many OJs do not offer explicit difficulty levels, and the same problem can be perceived differently by users with different proficiency levels, often leading to lower success rates on challenging problems. To better capture these proficiency-dependent differences, this study proposes a proficiency-group-aware framework for problem difficulty categorization using submission-derived features together with Rasch-based latent estimates. The framework integrates feature-attention-guided K-means (FA-K-means), which learns group-specific feature weights to improve within-group difficulty clustering and a global alignment step to mitigate cross-group inconsistency measured by Non-Monotonic Ordering (NMO). We compare the proposed method with standard clustering baselines, including K-means, Fuzzy c-means, Ward's method, and BIRCH, using the silhouette score (SS), Davies–Bouldin index (DBI), Calinski–Harabasz index (CHI), and NMO. Experiments on large-scale Aizu Online Judge logs show that performance varies markedly across proficiency groups, and FA-K-means consistently achieves superior clustering quality across all proficiency groups according to SS, DBI, and CHI. The alignment step reduces NMO from 30.41% to 9.69%, yielding a more coherent difficulty scale across groups. Feature analysis indicates that acceptance rate and average attempts contribute most to separating difficulty levels. Overall, the proposed framework provides an effective and interpretable basis for group-aware difficulty categorization in OJ-based programming education and is complemented by a lightweight web-based visualization to support user-level inspection of the derived difficulty levels.

Keywords: Programming learning; online judge systems; educational data mining; clustering algorithms; feature attention mechanism; global difficulty alignment; Non-Monotonic Ordering

Introduction

With the rapid advancement of information and communication technologies, programming has emerged as a fundamental skill and is increasingly essential for students in computer science, engineering, and many related fields. Programming is now incorporated into curricula across education levels, from school to university, to help students acquire practical programming skills^{1–4}. Such practice supports students in developing computational thinking and problem-solving skills, including logical reasoning and algorithmic design^{5,6}. Despite this importance, learning to program can be challenging for many novice learners (e.g., first-year undergraduates) because programming skill is largely developed through repeated practice across diverse problems⁷. However, providing sufficient practice opportunities and timely feedback is challenging for instructors, especially in large classes⁸.

Accordingly, automated programming assessment (APA) systems are increasingly adopted to assess student performance and alleviate instructors' workload⁹. Among such APA environments, Online Judge (OJ) platforms serve an important role in supporting individualized programming practice with immediate feedback and fostering programming skills. These platforms provide a large number of programming problems covering diverse course topics and varying problem complexity¹⁰. Typically, users submit solution code for a specific problem, and the system automatically executes it and assesses correctness and efficiency¹¹. Several well-known OJ systems, such as Aizu Online Judge (AOJ)¹², UVa⁴, and Judge.org¹⁰, have been used in university-level programming education for programming practice and course assignments. Moreover, these platforms record large-scale submission logs, including submission outcomes, attempt counts, and performance indicators (e.g., accuracy), which

support data-driven research on problem difficulty and learner behavior.

A key challenge in OJ-based analysis is learner heterogeneity: performance and interaction patterns differ substantially across users. Higher proficiency learners typically solve more problems and are more willing to attempt challenging tasks, whereas lower proficiency learners benefit from a gradual progression from easier to harder problems to build confidence and maintain motivation^{13,14}. However, many OJ problem sets are often not differentiated in a proficiency-aware manner^{14,15}, making it difficult for novice learners to select problems that match their current capability, which can in practice lead to repeated failures and frustration. Consequently, the success and failure patterns for the same problems may differ across user proficiency groups, and a single unified label may be misleading. Moreover, the acceptance rates in several OJ systems are below or around 50%, suggesting that a substantial proportion of submissions are unsuccessful^{16,17}. These observations highlight the need for user-group-aware problem difficulty categorization to better support learners.

Unsupervised machine learning (ML) has been widely used across various domains, including educational data^{2,3,5,18}. However, relatively few studies have explored clustering-based difficulty categorization for OJ problems. Existing studies mainly rely on traditional clustering methods (e.g., K-means and Fuzzy c-means (FCM)) applied to problem-level or log-derived features to cluster OJ problems into difficulty levels^{16,17,19–21}. Since these approaches rely on globally aggregated behavioral features, they can hide how the same problem exhibits different success/failure and attempt patterns across user proficiency groups²². In addition, OJ features are often highly skewed and long-tailed (e.g., submissions and attempts)^{17,23}, which is also observed in programming behavior logs before normalization²⁴. Without robust preprocessing, distance-based clustering can be biased toward high-activity problems.

Attention mechanisms (AM) have improved clustering quality in other fields such as image and text data, by emphasizing informative feature dimensions and reducing redundancy^{25,26}. Similarly, for tabular data, feature-weighting methods learn per-feature weights rather than assuming uniform feature importance²⁷. Mia et al.¹⁷ demonstrated that combining a self-attention mechanism with K-means improves global OJ difficulty clustering; however, that approach applies feature transformation across all users without proficiency stratification and does not measure cross-group label inconsistency. As a result, a problem that appears easy for advanced users may still be difficult for beginners. Furthermore, independent group-wise clustering can produce difficulty labels that are inconsistent across user groups, which may disrupt clear learning progression. However, to the best of our knowledge, AM-based feature reweighting that is simultaneously group-specific and interpretable has rarely been incorporated into clustering pipelines for educational OJ difficulty categorization. This combination is important because the goal is not only to improve clustering quality, but also to produce difficulty labels that remain interpretable across learner groups.

Considering the above challenges, this work proposes a user-aware framework for clustering OJ problem difficulty consistently across proficiency groups, aiming to support effective learning progression for both students and instructors. The framework uses diverse OJ behavioral features, including count-based, proportional, latent, and time-based indicators, to capture proficiency-dependent problem interactions. It integrates attention-based feature reweighting with cross-group difficulty alignment, yielding difficulty levels that remain comparable across user groups while preserving interpretability. The major contributions of this study are as follows:

- We propose a user-proficiency-aware clustering framework for consistent categorization of OJ problem difficulty, in which difficulty is interpreted separately for three proficiency groups, capturing proficiency-dependent interaction patterns that population-level approaches obscure.
- We formalize Non-Monotonic Ordering (NMO) as an interpretable measure of cross-group difficulty-label inconsistency. NMO enables quantitative evaluation of whether group-wise Easy $\rightarrow$ Medium $\rightarrow$ Hard labels remain coherent across proficiency levels.
- The proposed framework applies feature-attention-guided K-means (FA-K-means) to learn group-specific adaptive feature weights rather than assuming uniform feature importance as in standard K-means. It further applies global difficulty alignment to produce a unified and interpretable difficulty scale across user groups.
- Experiments on large-scale AOJ logs demonstrate that the proposed framework improves clustering quality under SS $\uparrow$, DBI $\downarrow$, and CHI $\uparrow$, and reduces NMO from 30.41% to 9.69%, achieving lower inconsistency than K-means, FCM, Ward, and BIRCH. Attention weights analysis, validated through Kruskal-Wallis (KW) testing, identifies acceptance rate and average attempts as key indicators for proficiency-aware difficulty interpretation and problem organization in OJ-based programming education.

The rest of the paper is structured as follows: [Related Work](#) section reviews related work on OJ learning environments and clustering-based analyses, including ML and Rasch modeling. [Materials and Methods](#) section introduces the AOJ dataset, details the feature set, preprocessing steps, and presents the proposed framework. [Experimental Settings](#) section summarizes

the hyperparameters employed across clustering methods and the validation metrics. In [Experimental Result](#) section, we present the experimental results and provide a comparative analysis across user groups. [Discussion](#) section interprets the findings and discusses their implications. Finally, [Conclusion and Future Work](#) section concludes the paper by summarizing key findings and offering potential directions for future work.

Related Work

This section summarizes prior work on OJ-based educational analytics and on clustering and difficulty modeling relevant to our proposed approach.

OJ Platforms and Learning Analytics

OJ platforms are not only used for programming practice in education and competitive contests, but also for broader contexts such as recruitment/skill assessment and data-mining challenges, enabled by automated and objective evaluation²⁸. Prior work used OJ submission logs to analyze learners' behavior and performance across repeated attempts over time²¹. As a result, OJ data has become a key empirical resource for programming education research. In addition, large-scale code resources such as CodeNet²⁹ and competitive-programming benchmarks such as CodeContests (AlphaCode)³⁰ and APPS³¹ facilitate ML-based analysis of programming behavior. Consequently, many studies have leveraged OJ-derived data to understand learning behavior and develop recommendations or learner support systems^{2,3,5,6,14,17,19–24,32–36}.

Learning analytics (LA) and educational data mining (EDM) provide methods for studying students' learning and behavioral patterns from large-scale educational data³⁷. In OJ-based programming education, prior studies have applied LA/EDM to submission logs to derive actionable insights for learner progression and data-driven interventions^{6,22,24,34,35}. Accordingly, prior studies have emphasized early identification of learner categories, especially at-risk groups, to alleviate the high failure rate in introductory programming (CS1) contexts^{24,34}. Study²² presented student profiling from OJ submissions to interpret success/failure predictions using multi-instance learning and explainable artificial intelligence, enabling feedback for different learner groups, while³⁵ used a neural network-based SCFH model to group students into risky, intermediate, and advanced levels. In addition, VisOJ provided an LA dashboard for OJ systems that allowed both teachers and students to monitor learning status and analyze submission behaviors³⁶. Overall, these studies motivate profiling users before clustering programming problems, since user-problem interaction patterns can vary markedly across heterogeneous user groups.

Difficulty Estimation

Difficulty modeling aims to estimate problem complexity as a continuous score or categorical levels using user interactions and/or problem content. Such difficulty estimation supports both evaluative assessment and formative online learning settings^{38,39}. Interaction-based approaches such as Item Response Theory (IRT) jointly capture learner ability and item difficulty from response patterns^{40,41}. IRT includes the Rasch (one-parameter logistic; 1PL) model and more flexible variants (e.g., 2PL/3PL and multidimensional extensions), depending on measurement assumptions and response characteristics⁴¹. Recent work has integrated IRT with ML; for example, combining IRT with a conditional variational autoencoder has been used to construct datasets aligned with participants' ability levels⁴². In OJ systems, study²¹ applied the Rasch (1PL) model to derive latent features for categorizing users and programming problems, while another work leveraged multidimensional IRT and Elo-style modeling to estimate user skills⁴³. These latent scores provide interpretable and scalable signals for clustering proficiency and problem difficulty in large-scale OJ data.

Clustering Approaches

Difficulty levels are commonly defined through clustering, which partitions problems (or users) into discrete groups based on behavioral and latent features. Zhao et al.¹⁶ proposed a topic-aware difficulty modeling approach that learned topics from OJ submission traces and estimated topic-specific difficulty and user capability, indicating that difficulty distributions can vary substantially across topics. In OJ analytics, K-means is frequently used to categorize users and/or problems for downstream modeling. For example, study³ clustered users before applying an LSTM-based problem recommendation, while study²¹ clustered both users and problems using latent features, and reported that problem order and difficulty can influence learner outcomes. Study²⁰ also used K-means clustering to estimate some behavioral OJ features weights for problem recommendation. For explicit difficulty categorization, study¹⁹ clusters problems into difficulty levels using FCM and compared multiple clustering methods such as K-means, hierarchical clustering (Ward's and complete-linkage), and Balanced Iterative Reducing and Clustering using Hierarchies (BIRCH). Beyond OJ, related clustering-based analyses have been explored in CS1 and other educational settings, including topic modeling (Latent Dirichlet Allocation/ Non-negative Matrix Factorization) with a K-means baseline for semantic topic groupings of problems⁴⁴, and hierarchical clustering on code-derived skill features to identify student groups requiring additional support⁴⁵. Similar applications have also been reported in other educational datasets^{2,5,18}.

To enhance clustering quality under heterogeneous feature spaces, study⁴⁶ employed a K-means-based feature weighting approach and reported gains over equal weighting, validated using precision and recall against known class labels. Another study extended feature-weighted K-means with regularization to improve weight stability⁴⁷. AMs have also been incorporated into clustering pipelines to focus on informative structures; for instance,²⁵ combined AM with graph-based clustering for robustness, and²⁶ integrated AM into representation learning for short-text clustering to obtain more discriminative clusters. Peng et al.⁴⁸ proposed an attention-integrated graph clustering method that learns embeddings by attention-based fusion, and reported improved performance (e.g., higher accuracy and normalized mutual information) over strong baselines on most benchmarks. Mia et al.¹⁷ combined AM with K-means and evaluated the clustering quality using SS, CHI, and DBI, comparing against K-means and FCM for problem difficulty clustering by reducing global feature dimensions. In supervised tabular learning, AM has also been used for feature weighting; TabNet produced instance-wise feature selection patterns via sequential attention⁴⁹, while the transformer-based feature weighting approach learned feature weights and reported improved classification accuracy (e.g., Multi-Layer Perceptron: 0.868 to 0.894 after weighting)²⁷.

Research Gap and Positioning of the Study

Prior studies reviewed in [OJ Platforms and Learning Analytics](#) to [Clustering Approaches](#) sections demonstrate the usefulness of OJ logs for difficulty clustering. However, most existing approaches focus on population-level (global) clustering and do not explicitly address cross-group label consistency. Feature weighting and attention-based representations have been studied more extensively in other domains to enhance clustering or learned representations; however, applying attention-based feature reweighting for low-dimensional numerical indicators to clustering remains less explored than in unsupervised learning. To the best of our knowledge, no prior work has simultaneously addressed proficiency-group-specific difficulty modeling, formal quantification of cross-group label inconsistency, and systematic cross-group alignment in OJ analytics. Building on these foundations, we propose a user-aware difficulty modeling approach that supports cross-group label consistency while preserving clustering quality and interpretability on low-dimensional interaction features. Table 1 summarizes how the proposed framework differs from the closest prior studies across the key dimensions that define the research gap.

Table 1. Positioning of the proposed framework against the prior studies.

Study	Group-Aware	Attention/weighting Type	Per-group Feature Weights	Cross-Group Alignment	NMO Metric
Zhao et al. ¹⁶ ; Intisar & Watanobe ¹⁹ ; Wang et al. ²¹	No	None	No	No	No
Modha & Spangler ⁴⁶	No	Feature weighting	No	No	No
Mia et al. ¹⁷	No	Global self-attention	No	No	No
Our study	Yes	Group-specific feature attention	Yes	Yes	Yes

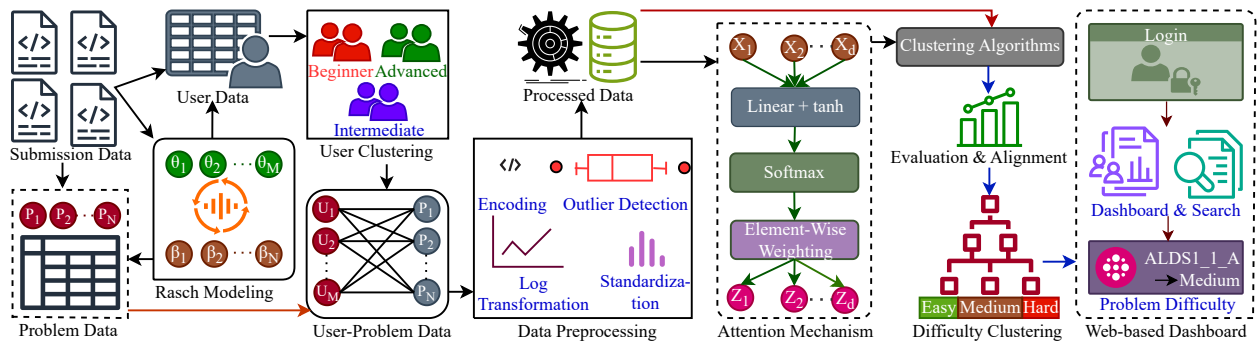


Figure 1. Overall workflow of user-aware problem difficulty clustering.

Materials and Methods

The proposed methodology for proficiency-group-aware problem difficulty categorization integrates statistical modeling, clustering, and attention-based learning to identify difficulty patterns of OJ platforms. As shown in Fig. 1, the overall workflow consists of five major steps. First, AOJ submission log data are collected, and user- and problem-based features are derived,

including latent features estimated via the Rasch model. Second, user clustering is performed using the K-means algorithm, after which user-problem interaction data are prepared and preprocessed. Third, problem difficulty levels are clustered for each user group using the proposed two-stage framework and evaluated against traditional algorithms using clustering validation metrics. Fourth, a comprehensive analysis combining graphical representations and statistical tests is conducted to examine data patterns and trends across user difficulty levels. Finally, a web-based visualization module is used to demonstrate how the derived difficulty labels can be inspected at the user level. The details of each step are described in the following subsections.

Dataset Description

The submission log dataset used in this study was obtained from the AOJ system, a large-scale programming platform with approximately 0.17 million users^{12,50}. The platform contains over 10 million submissions across over 3000 problems from programming courses and contests, and the source code and submission logs are available to researchers via Google Cloud Storage upon request⁵⁰. This study used secondary analysis of fully anonymized submission logs from the AOJ platform, with no access to personally identifiable information. Under the AOJ Terms of Use, this data may be utilized 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, The University of Aizu. All methods were carried out in accordance with the relevant guidelines and regulations. To obtain a better representation of the user-problem interaction set, only problems attempted more than 30 submissions, and users who submitted more than 10 submissions were included in the analysis. The resulting dataset comprised 9631901 submissions, 3047 problems, and 83495 users. Each AOJ submission record includes various attributes such as Problem ID, User ID, submission status as one of Accepted (AC), Wrong Answer (WA), Presentation Error (PE), Compile Error (CE), Time Limit Exceeded (TLE), Memory Limit Exceeded (MLE), or Output Limit Exceeded (OLE), accuracy, timestamps (Submission and Judge dates), using 18 programming languages including C, C++, Python, and Java^{2,50}.

Table 2. Description of extracted user- and problem-based features.

Feature	Type	Formula	Meaning and Rationale
Accuracy	User	$Acc_j = \frac{1}{ \mathcal{S}_j } \sum_{s \in \mathcal{S}_j} \frac{AC_s}{TC_s}$	Shows user performance, typically how many test cases are solved correctly across submissions, highlighting observed performance across various problems and distinguishing users by correctness behavior.
Ability Score	User	Rasch (θ_j) (Eq. 3)	Shows user proficiency estimated by the Rasch model, capturing latent ability across problems and distinguishing users by underlying ability over observed outcomes.
Submissions	Problem	$S_g(i) = \sum_{U_j \in U_g(i)} S_{ji}$	Represents how often problems are submitted by user groups, reflecting effort and popularity, while linking submission volumes to observed problem difficulty.
Avg. Attempts	Problem	$AA_g(i) = \frac{S_g(i)}{ U_g(i) }$	Measures the average number of attempts per user group, indicating persistence and problem-solving effort beyond correctness, and providing insights into how problems differ in the efforts typically required.
Acceptance Rate	Problem	$AR_g(i) = \frac{ \{U_j \in U_g(i) : y_{ji}=1\} }{ U_g(i) }$	Captures whether problems are solvable across user levels, reflecting overall solvability rather than per-submission correctness, and offering a distinct perspective on problem difficulty through overall solvability.
Avg. Time to Outcome	Problem	$ATTO_g(i) = \frac{1}{ U_g(i) } \sum_{U_j \in U_g(i)} (T_{ji}^{out} - T_{ji}^{first})$	Measures the average duration from the first submission to the outcome (first accepted or last attempt) for problems across user levels, capturing both successful and unsuccessful persistence and providing a time-based view of problem difficulty.
Difficulty Score	Problem	Rasch (β_i) (Eq. 3)	Represents the latent problem difficulty derived from the Rasch model, providing a psychometric scale that reflects inherent hardness beyond observed correctness rates.

Feature Extraction and Construction

To categorize user proficiency and problem difficulty, we derive behavioral features from AOJ submission logs and construct a user-problem interaction dataset (Fig. 1). First, submission records are aggregated into user-level features (accuracy) reflecting proficiency and problem-level features such as total submissions, average (Avg.) attempts, acceptance rate, and avg. time-to-outcome capturing popularity, persistence, performance, and time-based aspects. Next, we fit a Rasch model to binary outcomes of each submission to estimate latent user ability and problem difficulty, and integrate these parameters with corresponding user-problem feature sets to obtain augmented user and problem representations. Then, users are clustered into three proficiency groups: Beginner (Beg), Intermediate (Int), and Advanced (Adv) using user ability and accuracy. Finally, we build group-aware user-problem interaction records by attaching users' proficiency labels and augmented problem features.

To mathematically formalize these features, let problems be denoted as P_i ($1 \leq i \leq N$), users as U_j ($1 \leq j \leq M$), and S_{ji} is the number of submissions made by user U_j on problem P_i . Let s denote a submission and $\mathcal{S}_j$ be the set of submissions made by user U_j ; for each submission s , accepted test cases are AC_s and total test cases are TC_s . Also, let U_g denote the set of users in group g (Beg, Int, and Adv). For a given problem i , $U_g(i) = \{U_j \in U_g : S_{ji} > 0\}$, represents users in group g who attempted problem P_i . Let T_{ji}^{first} and T_{ji}^{out} be the first submission time and the outcome time, where T_{ji}^{out} equals the first accepted time if solved, and otherwise the last submission time made by the user for problem i .

Table 3. Descriptive statistics of features across user groups.

Feature	Group	Mean $\pm$ SD	Skewness		Attempted Problems
			Raw	Processed	
Submissions	Beg	1031.30 $\pm$ 6699.81	11.42	0.73	2171
	Int	1571.44 $\pm$ 7889.30	10.44	0.44	3032
	Adv	862.58 $\pm$ 2638.45	7.35	0.58	3047
	Overall	3161.08 $\pm$ 15846.36	10.77	0.79	3047
Avg. Attempts	Beg	7.24 $\pm$ 11.21	18.97	0.52	2171
	Int	4.95 $\pm$ 4.87	5.46	1.12	3032
	Adv	2.84 $\pm$ 2.38	17.80	1.47	3047
	Overall	3.67 $\pm$ 2.78	11.32	1.11	3047
Acceptance Rate	Beg	0.19 $\pm$ 0.21	2.19	2.19	2171
	Int	0.35 $\pm$ 0.18	0.55	0.55	3032
	Adv	0.54 $\pm$ 0.19	-0.13	-0.13	3047
	Overall	0.43 $\pm$ 0.17	0.23	0.23	3047
Avg. Time to Outcome	Beg	4746.58 $\pm$ 27099.54	17.91	0.00	2171
	Int	19742.42 $\pm$ 78108.62	13.19	-0.45	3032
	Adv	7304.91 $\pm$ 22678.16	13.53	-0.33	3047
	Overall	11156.07 $\pm$ 32721.59	13.44	-0.69	3047
Difficulty Score	Beg	0.00 $\pm$ 0.99	0.24	0.24	2171
	Int	0.00 $\pm$ 0.97	0.71	0.71	3032
	Adv	0.00 $\pm$ 0.92	0.05	0.05	3047
	Overall	0.00 $\pm$ 0.99	1.01	1.01	3047

To obtain Rasch-derived latent features⁴⁰, for each user-problem interaction, $y_{ji} \in \{0, 1\}$ denotes the binary outcomes, where $y_{ji} = 1$ indicates AC submissions and $y_{ji} = 0$ for incorrect submissions (e.g., WA, RE, TLE). The probability of a correct response is modeled as

$$P(y_{ji} = 1) = \sigma(\theta_j - \beta_i) = \frac{1}{1 + e^{-(\theta_j - \beta_i)}} \quad (1)$$

where θ_j and β_i represent the latent ability of user U_j , and difficulty of problem P_i , respectively. $\sigma(\cdot)$ denotes the logistic (sigmoid) function. The model is implemented using PyTorch with a binary cross-entropy loss function, and the parameters are estimated by the maximum likelihood function formulated in Eq. (2), under the identifiability constraint in Eq. (3). Parameter estimation is performed via mini-batch optimization (20 epochs; batch size = 2048) using AdamW (learning rate(lr) = 0.001, weight decay = 0.0001) within each user group, following the same approach as the overall user-problem interaction.

$$L(\theta, \beta) = \sum_{(j,i)} [y_{ji} \log \sigma(\theta_j - \beta_i) + (1 - y_{ji}) \log (1 - \sigma(\theta_j - \beta_i))] \quad (2)$$

$$(\hat{\theta}, \hat{\beta}) = \arg \max_{\theta, \beta} L(\theta, \beta) \quad \text{s.t.} \quad \sum_i \beta_i = 0 \quad (3)$$

A summary of features, including formulation, interpretation, and rationale, is presented in Table 2, while Table 3 reports descriptive statistics of problem-level features across users.

Data Preprocessing

For clustering-based modeling, data preprocessing plays a crucial role because raw data often contains missing values, outliers, and heterogeneous distributions that can adversely affect clustering performance. Preprocessing steps contain several steps, such as data cleaning, dimension reduction, integration, and transformations⁵¹. The preprocessing techniques used in this study are illustrated within Fig. 1. First, categorical features (e.g., status) were converted into numerical codes (e.g., AC = 1, WA = 2, and so on). Next, outliers were identified using boxplots based on the inter-quartile range. Submissions, avg. attempts, and avg. time to outcome were highly skewed and showed large variability (Table 3), with many outliers across user groups (Fig. 2), which can distort Euclidean distance and bias cluster assignment.

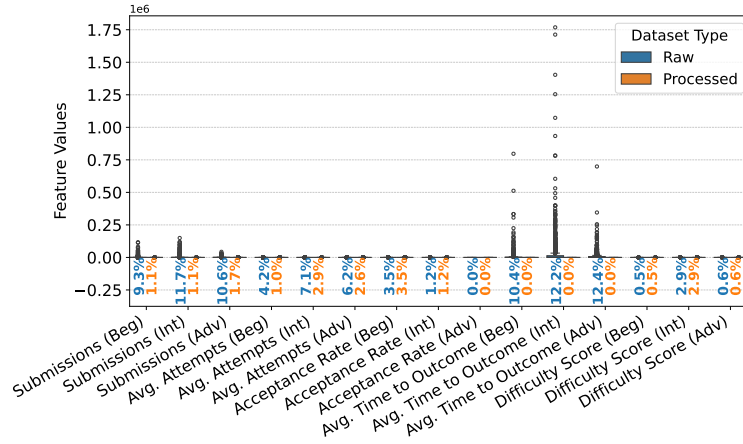


Figure 2. Box-plot of raw vs processed features.

To reduce the skewness and stabilize variability while preserving the observed AOJ patterns (i.e., without removing or imputing observations), we applied log-transformation to these three features, which is effective for right-skewed variables^{17,52}. After transformation, skewness reduced substantially; for example, the skewness of submissions for *Beg* dropped from 11.42 to 0.73 (moderately skewed) (Table 3). The outlier proportion also reduced markedly (e.g., Avg. time-to-outcome outliers dropped to 0 across all groups and submissions outliers for *Beg* decreased from 9.3% to 1.1%), as illustrated in Fig. 2. The remaining features were mildly skewed (approximately normal) with comparatively few outliers, and therefore were not transformed (Figs. 2, and 3). Finally, as feature means and standard deviations (SD) varied considerably across user groups, potentially affecting distance computations, we used Z-score normalization to map all features to a common scale before clustering.

Proposed Difficulty Clustering Approach

The proposed clustering approach has two steps: (i) group-wise clustering with feature-wise weighting and (ii) cross-group difficulty alignment. For each user group, a feature-wise attention module learns the importance of each feature, and K-means is applied separately to each group to produce initial difficulty levels. The resulting labels are then refined by a global alignment step to map levels onto a common monotonic ordering. To demonstrate the effectiveness and competitiveness of the proposed attention-based clustering, we compare it with several widely used clustering baselines, including distance-based methods such as K-means⁵³ and hierarchical clustering (Ward)⁵⁴, soft clustering via FCM⁵⁵, and the BIRCH algorithm⁵⁶. The two integrated components of the framework are detailed below.

Attention-Based Difficulty Clustering

AMs are widely used to learn feature weights for clustering and classification by emphasizing informative features while downweighting less relevant ones. In both sequence modeling⁵⁷ and tabular learning⁴⁹, attention has been shown to effectively capture complex patterns including nonlinear feature relationships. Since the features patterns in our dataset appear complex

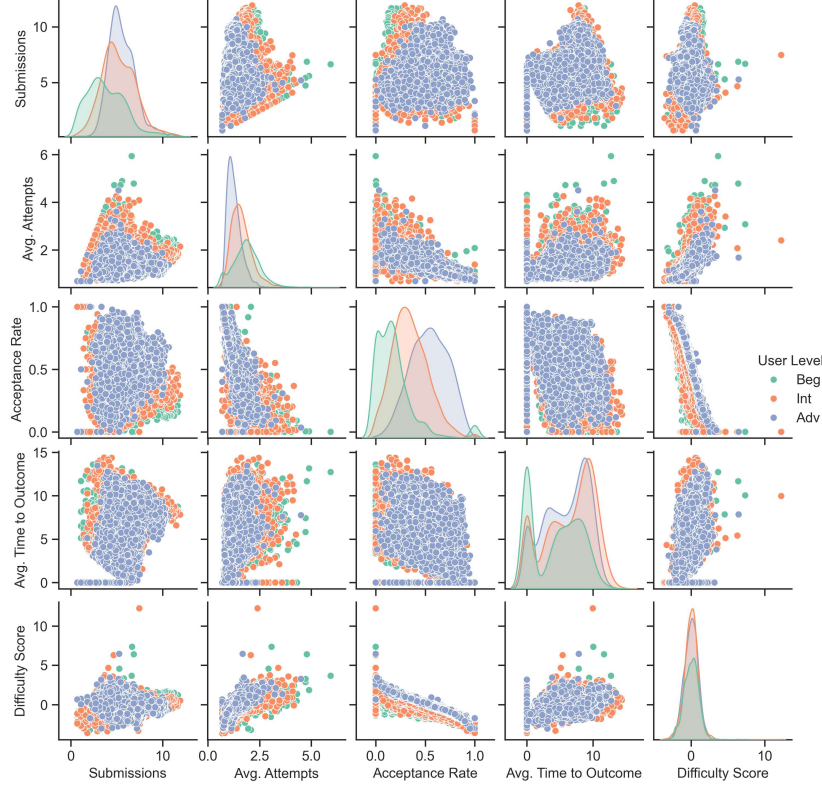


Figure 3. Pairwise scatter plots of user-group features, with diagonal panels showing the marginal distributions of each feature.

Algorithm 1 Feature-Wise Attention Weighted K-means

Input: Preprocessed feature sets $\{X^{(g)}\}$, clusters K , lr η , and epochs E

Output: Cluster labels $C^{(g)}$, centroids $\mu^{(g)}$, and attention weights $A^{(g)}$ for all $g \in \{\text{Beg}, \text{Int}, \text{Adv}\}$

- 1: **for** each user group $g \in \{\text{Beg}, \text{Int}, \text{Adv}\}$ **do**
 - 2: Initialize attention and reconstruction parameters $W_{att}^{(g)}, b_{att}^{(g)}, W_{dec}^{(g)}, b_{dec}^{(g)}$
 - 3: **for** epoch = 1 to E **do**
 - 4: Compute feature-wise attention scores, weights, and attention weighted representations using Eqs.(4)–(6) to obtain $A^{(g)}$ and $Z^{(g)}$
 - 5: Calculate mean reconstruction loss

$$L_{\text{rec}} = \frac{1}{n_g d} \sum_{i=1}^{n_g} \sum_{f=1}^d \left(x_{if}^{(g)} - \hat{x}_{if}^{(g)} \right)^2$$
 - 6: Update $W_{att}^{(g)}, b_{att}^{(g)}, W_{dec}^{(g)}, b_{dec}^{(g)}$ via backpropagation using η
 - 7: **end for**
 - 8: Apply K-means on $Z^{(g)}$ to obtain $\mu^{(g)}$ and $C^{(g)}$ using Eqs. (7) and (8)
 - 9: **end for**
 - 10: **return** $\{C^{(g)}\}, \{\mu^{(g)}\}, \{A^{(g)}\}$ for all user groups
-

and may not be fully characterized by linear relationships (Fig. 3), we employ a feature-wise attention module that assigns importance scores to each feature before clustering, and the procedure is summarized in Algorithm 1.

For each user group g , we use the preprocessed features matrix $X^{(g)} \in \mathbb{R}^{n_g \times d}$, where n_g and d are the number of problems and features, respectively. To obtain feature-wise attention scores for group g , a linear transformation followed by $\tanh$ activation is applied as follows:

$$S^{(g)} = \tanh(X^{(g)} W_{att}^{(g)} + b_{att}^{(g)}) \quad (4)$$

where $W_{att}^{(g)} \in \mathbb{R}^{d \times d}$ and $b_{att}^{(g)} \in \mathbb{R}^d$ are trainable parameters. Attention weights are obtained by applying the softmax function

across the feature dimension:

$$A^{(g)} = \text{softmax}(S^{(g)}) \quad (5)$$

and attention-weighted representation is computed as

$$Z^{(g)} = A^{(g)} \odot X^{(g)} \quad (6)$$

The attention module is optimized jointly with a linear reconstruction layer by minimizing the mean-squared reconstruction loss between the inputs and their reconstructions, where $\hat{X}^{(g)} = Z^{(g)}W_{\text{dec}}^{(g)} + b_{\text{dec}}^{(g)}$.

After training, $Z^{(g)}$ is used as input to K-means. Let $z_i^{(g)} = [z_{i1}^{(g)}, z_{i2}^{(g)}, \dots, z_{id}^{(g)}]$ denote the representation of problem i in group g , and $\mu_k^{(g)} = [\mu_{k1}^{(g)}, \mu_{k2}^{(g)}, \dots, \mu_{kd}^{(g)}]$ be the centroid of cluster k . Each problem is assigned to the nearest centroid by minimizing the Euclidean distance:

$$c_i^{(g)} = \arg \min_k \sum_{f=1}^d \left(z_{if}^{(g)} - \mu_{kf}^{(g)} \right)^2, f \in \{1, 2, \dots, d\} \quad (7)$$

and the centroids are updated as:

$$\mu_k^{(g)} = \frac{1}{|C_k^{(g)}|} \sum_{i \in C_k^{(g)}} z_i^{(g)} \quad (8)$$

where $c_i^{(g)} \in \{1, 2, \dots, K\}$ denotes the cluster index of problem i in group g , and $C_k^{(g)} = \{i | c_i^{(g)} = k\}$ is the set of problems assigned to cluster k . This iterative process continues until the cluster assignments converge.

Algorithm 2 Global Difficulty Refinement Across User Groups

Input: Preprocessed feature sets $\{X^{(g)}\}$, initial cluster labels $\{C^{(g)}\}$, and centroids $\{\mu^{(g)}\}$

Output: Refined cluster labels $C_{\text{ref}}^{(g)}$ and global difficulty centers $\mu_{\text{global}}(L)$ for $L \in \{\text{Easy}, \text{Medium}, \text{Hard}\}$

```

1: Group-wise cluster summarization
2: for each user group  $g \in \{\text{Beg}, \text{Int}, \text{Adv}\}$  do
3:   for each cluster  $k$  in  $C^{(g)}$  do
4:     Compute centroid  $\mu_k^{(g)}$  from transformed features
5:     Label cluster  $k$  as Easy, Medium, or Hard
6:   end for
7: end for
8: Global difficulty center computation
9: for each difficulty level  $L \in \{\text{Easy}, \text{Medium}, \text{Hard}\}$  do
10:  Collect the group-specific cluster means  $(\mu_k^{(g)})$  assigned to  $L$  from Beg, Int, and Adv groups
11:  Compute their feature-wise average to obtain the global difficulty center  $\mu_{\text{global}}(L)$ 
12: end for
13: Refined label assignment
14: for each user group  $g \in \{\text{Beg}, \text{Int}, \text{Adv}\}$  do
15:  Map each  $\mu_{\text{global}}(L)$  through the same preprocessing/attention used for group  $g$  to obtain refined  $r^{(g)}(L)$ 
16:  for each problem  $i$  in group  $g$  do
17:    Compute attention-weighted representation  $z_i^{(g)}$  (or standardized features for traditional clustering)
18:    Set  $C_{\text{ref}}^{(g)}(i) = \arg \min_L \sum_{f=1}^d (z_{if}^{(g)} - r_f^{(g)}(L))^2$ 
19:  end for
20: end for
21: return  $\{C_{\text{ref}}^{(g)}\}$  and  $\{\mu_{\text{global}}(L)\}$ 

```

Cross-Group Difficulty Alignment

After group-wise clustering using baseline and proposed methods, the resulting difficulty labels may be inconsistent across user groups for some problems. In educational settings, higher-proficiency users should perceive lower or equal difficulty

than lower-proficiency users; that is, for a given problem, the difficulty (L) should follow a monotonic order, e.g., $Beg(L) \geq Int(L) \geq Adv(L)$, where a larger label indicates higher difficulty. To reduce NMO and obtain aligned difficulty levels across user groups, we introduce a global difficulty refinement procedure. First, initial difficulty levels are assigned to clusters based on feature statistics within each group g . Next, centroids with the same labels across user groups are aggregated, and their feature-wise mean defines a global difficulty centroid. These global centers are then mapped through the same preprocessing (for traditional clustering) and attention module (for FA-K-means) for each group g , yielding group-specific anchors. Finally, each problem is reassigned to the closest anchor in Euclidean distance, ensuring consistent difficulty ordering across user groups. The full procedure is shown in Algorithm 2.

Experimental Settings

This section discusses the parameter configurations of the clustering methods. The clustering performance metrics and statistical validation are then presented.

Clustering Setup

The proposed FA-K-means was compared with standard baselines (K-means, FCM, Ward, and BIRCH) using a unified set of preprocessed user-problem features to ensure a consistent comparison, with $K = 3$ fixed for all methods (Fig. 7). For fair comparison, hyperparameters for each method were tuned over predefined ranges as shown in Table 4, and the final configurations were selected using validation metrics and ordering inconsistency measures. We then assessed the impact of global alignment by reporting clustering results before and after refinement for each method. Additionally, we evaluated on a synthetic dataset generated using a rank-based Gaussian copula with empirical-inverse-cumulative distribution function interpolation to preserve both inter-feature dependencies and marginal distributions⁵⁸, using AOJ-selected configurations for generalization.

Table 4. Hyperparameter settings for the clustering methods.

Method	Hyperparameters and Values
K-means	Distance = Euclidean; n_init = 20; max_iter = 1000
FCM	Fuzziness m : 1.5 – 3.0 (step 0.5); distance = Euclidean, max_iter = 1000; error = 0.0004
Ward	Agglomerative (Ward linkage); distance = Euclidean
BIRCH	Threshold: 0.10 – 0.50 (step 0.10); branching factor: 10 – 50 (step 10), distance = Euclidean
FA-K-means	Epochs: 20 – 60 (step 10); lr : 0.001 – 0.005 (step 0.001); optimizer = Adam; distance = Euclidean, n_init = 20; max_iter = 1000

Evaluation Metrics and Validation

The clustering quality of the proposed and baseline methods was evaluated using standard clustering metrics: silhouette score (SS), Davies-Bouldin (DBI), and Calinski-Harabasz index (CHI), defined in Eqs. (9)–(11). Higher SS and CHI indicate more compact and well-separated clusters, whereas lower DBI is preferred. We further introduced the NMO measure to quantify the ordering inconsistency of problem difficulty across user groups. In addition, the KW test was used to support clustering validation and examine statistically significant differences in feature distributions across user cluster levels^{17,59}. The formulas for these metrics are provided below.

Silhouette Score

SS measures the clustering quality of problems (data points) i within their assigned clusters and is defined as⁶⁰:

$$SS^{(g)} = \frac{1}{N_g} \sum_{i=1}^{N_g} \frac{b(i) - a(i)}{\max\{a(i), b(i)\}} \quad (9)$$

where $a(i)$ is the average intra-cluster distance of i and $b(i)$ is its minimum mean distance to any other cluster. N_g denotes the number of problems in group g .

Davies-Bouldin Index

DBI quantifies cluster overlap and is defined as⁶¹:

$$DBI^{(g)} = \frac{1}{K} \sum_{k=1}^K \max_{\ell \neq k} \frac{D_k + D_\ell}{D_{k\ell}} \quad (10)$$

where k and ℓ denote cluster indices, D_k and D_ℓ represent within-cluster dispersion, and $D_{k\ell}$ measures the distance between clusters k and ℓ .

Calinski-Harabasz Index

CHI compares between and within-cluster separation and is defined as¹⁸:

$$CHI^{(g)} = \frac{tr(B_K)/(K-1)}{tr(W_K)/(N_g - K)} \quad (11)$$

where $tr(B_K)$ and $tr(W_K)$ are the traces of between and within-cluster dispersion metrics, respectively.

Non-Monotonic Ordering

NMO is defined as the percentage of problems that violate the expected monotonic difficulty order across user groups (i.e., $Beg(L) \geq Int(L) \geq Adv(L)$, where larger L indicates greater difficulty). Let $v_p = 1$ if problem p violates the order and 0 otherwise. Then,

$$NMO(\%) = \frac{1}{P} \sum_{p=1}^P v_p \times 100 \quad (12)$$

where P is the number of evaluated problems. Lower NMO indicates more consistent difficulty ordering.

Experimental Result

This section presents the user clustering results and key insights, followed by problem difficulty clustering outcomes. We then compare difficulty-related behavioral features across user groups by statistical tests and visualizations.

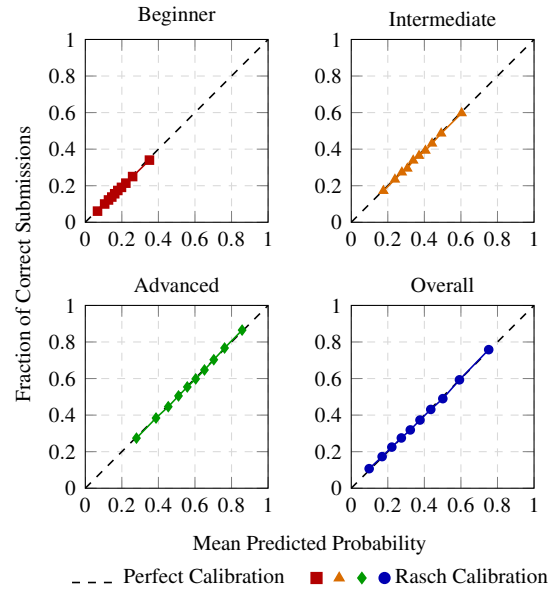


Figure 4. Calibration plots of the Rasch model for different user groups and all users. In each plot, predicted probabilities were divided into 10 quantile-based bins, and each dot represents one bin.

Table 5. Summary of user-level clustering results.

User Level	Users (%)	Submissions (%)	Avg. Accuracy	Avg. Ability Score
Beginner	28.92	23.24	0.20	-1.50
Intermediate	44.01	49.47	0.42	-0.29
Advanced	27.07	27.29	0.66	0.82

User Clustering and Group-Specific Analysis

Using the Rasch-based user ability estimates defined in [Feature Extraction and Construction](#) section, users were clustered based on overall user ability and accuracy. The calibration plot in Fig. 4 showed close agreement between Rasch-predicted probabilities and the observed fraction of correct submissions, indicating that the Rasch model provided well-calibrated probability estimates based on latent user ability and problem difficulty scores. We evaluated three to five clusters using SS, CHI, and DBI, and selected three clusters, as this setting achieved the highest SS (0.49), and CHI (163383.63), and the lowest DBI (0.63). Fig. 5 presents a principal component analysis (PCA) plot of the selected three clusters using the first two principal components (PC1 and PC2), showing well-separated clusters with minimal overlap. The clusters were labeled as *Beg*, *Int*, and *Adv* by ordering their mean ability and accuracy (Table 5), which increased monotonically from *Beg* to *Adv*. We then examined the resulting group composition and activity patterns, and found that *Int* users constituted the largest group (44.01%), followed by *Beg* (28.92%) and *Adv* (27.07%). Regarding submission activity, *Int* users generated the highest volume of submissions, while *Adv* users submitted more than *Beg* users.

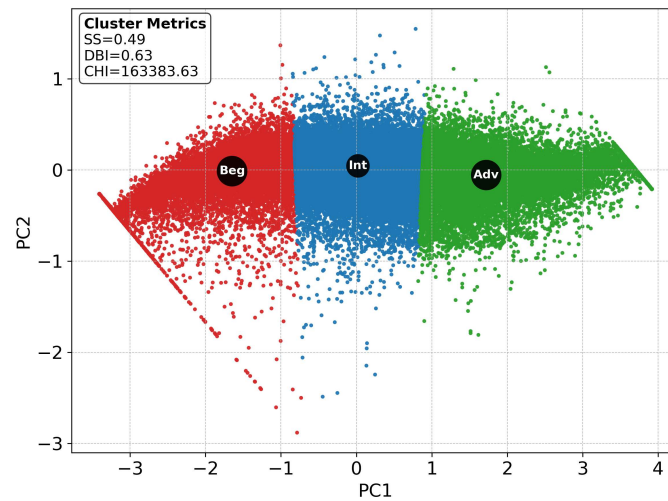


Figure 5. PCA visualization of user clusters with validation metrics.

Fig. 6 summarizes the users' submission outcome distributions and reveals a clear proficiency-dependent trend: the proportion of AC submissions increased from *Beg* to *Adv* (AC: *Beg* 17.45%, *Int* 35.81%, and *Adv* 57.44%), while WA decreased correspondingly (*Beg*: 34.54%, *Int*: 30.54%, *Adv*: 22.88%). Other incorrect outcomes, such as CE and RE, also declined across groups, whereas the remaining outcomes exhibited only minor differences. Consistent with this outcome separation, Table 3 reports the feature average by user group (and overall). Avg. attempt decreased from *Beg* to *Adv* (7.24 $\rightarrow$ 2.84), and the acceptance rate rose substantially (0.19 $\rightarrow$ 0.54), while avg. time-to-outcome and submissions varied across groups without a strictly increasing trend.

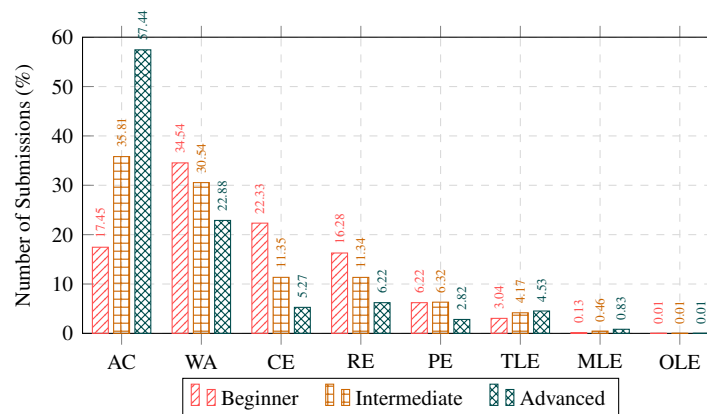


Figure 6. User submission outcomes in the AOJ system.

We further conducted cross-group statistical analyses on the processed feature set using permutational multivariate analysis of variance (PERMANOVA)⁶² test, an intra-class correlation (ICC)⁶³ analysis, and Spearman correlation coefficients. PERMANOVA results in Table 6 rejected the null hypothesis of equal multivariate feature distributions across user groups ($p < 0.01$), indicating that pooling all users into a single population can hinder group-specific structure. In addition, feature-wise ICCs with 95% confidence intervals (CI) indicated only poor-to-moderate cross-group consistency ($r_{int} = 0.34\text{--}0.63$), suggesting that several features were not equally stable across user proficiency levels. For example, average attempts exhibited lower consistency (0.34), whereas submissions were comparatively more stable (0.63). Spearman correlations further demonstrated the feature-specific similarity across groups: submissions remained strongly aligned across group pairs (0.69–0.91), while avg. attempts and avg. time-to-outcome showed lower-to-moderate similarity. Acceptance rate exhibited moderate-to-strong similarity, whereas the difficulty score was particularly low between *Beg* and *Adv* (0.16). Collectively, these findings offered quantitative evidence supporting user-aware difficulty analysis in subsequent clustering stages.

Table 6. Statistical tests of features across user groups.

Feature	Pairwise Correlation			ICC		PERMANOVA	
	Beg vs Int	Beg vs Adv	Int vs Adv	r_{int}	95% CI	Test Stat	p-value
Submissions	0.79	0.69	0.91	0.63	0.61 - 0.65	298.84	0.001
Avg. Attempts	0.32	0.25	0.63	0.34	0.31 - 0.37		
Acceptance Rate	0.54	0.54	0.77	0.37	0.35 - 0.39		
Avg. Time to Outcome	0.25	0.25	0.47	0.45	0.42 - 0.47		
Difficulty Score	0.34	0.16	0.68	0.60	0.56 - 0.63		

Problem Difficulty Clustering Results

To cluster the problems, the number of clusters (K) was determined using the elbow method based on the K-means sum of squared errors (SSE). As illustrated in Fig. 7, SSE decreased rapidly up to $K = 3$ and then formed an elbow point, after which reductions became gradual as K increased. Therefore, $K = 3$ was adopted for all user groups and used throughout the remaining clustering experiments.

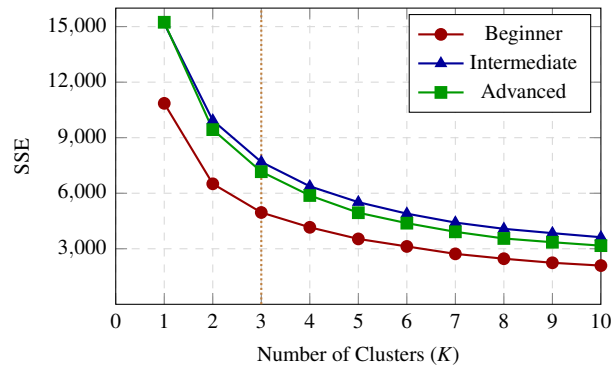


Figure 7. Optimal cluster size identification for each user group. The vertical brown dotted line marks the selected cluster size ($K = 3$).

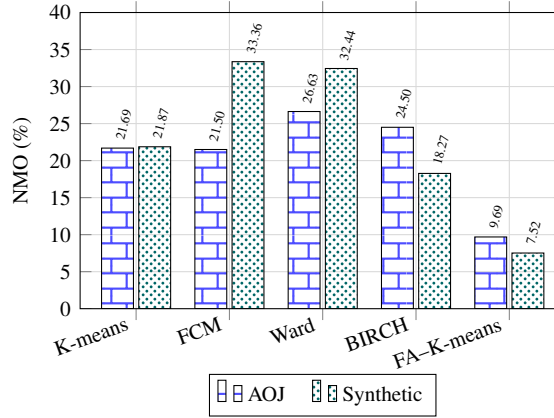
Using $K = 3$ fixed, the proposed FA-K-means was evaluated against K-means, FCM, Ward, and BIRCH under the same preprocessed features on both AOJ and synthetic datasets. As reported in Table 7, for the *Beg* group on the AOJ dataset, the baseline methods yielded validity metrics within the ranges SS: 0.23–0.34; DBI: 1.01–1.38; CHI: 681.28–1288.86, whereas FA-K-means improved these to SS = 0.42, DBI = 0.85, and CHI = 1438.86. Among these baselines, K-means remained the strongest performer; nevertheless, FA-K-means consistently attained higher SS and CHI and lower DBI across groups. Comparable gains were obtained for *Int* (FA-K-means SS: 0.30; DBI: 1.16; CHI: 1624.88) and *Adv* (FA-K-means SS: 0.40; DBI: 0.93; CHI: 2019.08), maintaining improved validity relative to baselines. On the synthetic dataset, a similar improvement was observed across *Beg*, *Int*, and *Adv*; for example, in the *Beg* group, the baseline methods produced SS: 0.20–0.34, DBI: 1.06–1.56, and CHI: 657.25–1080.58, whereas FA-K-means improved these to SS = 0.46, DBI = 0.81, and CHI = 1311.70. Overall, these findings indicate that attention-based feature reweighting captures difficulty-related structure more effectively than uniform feature weighting across both AOJ and synthetic settings.

Table 7. Performance comparison of clustering methods for problem difficulty across user groups.

Method	AOJ Dataset										Synthetic Dataset											
	Beginner			Intermediate			Advanced			NMO (%)		Beginner			Intermediate			Advanced			NMO (%)	
	SS	DBI	CHI	SS	DBI	CHI	SS	DBI	CHI	Pre	Post	SS	DBI	CHI	SS	DBI	CHI	SS	DBI	CHI	Pre	Post
K-means	0.34	1.01	1288.86	0.28	1.18	1471.93	0.31	1.11	1715.00	26.95	21.69	0.34	1.06	1080.58	0.24	1.35	1335.96	0.28	1.22	1709.35	32.72	21.87
FCM	0.34	1.03	1285.41	0.28	1.20	1468.09	0.30	1.12	1709.89	26.53	21.50	0.22	1.47	988.97	0.23	1.39	1327.28	0.27	1.24	1699.26	42.59	33.36
Ward	0.23	1.38	1023.60	0.23	1.34	1088.00	0.26	1.22	1506.06	43.65	26.63	0.20	1.56	743.79	0.18	1.57	1059.17	0.22	1.37	1416.90	43.19	32.44
BIRCH	0.28	1.12	681.28	0.25	1.27	1263.83	0.23	1.29	1317.95	47.30	24.50	0.26	1.09	657.25	0.16	1.14	772.94	0.24	1.31	1440.37	25.47	18.27
FA-K-means	0.42	0.85	1438.86	0.30	1.16	1624.88	0.40	0.93	2019.08	30.41	9.69	0.46	0.81	1311.70	0.36	0.97	1581.97	0.44	0.80	2097.98	45.50	7.52

Note: Results are reported under the selected hyperparameter configuration: FA-K-means (epochs = 60 and $lr = 0.005$), FCM ($m = 1.5$), and BIRCH (threshold = 0.5 and branching factor = 10). Bold values indicate the best result within each dataset.

Beyond clustering validity, cross-group inconsistency was evaluated using the NMO measure, reported as pre-NMO (before refinement) and post-NMO (after refinement). On the AOJ dataset, pre-NMO ranged from 26.53% to 47.30%, implying noticeable ordering discrepancies when clusters were derived independently within each group. After alignment, post-NMO declined substantially for every method; for instance, K-means dropped from 26.95% to 21.69%, and FA-K-means from 30.41% to 9.69%, indicating improved agreement across groups, as shown in Table 7. Notably, FA-K-means + alignment achieved the lowest post-NMO (9.69%), whereas the baseline methods remained at 21.50%–26.63% (Fig. 8), indicating a more stable unified difficulty scale across groups. A consistent trend was observed on the synthetic dataset: alignment consistently reduced post-NMO across all methods (Table 7). FA-K-means yielded the lowest post-NMO (7.52%) compared with 18.27%–33.36% for baselines (Fig. 8), reinforcing cross-group ordering consistency.

**Figure 8.** Comparison of post-refinement NMO rates on the AOJ and synthetic datasets.

To assess separability, two-dimensional PCA projections of the aligned FA-K-means were provided for each user group on the AOJ dataset. The projections depicted a distinct group-wise cluster structure with limited overlap (Fig. 9). Consistent with this visualization, the KW test results in Table 8 rejected the null hypothesis of identical feature distributions at the 1% significance level, indicating statistically significant differences between clusters in the original feature space within each user group.

Problem Difficulty Level Distribution and Feature Patterns

After clustering, the unlabeled clusters were assigned as *easy*, *medium*, or *hard* by ordering them using cluster-wise feature averages reported in Table 8. Specifically, clusters were ranked such that higher difficulty corresponded to a lower acceptance rate and higher values of difficulty score, avg. attempts, and avg. time-to-outcome; the lowest-, middle-, and highest ranked clusters were labeled as *easy*, *medium*, and *hard*, respectively. We also report submissions per level as an activity indicator. In the AOJ dataset, the representative feature values exhibited a clear trend from *easy* to *hard* problems; for instance, in the *Beg* group, the acceptance rate declined from $0.69 \rightarrow 0.24 \rightarrow 0.10$, and avg. attempts increased ($1.85 \rightarrow 3.62 \rightarrow 9.31$), avg. time-to-outcome ($160.12 \rightarrow 1773.91 \rightarrow 6471.20$), and difficulty score ($-1.75 \rightarrow -0.66 \rightarrow 0.49$). Submissions also increased

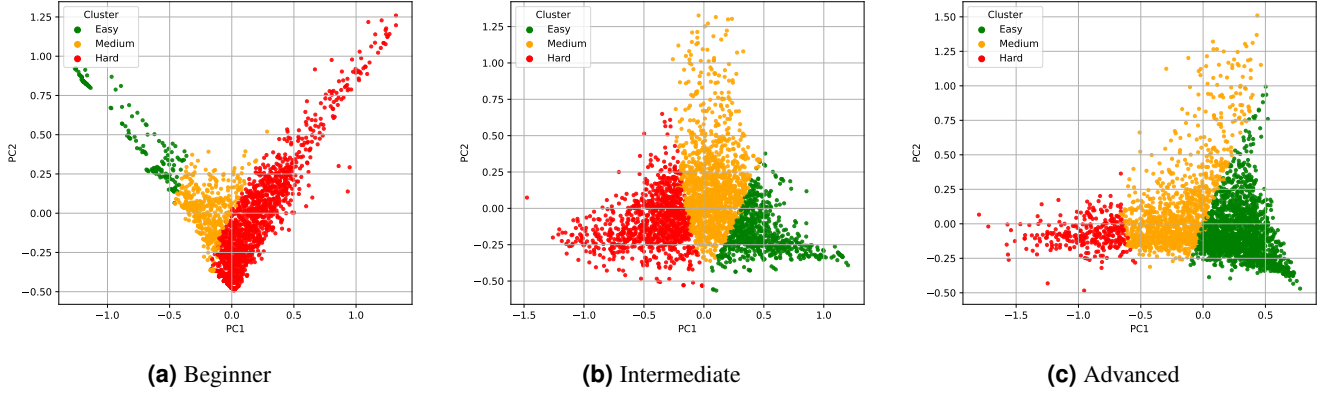


Figure 9. PCA plots of clustered problems for each user group on the AOJ dataset.

Table 8. Cluster-wise feature statistics and KW tests within each user group.

User	Feature	AOJ Dataset			Synthetic Dataset			KW Test on AOJ Dataset			
		Easy	Medium	Hard	Easy	Medium	Hard	Test Stat	p-value	r	ϵ^2
Beginner	Submissions	32.17	90.18	1513.31	363.18	723.19	975.22	348.74	0.000	0.40	0.16
	Avg. Attempts	1.85	3.62	9.31	2.98	3.22	7.35	926.79	0.000	0.65	0.43
	Acceptance Rate	0.69	0.24	0.10	0.73	0.32	0.12	984.50	0.000	0.67	0.45
	Avg. Time to Outcome	160.12	1773.91	6471.20	850.21	4128.27	4535.28	360.40	0.000	0.41	0.17
	Difficulty Score	-1.75	-0.66	0.49	-1.22	-0.94	0.27	1227.06	0.000	0.75	0.56
Intermediate	Submissions	217.03	3223.09	268.47	496.94	4417.86	888.21	674.69	0.000	0.47	0.22
	Avg. Attempts	2.05	4.28	7.84	2.19	3.75	6.46	1716.47	0.000	0.75	0.57
	Acceptance Rate	0.57	0.37	0.16	0.57	0.40	0.21	2304.76	0.000	0.87	0.76
	Avg. Time to Outcome	4047.96	16623.18	34700.25	2376.20	16639.87	24438.57	640.81	0.000	0.46	0.21
	Difficulty Score	-1.19	-0.07	0.91	-1.11	-0.16	0.71	2260.81	0.000	0.86	0.74
Advanced	Submissions	535.97	1540.28	177.89	539.11	2724.37	576.63	153.90	0.000	0.22	0.05
	Avg. Attempts	1.88	3.31	6.28	1.91	3.05	5.07	2033.30	0.000	0.82	0.67
	Acceptance Rate	0.68	0.43	0.21	0.68	0.47	0.26	2190.23	0.000	0.85	0.72
	Avg. Time to Outcome	2642.05	11352.32	17761.12	2678.06	11253.47	18214.28	451.49	0.000	0.39	0.15
	Difficulty Score	-0.63	0.51	1.55	-0.68	0.30	1.31	2107.32	0.000	0.83	0.69

from 32.17 $\rightarrow$ 90.18 $\rightarrow$ 1513.31, reflecting higher activity for harder problems in this group. Similar patterns were observed for the *Int* and *Adv* groups; however, submissions varied (e.g., peaking at the *medium* levels in *Int* and *Adv*), indicating that activity is not only influenced by difficulty but also by problem popularity/visibility. The synthetic data reproduced the same easy-to-hard feature progression, with minor differences primarily in the submission pattern for the *Adv* group.

After assigning aligned cluster labels, the distribution of difficulty levels across user groups for both the AOJ and synthetic datasets is displayed in Fig. 10. The distributions exhibited a distinct proficiency-related shift across difficulty levels, with the proportion of *hard* problems decreasing and that of *easy* problems increasing from *Beg* to *Adv*. In the *Beg* group, most problems were labeled as *hard* (66.51%), followed by *medium* (24.09%), and *easy* (9.40%). The distribution becomes more balanced for the *Int* group, with an increased proportion of *easy* and reduced *hard* proportions, while *medium* remained the largest level (44.49%). In the *Adv* group, the distribution mass shifted toward *easy* (53.86%) with a comparatively smaller *hard* proportion. The synthetic dataset showed a similar group-wise redistribution, suggesting consistent user-group difficulty clustering.

As some problems remained without assigned difficulty levels in the *Beg* and *Int* groups (Table 3), these missing labels were imputed using a random forest (RF) model due to its ability to capture non-linear feature interactions⁶⁴. The RF model was trained on the available labeled problems, and labeled instances from other groups were incorporated when needed (e.g., imputing *Beg* using labels from *Adv* and/or *Int*). Thereafter, to support user-level inspection, a user-aware web-based dashboard was developed to visualize proficiency-specific clustering results as shown in Fig. 11. Using this interface, an AOJ User ID is used for login, and a Problem ID is queried to obtain the inferred user proficiency level and corresponding predicted difficulty level.

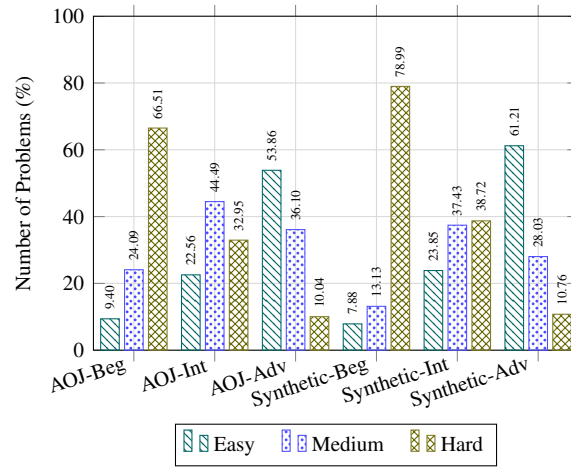


Figure 10. Clustered problem difficulty distributions across user groups.

User Login

User ID:

(a) Login interface

Welcome, U001

User Level: **Beginner**

[Logout](#)

Enter Problem ID:

Problem ID: ALDS1_11_A

Predicted Difficulty: Hard

(b) Beginner

Welcome, U002

User Level: **Intermediate**

[Logout](#)

Enter Problem ID:

Problem ID: ALDS1_11_A

Predicted Difficulty: Medium

(c) Intermediate

Welcome, U003

User Level: **Advanced**

[Logout](#)

Enter Problem ID:

Problem ID: ALDS1_11_A

Predicted Difficulty: Easy

(d) Advanced

Figure 11. Web-based dashboard showing user-specific problem difficulty results across user groups. The figure presents outputs for demo users for a specific problem.

Identification of Key Features for Problem Difficulty Levels

Beyond difficulty distribution, we further identified the key features within each proficiency group using the attention weights learned by FA-K-means. As reported in Fig. 12, acceptance rate was the top-contributing indicator for distinguishing difficulty levels, receiving the largest weights across groups (0.26–0.32). Avg. attempts also contributed substantially (0.18–0.20), underscoring the role of effort-based indicators. Submissions exhibited group-dependent weighting, with a higher weight for the *Beg* group (0.25) and comparatively lower weights for the other two groups. The remaining features, such as difficulty score (0.16–0.18) and avg. time-to-outcome (0.15 for all groups), yielded relatively lower weights, reflecting a smaller contribution to difficulty separation under the learned attention.

To quantify the magnitude of feature-wise separation among difficulty levels, we additionally reported the KW effect sizes

using ϵ^2 , and the correlation-like summary measure r ; larger values indicated stronger separation across levels. As shown in Table 8, large effects were observed for the acceptance rate ($\epsilon^2 = 0.72, r = 0.85$) in the *Adv* group, followed by difficulty score ($\epsilon^2 = 0.69, r = 0.83$), and avg. attempts ($\epsilon^2 = 0.67, r = 0.82$). In contrast, avg. time-to-outcome ($\epsilon^2 = 0.15, r = 0.39$), and submissions ($\epsilon^2 = 0.05, r = 0.22$) yielded comparatively smaller effects, reflecting weaker separation. The *Beg* and *Int* groups showed a largely similar ordering, with minor rank-order changes (e.g., difficulty scores slightly exceeding acceptance rate in the *Beg* group; and submissions slightly exceeding avg. time-to-outcome in the *Int* group), indicating that assigned difficulty levels corresponded to distinct feature distributions. Overall, acceptance rate and avg. attempts emerged as key features, while the ordering of submissions and difficulty score varied, because attention reflected joint contributions, whereas the KW effect sizes captured univariate separation.

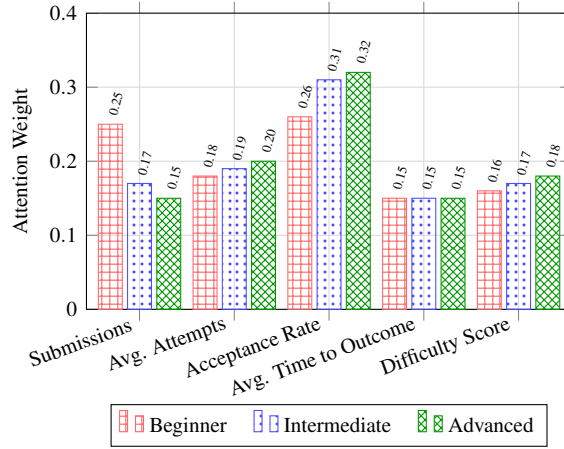


Figure 12. Feature attention weights across user groups on the AOJ dataset.

Discussion

Building on the clustering results and experimental analyses, this section discusses the major findings, practical implementation, and key limitations of the study.

Performance Analysis

This study conducts a user-based analysis using submission logs and Rasch-derived features. The AOJ logs showed marked behavioral differences across proficiency levels (Tables 3 and 5). In particular, more advanced users generally required fewer attempts and achieved higher acceptance rates, indicating more efficient problem-solving behavior. Comparable acceptance rates have also been reported for other OJ platforms (e.g., 0.46 for Timus OJ, 0.47 for POJ, and 0.53 for HDU OJ)¹⁶, broadly aligned with the AOJ rate of 0.43 and suggesting that observed differences may extend beyond AOJ and that difficulty analysis may benefit from group-specific modeling and interpretation. Within this context, FA-K-means yielded more compact and better-separated clusters than the baselines, as reflected by the highest SS and CHI and the lowest DBI (Table 7). These results suggest that attention-guided feature reweighting strengthened within-group difficulty structure by capturing complex feature patterns in our dataset (Fig. 3).

Moreover, FA-K-means incorporated a global alignment step because group-wise clustering alone did not guarantee cross-group comparability; behavioral feature distributions shifted with proficiency, as reflected by group-wise differences in average feature values (Table 3). Global alignment addressed this issue by reducing NMO from 30.41% to 9.69%, and the post-alignment NMO remained lower than that of the baselines, yielding a more stable *Easy* $\rightarrow$ *Medium* $\rightarrow$ *Hard* ordering across groups (Table 7). Overall, the proposed framework preserved strong within-group difficulty structure while enabling coherent cross-group interpretation for OJ analytics.

Feature Contributions and Difficulty-Level Trends

Beyond clustering performance, the framework provides interpretable difficulty levels by exposing the key features that differentiate levels and linking them to learning behavior. As shown in Fig. 12, the learned weights emphasized outcome and effort signals: acceptance rate consistently received the largest weights across groups and avg. attempts also contributed meaningfully (≈ 0.20 , close to a uniform baseline of $1/5$), while the remaining features received lower than uniform weights. This aligned with the KW analysis, which identified these indicators (together with difficulty score) as strong separators across

levels (Table 8), suggesting that difficulty was primarily expressed through success likelihood and effort. Using the final aligned difficulty levels, the level-wise profiles remain pedagogically consistent, showing a stable, easy-to-hard progression in performance and effort indicators across groups (Table 8).

In addition, although clustering was performed on processed features that were obtained by reducing skewness and controlling outliers (Table 3 and Fig. 2), to stabilize distance-based clustering, the level-wise feature patterns were reported in Table 8 on the raw scale to preserve interpretability of original units. Consequently, some group-wise feature differences appeared large because several submission-derived features were right-skewed and heavy-tailed in the raw data, and typical user behavior differed across proficiency groups, e.g., lower proficiency users tending to require more effort than other groups (Tables 3 and 8). These distributional properties mainly affected magnitudes, but they did not change the steady trends (*Easy* $\rightarrow$ *Hard*) that give the levels clear educational meaning. The difficulty level distributions also showed significant differences across user groups (Fig. 10), indicating a proficiency-dependent shift in exposure from harder to easier levels.

Educational Implications for OJ-Based Programming Practice

The derived difficulty levels provide practical implications for OJ-based programming education. By categorizing problems into *easy*, *medium*, and *hard* levels within each proficiency group, the framework can present problem difficulty in a clearer and more user-friendly way. Presenting these difficulty results and user-level profiles through simple visualizations may help learners understand the problem difficulty relative to their skill level and may support more informed, self-directed problem selection. For instructors, the level-wise feature patterns provide interpretable evidence for organizing practice problems across *Beg*, *Int*, and *Adv* learner groups. In addition, the observed level-wise feature patterns suggest a practical rule for problem categorization in OJ systems: problems with higher acceptance rates and lower attempts, avg. time-to-outcome, and difficulty score tend to be categorized as *easy*, whereas those with the opposite pattern tend to be categorized as *hard*, with *medium* problems falling between these two extremes. Overall, the proposed framework supports group-aware reporting and interpretation, instructor-guided problem organization, and learner support in programming practice.

Generalizability and Scalability

The synthetic data validation serves as a practical stability check by evaluating whether the main conclusions depend on a single AOJ dataset. As summarized in Table 7 and Fig. 8, the synthetic results reproduced the same overall difficulty level structures, showing improved clustering under attention-based feature reweighting and a similar reduction in cross-group ordering inconsistency after alignment, consistent with the AOJ findings. The synthetic data also provided a comparable difficulty level distribution (Fig. 10), and preserves the same easy-to-hard progression in the key features with only minimal deviations in submission patterns across user groups, largely in the *Adv* group (Table 8). These deviations were expected due to stochastic sampling variability and because submission volume can be influenced by popularity/visibility, in addition to difficulty. Overall, the agreement between AOJ and synthetic results supports the robustness and generalizability of the proposed framework, as well as the interpretability of the derived difficulty levels across both settings.

The proposed framework is scalable to large volumes of OJ submission data because it aggregates raw submissions into compact problem-level indicators within each proficiency group and performs clustering on these aggregated representations. The pipeline follows standard K-means, adding only feature weights and an alignment step based on cluster-level summaries to reduce cross-group inconsistency. It achieved better validity scores than the baselines. As the dataset grows, the aggregated indicators can be updated with new submissions and users, enabling periodic re-training without changing the overall pipeline in OJ systems. These findings suggest that FA-K-means has potential for other domains such as education, business, economics, and engineering, where lower-dimensional numerical features are available.

Although we applied the framework to AOJ problem difficulty clustering across heterogeneous users, it can be extended to other OJ platforms and large-scale datasets with similar interaction logs. More broadly, it can be applied to domains with shared tasks/items and diverse user groups, such as quizzes/questions/assignments of e-learning platforms, adaptive practices across different subjects (e.g., engineering, science, and arts), educational games (*Beg* vs *Int* vs *Adv* players), and help-center services for novice and expert customers.

Limitations

The proposed framework showed strong performance for categorizing problems on the AOJ dataset; however, its effectiveness may vary due to several practical limitations: (i) user behavior, submission patterns, and problem characteristics may differ across other OJ platforms and data contexts (e.g., smaller or sparser datasets, or higher-dimensional feature settings); (ii) the synthetic dataset provides a stability check, but it was constructed to resemble the AOJ feature structures and may not fully capture platform-specific factors; (iii) clustering results depend on methodological choices (preprocessing, the selected K , feature weighting, and distance measures), and the proposed alignment was used as simple and interpretable post hoc strategy to improve cross-group consistency, however it assumes that cluster-level summaries are comparable across groups, does not guarantee a globally optimal alignment, and may degrade when the initial clusters are weak; (iv) difficulty levels are inferred

from submission-derived behavioral features and may not fully represent problem complexity or code-quality aspects; and (v) since ground-truth levels are unavailable at scale, validation relies on internal metrics and statistical tests rather than direct label agreement. Overall, these limitations suggest that performance may differ when applying the proposed approach to other platforms or data conditions.

Conclusion and Future Work

This paper presents a user-aware framework for consistent programming problem difficulty categorization across proficiency groups using submission-derived behavioral features along with Rasch-based latent estimates. On AOJ, the overall acceptance rate was 0.43, and outcomes differed across proficiency groups, underscoring the importance of user-group-specific difficulty interpretation. The integrated framework combines FA–K-means to strengthen within-group difficulty structures, achieving better validity scores (highest SS, and CHI with lowest DBI) than traditional clustering baselines; and a global alignment step to improve cross-group comparability by reducing inconsistency (NMO from 30.41% to 9.69%). The learned feature contributions, supported by the KW test analysis, further indicate that acceptance rate and effort-based indicators are most effective for distinguishing difficulty levels. The synthetic data validation reproduced similar trends and consistency improvements, providing an additional stability check. Overall, the proposed framework offers a practical basis for user-aware difficulty interpretations and potential user-facing visualization in OJ analytics, and it can be extended to other domains with heterogeneous user groups.

While this study focuses on user-based problem categorization, future work will examine users' problem selection patterns and how the ordering of problem difficulty is associated with performance and progression across proficiency groups. We also plan to validate on additional OJ platforms and datasets with smaller or sparser data conditions to assess generalizability. In addition, richer feature sets (e.g., code-quality proxies and problem-content indicators) will be incorporated to capture problem difficulty more comprehensively beyond submission-derived features, alongside more systematic alignment strategies to improve cross-group consistency.

References

1. Woo, K. & Falloon, G. Integrating coding across the curriculum: a scoping review. *Comput. Sci. Educ.* **35**, 216–237, DOI: [10.1080/08993408.2024.2344402](https://doi.org/10.1080/08993408.2024.2344402) (2025).
2. 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).
3. Saito, T. & Watanobe, Y. Learning path recommendation system for programming education based on neural networks. *Int. J. Distance Educ. Technol. (IJDET)* **18**, 36–64, DOI: <https://doi.org/10.4018/IJDET.2020010103> (2020).
4. Revilla, M. A., Manzoor, S. & Liu, R. Competitive learning in informatics: The uva online judge experience. *Olymp. Informatics* **2**, 131–148 (2008).
5. 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).
6. Mi, H., Liu, Y., Meng, R., Tang, Y. & Zhang, Q. A study of learning behavior analysis and student achievement based on an online judge system. In *Proceedings of the 2024 9th International Conference on Distance Education and Learning*, 278–284, DOI: <https://doi.org/10.1145/3675812.3675817> (2024).
7. Omer, U., Farooq, M. S. & Abid, A. Introductory programming course: review and future implications. *PeerJ Comput. Sci.* **7**, e647, DOI: <https://doi.org/10.7717/peerj-cs.647> (2021).
8. Lai, C.-H. & Lin, C.-Y. Analysis of learning behaviors and outcomes for students with different knowledge levels: A case study of intelligent tutoring system for coding and learning (its-cal). *Appl. Sci. (2076-3417)* **15**, DOI: <https://doi.org/10.3390/app15041922> (2025).
9. Mekterović, I., Brkić, L., Milašinović, B. & Baranović, M. Building a comprehensive automated programming assessment system. *IEEE Access* **8**, 81154–81172, DOI: [10.1109/ACCESS.2020.2990980](https://doi.org/10.1109/ACCESS.2020.2990980) (2020).
10. Petit, J. *et al.* Jutge. org: Characteristics and experiences. *IEEE Transactions on Learn. Technol.* **11**, 321–333 (2017).
11. 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).
12. Watanobe, Y. Aizu online judge. [Online]. Available: <https://onlinejudge.u-aizu.ac.jp> (2018). Accessed: 24 Oct. 2025.

13. Verdú, E. *et al.* A distributed system for learning programming on-line. *Comput. & Educ.* **58**, 1–10, DOI: <http://dx.doi.org/10.1016/j.compedu.2011.08.015> (2012).
14. Wang, Z., Zhang, W. & Wang, J. Estimating difficulty levels of programming problems with pre-trained model. *arXiv preprint arXiv:2406.08828* (2024).
15. Malik, S., Al-Emran, M., Mathew, R., Tawafak, R. & AlFarsi, G. Comparison of e-learning, m-learning and game-based learning in programming education—a gendered analysis. *Int. J. Emerg. Technol. Learn. (IJET)* **15**, 133–146, DOI: <https://doi.org/10.3991/ijet.v15i15.14503> (2020).
16. Zhao, W. X., Zhang, W., He, Y., Xie, X. & Wen, J.-R. Automatically learning topics and difficulty levels of problems in online judge systems. *ACM Transactions on Inf. Syst. (TOIS)* **36**, 1–33, DOI: <https://doi.org/10.1145/3158670> (2018).
17. 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).
18. Suciati, N., Fabroyir, H., Pardede, E. *et al.* Educational data mining clustering approach: case study of undergraduate student thesis topic. *IEEE Access* **11**, 130072–130088, DOI: <https://doi.org/10.1109/ACCESS.2023.3332818> (2023).
19. Intisar, C. M. & Watanobe, Y. Cluster analysis to estimate the difficulty of programming problems. In *Proceedings of the 3rd International Conference on Applications in Information Technology*, 23–28, DOI: <https://doi.org/10.1145/3274856.3274862> (2018).
20. Muepu, D. M. & Watanobe, Y. From code to ratings: Converting programming data to enhance collaborative filtering in educational online judge systems. *IEEE Access* **12**, 196670–196687, DOI: <https://doi.org/10.1109/ACCESS.2024.3522118> (2024).
21. 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).
22. 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: <https://doi.org/10.1109/TLT.2023.3239110> (2023).
23. Mia, M. S., Watanobe, Y., Rahman, M. M., Ibne Amin, M. F. & 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) (2025).
24. Pereira, F. D. *et al.* Explaining individual and collective programming students' behavior by interpreting a black-box predictive model. *IEEE Access* **9**, 117097–117119, DOI: <https://doi.org/10.1109/ACCESS.2021.3105956> (2021).
25. Xia, H. *et al.* Robust clustering model based on attention mechanism and graph convolutional network. *IEEE Transactions on Knowl. Data Eng.* **35**, 5203–5215, DOI: <https://doi.org/10.1109/TKDE.2022.3150300> (2022).
26. Yao, Z., Li, B. & Liao, Y. Discriminative representation learning via attention-enhanced contrastive learning for short text clustering. *Neural Networks* 108101, DOI: <https://doi.org/10.1016/j.neunet.2025.108101> (2025).
27. Zhang, X. *et al.* Tfw: Tabular feature weighting with transformer. *arXiv preprint arXiv:2405.08403* (2024).
28. Wasik, S., Antczak, M., Badura, J., Laskowski, A. & Sternal, T. A survey on online judge systems and their applications. *ACM Comput. Surv. (CSUR)* **51**, 1–34, DOI: <https://doi.org/10.1145/3143560> (2018).
29. Puri, R. *et al.* Codenet: A large-scale ai for code dataset for learning a diversity of coding tasks. *arXiv preprint arXiv:2105.12655* (2021).
30. Li, Y. *et al.* Competition-level code generation with alphacode. *Science* **378**, 1092–1097, DOI: <https://doi.org/10.1126/science.abq1158> (2022).
31. Hendrycks, D. *et al.* Measuring coding challenge competence with apps. *arXiv preprint arXiv:2105.09938* (2021).
32. Amin, M. F. I., Shirafuji, A., Rahman, M. M. & Watanobe, Y. Multi-label code error classification using codet5 and ml-knn. *IEEE Access* **12**, 100805–100820, DOI: <https://doi.org/10.1109/ACCESS.2024.3430558> (2024).
33. Amin, M. F. I., Watanobe, Y., Rahman, M. M. & Shirafuji, A. Source code error understanding using bert for multi-label classification. *IEEE Access* **13**, 3802–3822, DOI: <https://doi.org/10.1109/ACCESS.2024.3525061> (2025).
34. Llanos, J., Bucheli, V. A. & Restrepo-Calle, F. Early prediction of student performance in cs1 programming courses. *PeerJ Comput. Sci.* **9**, e1655, DOI: <https://doi.org/10.7717/peerj-cs.1655> (2023).

35. Xu, B., Yan, S., Jiang, X. & Feng, S. Scfh: A student analysis model to identify students' programming levels in online judge systems. *Symmetry* **12**, 601, DOI: <https://doi.org/10.3390/sym12040601> (2020).
36. Fu, Q. *et al.* Visoj: real-time visual learning analytics dashboard for online programming judge. *The Vis. Comput.* **39**, 2393–2405, DOI: <https://doi.org/10.1007/s00371-022-02586-z> (2023).
37. Ferguson, R. Learning analytics: drivers, developments and challenges. *Int. J. Technol. Enhanced Learn.* **4**, 304–317, DOI: <https://doi.org/10.1504/IJTEL.2012.051816> (2012).
38. Uto, M. & Ueno, M. Item response theory for peer assessment. *IEEE Transactions on Learn. Technol.* **9**, 157–170, DOI: <https://doi.org/10.1109/TLT.2015.2476806> (2015).
39. Wauters, K., Desmet, P. & Van Den Noortgate, W. Adaptive item-based learning environments based on the item response theory: Possibilities and challenges. *J. Comput. Assist. Learn.* **26**, 549–562, DOI: <https://doi.org/10.1111/j.1365-2729.2010.00368.x> (2010).
40. Yeung, C.-K. Deep-irt: Make deep learning based knowledge tracing explainable using item response theory. *arXiv preprint arXiv:1904.11738* (2019).
41. van der Linden, W. J. (ed.) *Handbook of Item Response Theory*, vol. 1 (CRC Press, New York, 2016).
42. Sakabe, T., Sakurai, Y., Tsutsumi, E. & Oyama, S. Dataset construction using item response theory for educational machine learning competitions. *IEEE Access* **13**, 110226–110240, DOI: <https://doi.org/10.1109/ACCESS.2025.3582016> (2025).
43. Zaffalon, F. *et al.* Student's multiple abilities and skills model for online judge systems. In *2022 IEEE Frontiers in Education Conference (FIE)*, 1–7, DOI: <https://doi.org/10.1109/FIE56618.2022.9962428> (IEEE, 2022).
44. Moraes, L. O. & Pedreira, C. E. Clustering introductory computer science exercises using topic modeling methods. *IEEE Transactions on Learn. Technol.* **14**, 42–54 (2021).
45. Silva, D. B. & Silla, C. N. Evaluation of students programming skills on a computer programming course with a hierarchical clustering algorithm. In *2020 IEEE Frontiers in Education Conference (FIE)*, 1–9, DOI: <https://doi.org/10.1109/FIE44824.2020.9274130> (IEEE, 2020).
46. Modha, D. S. & Spangler, W. S. Feature weighting in k-means clustering. *Mach. Learn.* **52**, 217–237, DOI: <https://doi.org/10.1023/A:1024016609528> (2003).
47. Huang, X., Yang, X., Zhao, J., Xiong, L. & Ye, Y. A new weighting k-means type clustering framework with an l2-norm regularization. *Knowledge-Based Syst.* **151**, 165–179, DOI: <https://doi.org/10.1016/j.knosys.2018.03.028> (2018).
48. Peng, Z., Liu, H., Jia, Y. & Hou, J. Deep attention-guided graph clustering with dual self-supervision. *IEEE Transactions on Circuits Syst. for Video Technol.* **33**, 3296–3307, DOI: <https://doi.org/10.1109/TCSVT.2022.3232604> (2022).
49. Arik, S. Ö. & Pfister, T. Tabnet: Attentive interpretable tabular learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, 6679–6687, DOI: <https://doi.org/10.1609/aaai.v35i8.16826> (2021).
50. Aoj developers portal. <https://judge.u-aizu.ac.jp/onlinejudge/api.jsp> (2025). Accessed: 24 Oct. 2025.
51. Alghamdi, T. A. & Javaid, N. A survey of preprocessing methods used for analysis of big data originated from smart grids. *IEEE Access* **10**, 29149–29171, DOI: <https://doi.org/10.1109/ACCESS.2022.3157941> (2022).
52. West, R. M. Best practice in statistics: The use of log transformation. *Annals Clin. Biochem.* **59**, 162–165, DOI: <https://doi.org/10.1177/00045632211050531> (2022).
53. Lloyd, S. Least squares quantization in pcm. *IEEE Transactions on Inf. Theory* **28**, 129–137, DOI: <https://doi.org/10.1109/TIT.1982.1056489> (1982).
54. Ward Jr, J. H. Hierarchical grouping to optimize an objective function. *J. Am. Stat. Assoc.* **58**, 236–244 (1963).
55. Bezdek, J. C. *Pattern recognition with fuzzy objective function algorithms* (Springer Science & Business Media, New York, 2013).
56. Zhang, T., Ramakrishnan, R. & Livny, M. Birch: an efficient data clustering method for very large databases. *ACM Sigmod Rec.* **25**, 103–114, DOI: <https://doi.org/10.1145/235968.233324> (1996).
57. Bahdanau, D., Cho, K. & Bengio, Y. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473* (2014).
58. Liu, H., Lafferty, J. & Wasserman, L. The nonparanormal: semiparametric estimation of high dimensional undirected graphs. *J. Mach. Learn. Res.* **10**, 2295–2328 (2009).
59. Okoye, K. & Hosseini, S. *R Programming: Statistical Data Analysis in Research* (Springer Nature, Singapore, 2024).

60. Rousseeuw, P. J. Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. *J. Comput. Appl. Math.* **20**, 53–65, DOI: [https://doi.org/10.1016/0377-0427\(87\)90125-7](https://doi.org/10.1016/0377-0427(87)90125-7) (1987).
61. Davies, D. L. & Bouldin, D. W. A cluster separation measure. *IEEE Transactions on Pattern Analysis Mach. Intell.* 224–227, DOI: <https://doi.org/10.1109/TPAMI.1979.4766909> (2009).
62. Anderson, M. J. Permutational multivariate analysis of variance (permanova). *Wiley StatsRef: Stat. Ref. Online* 1–15, DOI: <https://doi.org/10.1002/9781118445112.stat07841> (2014).
63. Koo, T. K. & Li, M. Y. A guideline of selecting and reporting intraclass correlation coefficients for reliability research. *J. Chiropr. Medicine* **15**, 155–163, DOI: <https://doi.org/10.1016/j.jcm.2016.02.012> (2016).
64. Stekhoven, D. J. & Bühlmann, P. Missforest—non-parametric missing value imputation for mixed-type data. *Bioinformatics* **28**, 112–118, DOI: <https://doi.org/10.1093/bioinformatics/btr597> (2012).

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. F.L.: Resources, Investigation, 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.