IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received August 26, 2021, accepted October 5, 2021, date of publication October 8, 2021, date of current version October 19, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3119145

Impact of Practical Skills on Academic Performance: A Data-Driven Analysis

MD. MOSTAFIZER RAHMAN[1,2], YUTAKA WATANOBE[1], (Member, IEEE), RAGE UDAY KIRAN[1], TRUONG CONG THANG[1], (Senior Member, IEEE), AND INCHEON PAIK[1], (Senior Member, IEEE)

[1]Graduate Department of Computer Science and Engineering, The University of Aizu, Aizuwakamatsu, Fukushima 965-8580, Japan
[2]Department of Computer Science and Engineering, Dhaka University of Engineering & Technology, Gazipur 1707, Bangladesh

Corresponding authors: Md. Mostafizer Rahman (mostafiz26@gmail.com) and Yutaka Watanobe (yutaka@u-aizu.ac.jp)

This work was supported by the Japan Society for the Promotion of Science (JSPS) KAKENHI under Grant 19K12252.

This work involved human subjects or animals in its research. Approval of all ethical and experimental procedures and protocols was granted by the Research Ethics Examination Boards, The University of Aizu, Japan.

ABSTRACT Most academic courses in information and communication technology (ICT) or engineering disciplines are designed to improve practical skills; however, practical skills and theoretical knowledge are equally important to achieve high academic performance. This research aims to explore how practical skills are influential in improving students' academic performance by collecting real-world data from a computer programming course in the ICT discipline. Today, computer programming has become an indispensable skill for its wide range of applications and significance across the world. In this paper, a novel framework to extract hidden features and related association rules using a real-world dataset is proposed. An unsupervised k-means clustering algorithm is applied for data clustering, and then the frequent pattern-growth algorithm is used for association rule mining. We leverage students' programming logs and academic scores as an experimental dataset. The programming logs are collected from an online judge (OJ) system, as OJs play a key role in conducting programming practices, competitions, assignments, and tests. To explore the correlation between practical (e.g., programming, logical implementations, etc.) skills and overall academic performance, the statistical features of students are analyzed and the related results are presented. A number of useful recommendations are provided for students in each cluster based on the identified hidden features. In addition, the analytical results of this paper can help teachers prepare effective lesson plans, evaluate programs with special arrangements, and identify the academic weaknesses of students. Moreover, a prototype of the proposed approach and data-driven analytical results can be applied to other practical courses in ICT or engineering disciplines.

INDEX TERMS Practical skills, programming education, feature extraction, educational data mining, learning analytics, e-learning, online judge, clustering, association rule mining.

I. INTRODUCTION

Most courses in information and communication technology (ICT), computer science, and engineering-related disciplines are designed with a practical basis. Basically, each course consists of two parts namely, theory and exercise where theory develops students' theoretical knowledge, ideas, and memorization. In contrast, exercise or practical application develops logic, critical thinking, problem-solving skills, and implementation skills. Computer programming is an example of a practical course in these disciplines. The necessity of programming education is rapidly growing in pace with the increasing expansion of computers into our daily lives; thus, computer programming is among the key courses in the ICT discipline and has become a foundational course in other disciplines, as well [1]. In a recent effort to encourage students, including children, to take an increased interest in programming, numerous online programming platforms have become available. Here, it should be noted that because the primary requirement of programming education is to ensure

The associate editor coordinating the review of this manuscript and approving it for publication was Meriel Huggard.

that students achieve computer literacy [1], most educational institutions that teach programming have redesigned their academic curriculum to effectively meet the basic literacy requirements of programming education.

Basic computer programming courses are normally available in the first semester of university studies. Initial programming classes have the ancillary role of attracting students to the field of computer programming. Because students may make decisions based on these initial programming classes, it is essential that those classes impart positive programming experiences. Note that, introductory programming courses have a significant rate of failure and dropout [2]. However, due to limited amounts of class time, classrooms and teachers, and limitations in other forms of logistic support, it is difficult to fully educate students in programming through traditional programming classes alone. To overcome these problems, online judge (OJ) systems provide additional platforms that enable students to continue their programming studies over a period of years [3]. Such systems normally contain large collections of interesting programming problems [4] that students can pursue independently or teachers can assign to stimulate students' interest. The concept of the OJ system was first introduced at the 1977 International Collegiate Programming Contest (ICPC) [5], [6], which is now held annually. Furthermore, because OJ systems have proven useful, many universities and colleges are now attempting to develop online support systems for programming education [7]–[9].

Today, OJs are used by many educational institutions to conduct courses related to programming, computing, and software engineering [10], [11]. Many universities have created their own automated program assessment (APA) systems for programming courses to accelerate students' learning [12]–[14]. As a result, a large number of programming-related submission logs are created every day by OJ or APA systems in various organizations worldwide, which can be valuable resources for research and analysis [15], [16]. Therefore, this research aims to use programming-related resources (submission logs) for empirical research and analysis.

Educational data collected from various e-learning platforms such as Moodle, MOOCs, OJs, and APAs are not unified, structured, well-organized, neat and in a collected format because the data archiving format differs from one e-learning platform to another. Therefore, educational data mining (EDM) and learning analytics (LA) techniques are effective in transforming these big educational data into useful knowledge and patterns that can be applied to improve overall education. EDM has become an effective technique for exploring invisible knowledge and useful patterns in educational data. Nowadays, traditional education is changing at an unprecedented pace and many academic activities are conducted on e-learning platforms. The collection of this vast amount of educational data has opened up opportunities for research and analysis to understand and improve learning outcomes. V. Hegde and S. Rao H.S. [17] presented an EDM-based framework to analyze students' performance in programming. The results of the analysis helped students to improve their weak concepts through frequent faculty support and also offered benefits to institutions. Ang *et al.* [18] conducted a comprehensive survey and presented architectures and their challenges for growing big educational data.

On the other hand, learning analytics (LA) refers to the collection, analysis, and visualization of educational data to understand and improve the learning processes and outcomes better. LA provides interventions based on the analysis of educational data to improve both learning and the learning environment [19]. Also, LA encompasses broader components of other disciplines such as EDM, academic analytics, learning sciences, cognitive sciences, human factors, psychology, and so on. Maher *et al.* [20] proposed a Personalized Adaptive Gamified E-learning (PAGE) model to enhance MOOCs LA and visualization in the learning process. The PAGE model helped learners in learning adaptation and visualization.

In this research, our goal is to investigate the impact of practical skills on academic performance through a comprehensive analysis using real-world e-learning data. Considering the context of this study, these two important terms such as practical skills and academic performance are defined as follows.

Practical skills relate to reasoning, critical-thinking, problem-solving, and implementation skills. Let consider a basic programming course that consists of two learning activities such as theory-based and practice-based. The practice-based activities include programming, programming-related assignments, and coding tests. In this research, performance in practice-based activities is referred to as practical skills. On the other hand, academic performance refers to theoretical knowledge, innovative ideas, and memorization. Performance in various theory-based activities includes algorithmic-based assignments, theory-based assignments, and paper-based tests which are referred to as academic performance.

To accomplish this study, a novel framework is proposed to extract students' hidden features and association rules. Hidden features derived from submission logs and scores carry significant meaning. This work makes the following contributions:

- We present the correlation between practical skills and academic performance.
- We find students' programming and academic weaknesses and strengths through empirical analysis using submission logs and scores. Further, necessary recommendations have been provided accordingly.
- We extract important and relevant features from the submission logs and scores that are not clearly visible in a simple form of dataset.
- We determine that the hidden features are useful to students as well as teachers to achieve programming and academic goals.
- The proposed framework and its data analysis process can be useful for other related academic courses and

disciplines to discover hidden features/correlations in e-learning data. For example, this framework can be applied to a course that consists of theory and hands-on activities and collects resources/data, like a programming course.

The rest of the article is structured as follows. In Section II, the background and related works are presented. Section III describes the dataset and preprocessing. In Section IV, the proposed approach is presented. The experimental results are presented in Section V and discussed in Section VI. Section VII concludes the study with outlooks on the future works.

II. BACKGROUND AND RELATED WORKS

In this section, we briefly introduce OJ or APA systems and their applications in programming education. In addition, supervised and unsupervised learning algorithms, association rule mining (ARM) algorithms, educational data mining and learning analytics are also presented.

A. ONLINE PROGRAMMING LEARNING PLATFORM

OJ or APA systems are now widely used by educational institutions as academic learning tools in programming and other exercise-based classes. These platforms play an important role in improving students' programming skills, knowledge, and overall academic performance. The vast resources (e.g., code archives, submission logs, etc.) generated by these systems can help researchers to find students' flaws in programming and thus expands the scope of available improvements. As a result, numerous studies have focused on programming education, educational data mining, and data-driven analysis using resources from OJ or APA systems.

In [7], the authors used learning log data extracted from the M2B system. A recurrent neural network is used to predict student performance. This study showed that numerous useful hidden features can be extracted by analyzing the M2B system's data. Mekterovič *et al.* [12] proposed an APA system for conducting programming courses and created the educational software Edgar to automatically evaluate programming assignments and other programming-related tasks. Edgar provides a variety of services, including content writing, course administration, system monitoring, and troubleshooting. Furthermore, Edgar produces the results of various statistics in a visual format. APA systems provide many benefits for students as well as instructors. Meanwhile, a ranking system [21] based on student performance and quick responses has positively impacted programming learning. APA systems have extended the conventional use of the OJ systems for evaluating programming assignments and their use significantly stimulates students' interest in programming.

In [22], the authors extended the BOCA OJ system to improve its suitability for programming classes. The resulting PROBOCA project was used to aid classroom teachers. This method identifies problems by degree of difficulty, thus making it easier for teachers to match problems with each student's programming experience. Another study [23] presents a continuous programming assessment system for programming courses using automated assessment tools (AATs). A quantitative analysis was performed based on the relationship between the student and the AAT outcome. The submitted solutions are analyzed in depth using an AAT and judgments (either correct or incorrect) are provided. The experimental results showed that AATs help students to better understand computer programming. Lu *et al.* [24] presented programming education via an OJ system that has increased students' performance curve in programming and other academic activities. Their experimental results show that the OJ system enhanced performance levels, as well as stimulated students' interest throughout the year- or semester-long course.

Toledo *et al.* [25] presented a fuzzy recommender system for OJ programming that provides suggestions to learners regarding their upcoming problems based on their past performance in the OJ system. That method also provided useful information to students via recommendations. In [26], OJ programming problems were classified using two topic-modeling algorithms, latent dirichlet allocation and non-negative matrix factorization in order to extract relevant features from problem descriptions. The classification of OJ programming problems can help novice and advanced students to pick and solve appropriate problems.

Our approach differs from that of existing research by focusing on discovering hidden features from submission logs and scores to improve programming skills and academic performance. We also focus on finding the correlation between practical skills and academic performance based on the extracted hidden features. To the best of our knowledge, no study has been conducted to address this issue by using submission logs and scores.

B. SUPERVISED AND UNSUPERVISED LEARNING ALGORITHMS

Within the context of artificial intelligence and machine learning (ML), supervised and unsupervised learning algorithms are frequently used in real-world applications. In short, both input data and output labels are known in supervised learning (SL) algorithms. Formally, SL involves ML algorithms that are trained with known input data and associated output labels. Let $U = \{u_1, u_2, u_3, \ldots, u_n\}$ be the set of input data and $V = \{v_1, v_2, v_3, \ldots, v_n\}$ be the set of corresponding output labels of the input data U. Thus, the output function can be written as $V = f(U)$, where the output V depends on the input U and f is a mapping function. After training, the ML algorithm can predict the output label for all new input data. SL algorithms are divided into two categories such as classification and regression.

Classification is an SL approach that classifies a given set of data into classes. The classification model predicts the target class for a given data point. After training, the model predicts class names for data that it has not seen before. There are two types of classification in ML such as binary

classification (*true* or *false*) and multi-class classification. Typically, the evaluation of a classification model is done by computing the *precision*, *recall*, and *accuracy* scores. Examples of some classification algorithms include support vector machine, decision tree, random forest tree, artificial neural network, similarity learning, and k-nearest neighbor [27]. Similarly, regression is an SL approach used to predict the continuous output variable based on one or more independent (predictors) variables. Mainly, this approach is used for forecasting, time series modeling, prediction, and determining market trends. Examples of regression algorithms include linear regression, logistic regression, polynomial regression, decision tree regression, and random forest regression [27].

In contrast, unsupervised learning (USL) is a kind of ML algorithm in which models are trained with unlabeled datasets. The USL algorithm can group the data based on their similarity features by applying some mathematical procedures. The USL algorithms have the following advantages over SL including hidden feature extraction, useful insights, human-like learning, handling unlabeled and uncategorized data. USL algorithms are divided into two categories such as clustering and association. Clustering is a USL algorithm used to group data into clusters, similarity characteristics of the data in a group are high, on the other hand, there is a minimal similarities with the data of another group. Examples of clustering algorithms include k-means, k-medoids, Density-based Spatial Clustering of Applications with Noise (DBSCAN), Clustering Large Applications based on RANdomized Search (CLARANS), and Clustering Large Applications (CLARA). Association is a USL algorithm that is used to find relationships between items in a large database. This algorithm determines the set of items that co-occur in a database. For example, if three items M, N, and O exist in the database, the algorithm can generate patterns/rules that co-occur such as $M \longrightarrow N$, $(M \,\&\, N) \longrightarrow O$, and $N \longrightarrow M$. These patterns/rules are useful for analyzing market-basket, educational data, and so on. Examples of association algorithms include Apriori and frequent pattern (FP)-growth.

In [28], students have been classified by a clustering approach based on their learning behaviors. The clustering by fast search and finding of density peaks via heat diffusion (CFSFDP-HD) algorithm has achieved a better clustering performance than other clustering algorithms. The authors also proposed an e-learning system architecture that detects and responds to teaching content based on student learning capabilities. Tabanao *et al.* [29] proposed a method that classifies programmers using submission log data, such as compilation profiles, error profiles, compilation frequency, and error quotient profiles produced during an introductory programming course. This study identified correlations between the submission log data and the midterm examination scores of students.

In our dataset, the output labels are unknown because the submission logs and class performance scores have not yet output information that could be used for labeling (e.g., poor, good, very good, or genius). Accordingly, as it is necessary to select an algorithm that can group students based on their source code submission logs and class performance scores, we expected that a clustering approach would provide the best-suited solution to group the students from unlabeled datasets. The most commonly used and effective clustering approaches, such as k-means, k-medoids, DBSCAN, agglomerative hierarchical cluster tree, and other variations of k-means were found based on a review. The modified k-means clustering algorithm [30], which we found to be a robust, scalable, and effective tool, is a variant of the conventional k-means clustering algorithm.

C. ASSOCIATION RULE MINING ALGORITHMS

ARM algorithm is a USL algorithm used for data mining in big data. ARM was first proposed by Agrawal [31] and has since been used in many fields, such as educational data analysis, medical data analysis, market-basket analysis, and census data. Usually, ARM aims to find a set of cooccurring high-frequency items and extract the correlation among items from large dataset. Although the Apriori algorithm [31] is often used for data mining, many enhancements are proposed based on Apriori to improve performance and scalability, such as the sampling approach [32], hashing technique [33], dynamic counting [34], partitioning technique [35], and incremental mining [36]. Prior studies showed that the Apriori algorithm achieved significant results, but some methods also reported the worse results by generating a large number of candidate item sets, additional scans, etc.

Subsequently, a new algorithm called FP-growth was proposed without the leverage of candidate item set generation [37]. This method used a partitioning-based divide-and-conquer approach. Previous studies have shown that it significantly reduced the search space and time compared to Apriori [38]. Similarly, many extensions are added to the FP- growth algorithm to improve efficiency. Some examples of enhanced FP-growth algorithms are h-mine [39], depth-first mining [40], pattern-growth mining in both directions (bottom-up and top-down), and tree structures [41], [42]. In contrast, Zaki [43] proposed the Equivalence CLASS Transformation (Eclat) algorithm for ARM applied to vertical data. The Eclat uses the same candidate-generation process like Apriori. In brief, Apriori, FP-growth, and Eclat ARM algorithms are most frequently used in many applications, and also serve as the foundation of many other ARM algorithms. In this research, an FP-growth algorithm is used.

D. EDUCATIONAL DATA MINING AND LEARNING ANALYTICS

EDM is the same as traditional data mining, except that it is applied to educational fields. EDM is used to extract hidden knowledge and discover patterns from the data in different educational learning platforms [44]. In the study [44], various data mining techniques including clustering, classification, and ARM are exploited to discover useful information from the educational data. They used EDM tools (Rapid Miner and Weka) to analyze data from Moodle in a programming

course. Fernandes *et al.* [45] presented a predictive analysis of students' academic performance. The Gradient Boosting Machine (GBM) classification model was applied to predict students' academic performance at the end of the school year. In another study [46], a semi-supervised learning algorithm was used to predict the students' performance in the final exams. In the study [47], a survey of EDM and its future directions is presented. It also discusses some recent trends in the field of EDM research.

LA has become an important research topic in the field of educational technology. This involves understanding and analyzing real-world educational data to provide useful support for improving learning and teaching. Tran *et al.* [48] used LA for a learning management system (LMS). The experimental results showed that LA plays an important role in improving productivity, learning, and support for LMS user. Ang *et al.* [18] discussed LA from five different perspectives: learning and assessment analysis, personalized learning, behavior learning, collaborative and interactive learning, and social network analytics. Current LA trends and practices to improve teaching and learning in education are presented in [49], [50].

In addition, numerous studies have been conducted using the resources of OJ or APA systems. These systems are actively used for education, e-learning, computing, programming competitions, and software engineering. The importance of empirical data-driven analysis to make critical decisions, and even to change algorithm configurations automatically, is growing [51]. However, this data-driven analytical research differs from previous studies in that a real-world dataset has been used. The analytical findings of this research are beneficial to assist students in improving their academic and practical performance, as well as for educational planning.

III. DATASET AND PREPROCESSING

In this section, we introduce the Aizu Online Judge (AOJ) system, which is the source of the submission logs. Moreover, we describe submission logs and class performance scores collected from the AOJ and a programming course, respectively as our datasets. The data types, structure, and preprocessing steps are also presented.

A. AIZU ONLINE JUDGE SYSTEM

The AOJ system [52], [53] is a popular OJ platform in Japan and worldwide. It has been running for more than 15 years to host programming competitions, practices, assignments, and education. In addition, the AOJ system is officially employed to conduct programming- and algorithm-related courses at the University of Aizu, Japan. The AOJ's typical courses include Introduction to Programming I (ITP1), Algorithms and Data Structures I (ALDS1), Introduction to Programming II (ITP2), Datasets and Queries, Discrete Optimization Problems, Graph Algorithms, Computational Geometry, and Number Theory. Thus, plenty of source codes and submission logs are generated on a regular basis. AOJ has a rich repository with approximately 100,000 users, 3,000 problems, and 5.5 million code archives and submission logs. All the problems are systematically categorized [54]. The AOJ's resources have been used for various research and application purposes [55], [56]. Recently, AOJ's dataset has been used in the IBM CodeNet Project [57].

TABLE 1. Topic-wise problem list of ALDS1 course.

No.	Topic	Theme	Average Success (%)
1	Getting Started	Insertion Sort Greatest Common Divisor Prime Numbers Maximum Profit	36.81
2	Sort I	Bubble Sort Selection Sort Stable Sort Shell Sort	44.92
3	Elementary Data Structures	Stack Queue Doubly Linked List Application of Stack	39.57
4	Search	Linear Search Binary Search Dictionary Application of Binary Search	40.10
5	Recursion/ Divide and Conquer	Exhaustive Search Merge Sort Koch Curve The Number of Inversions	42.74
6	Sort II	Counting Sort Partition Quick Sort Minimum Cost Sort	43.18
7	Tree	Rooted Trees Binary Trees Tree Walk Reconstruction of a Tree	43.95
8	Binary Search Trees	Binary Search Tree-I Binary Search Tree-II Binary Search Tree-III Treap	54.96
9	Heaps	Complete Binary Tree Maximum Heap Priority Queue Heap Sort	38.43
10	Dynamic Programming	Fibonacci Number Matrix Chain Multiplication LCS of Strings Optimal Binary Search Tree	51.82
11	Graph I	Graph Depth First Search Breadth First Search Connected Components	47.16
12	Graph II	Minimum Spanning Tree Single Source Shortest Path I Single Source Shortest Path II	54.38
13	Heuristic Search	8 Queens Problem 8 Puzzle 15 Puzzle	45.76

B. SOLUTION SUBMISSION LOGS

In this research, submission logs from a programming course (ALDS1) were collected for experiments. Usually, the problems in the ALDS1 course are assigned to students to solve, as shown in Table 1. The overall topic-wise success rate (%) of this course is also mentioned in Table 1. The ALDS1 course has 13 topics, and each topic consists of

TABLE 2. Sample submission logs generated by the AOJ system.

UID	JID	P	Verd	Lang	CPU time (1/100 s)	Memory usage (KB)	Code size (Byte)	Submission date (ms)	Judge date (ms)
u_1	3573821	p_1	RTE	Python3	0	0	71	1557999872496	1557999872657
u_2	3574251	p_2	AC	C++11	0	3444	1885	1558007384660	1558007386063
u_3	3573537	p_3	CE	C++	1	3404	1684	1557998419203	1557998420148
u_4	3556699	p_4	WA	C++	2	3104	1708	1557482997985	1557482998719
u_5	3536901	p_5	TLE	C++	399	3280	2199	1556795320402	1556795326021
u_4	3383318	p_{10}	AC	C	29	5748	936	1550129425882	1550129426378

three (03) or four (04) problems which we call problems *A*, *B*, *C*, and *D*.

The logs are generated by the AOJ system based on the submitted solution codes by the students over two semesters, the size of the submission logs is approximately 69,000. Each solution log has a set of information, such as the judge id (*jid*), user id (*uid*), problem id (*pid*), language (C, C++, python, etc.), accuracy, verdict (accepted, wrong answer, compile error, etc.), CPU time, memory usage, code size, submission date, and judge date. Let *UID* be a set of users (students) i.e., $UID = \{uid_1, uid_2, \ldots, uid_n\}$, $n \geq 1$. *JID* is a set of judge IDs $JID = \{jid_1, jid_2, \ldots, jid_m\}$, $m \geq 1$; *Prob* is a set of problems $Prob = \{prob_1, prob_2, \ldots, prob_j\}$, $j \geq 1$ where $prob_1, prob_2, \ldots prob_j$ are unique problems; judge verdicts are $Verd = \{AC, WA, CE, RTE, MLE, TLE, OLE, PrE\}$ where $AC =$ Accepted, $WA =$ Wrong Answer, $CE =$ Compile Error, $RTE =$ Run Time Error, $TLE =$ Time Limit Exceeded, $OLE =$ Output Limit Exceeded, $MLE =$ Memory Limit Exceeded, and $PrE =$ Presentation Error; and the programming languages are $Lang = \{C, C++, C++11, Ruby, Python\ 2, Python\ 3, Java, Haskell, C\#, PHP, Rust, \ldots\}$. A corresponding submission log is created immediately after submitting a solution code to the AOJ system. Thus, a sample output log of AOJ system can be written as $O_{logs} = \{u_r, j_s, p_t, v_u, l_v, ct, mu, cs, sd, jd\}$, where $u_r \in UID, j_s \in JID, p_t \in Prob, v_u \in Verd, l_v \in Lang$, $ct =$ CPU time, $mu =$ Memory usage, $cs =$ Code size, $sd =$ Submission date, $jd =$ Judge date. Some sample logs generated by the AOJ system are listed in Table 2.

C. CLASS PERFORMANCE SCORES

In addition to the submission logs, we collected various test (exam) scores for the ALDS1 course from 357 students in two different years at the University of Aizu, Japan. Usually, most students take the ALDS1 course as part of their regular study. This course consists of various tests, such as algorithm assignment (*AA*), programming assignment (*PA*), code validation (*CVal*), coding test (*CoT*), and paper-based test (*PT*). Note that *PA* and *CVal* are calculated based on the student's program submission to the AOJ system. To check the plagiarism/similarity/duplication of submitted solution codes, a plagiarism checking software (PCS) has been developed and integrated into the management system of AOJ. The PCS checks solution codes submitted by the students against the existing source codes in the AOJ. The PCS generates a *CVal* score for submitted codes based on the degree of similarity, and the codes are collected from a specific time period and users. A score of 1 means that there is no copying/duplication, 0.5 means that a few codes are copied from others, and 0 means that a number of codes are copied from others. In addition, *CVal* is used to justify the scores of *PA*. Some sample data distribution of student evaluations are listed in Table 3.

TABLE 3. Sample data distribution of student evaluations.

UID	AT	AA	PA	CVal	PT	CoT	T	Prac
u_1	12	85	90	1	80	75	82.5	82.2
u_2	10	75	85	0.5	85	70	79.8	38.6
u_3	11	80	100	1	75	90	77.5	94.9
u_4	13	90	110	0.5	90	110	90.0	55.0
u_5	9	65	70	0.5	75	60	69.8	32.4
u_6	13	95	105	1	90	100	92.5	102.5
u_7	10	78	85	0	60	70	68.4	0.0

Definition 1: The *CVal* score refers to the degree/level of program code plagiarism.

Example 1: If a user u_{15} copies/replicates programs from others, then u_{15} receives a *CVal* score of 0.5; if user u_{15} copies/replicates the code from others with malicious intent, *CVal* is 0.

Each exercise class is divided into two parts. First, students are asked to submit an *AA*, which consists of a few questions and is also considered student attendance (*AT*). Second, three or four problems are given to the students as a *PA*. The students are then encouraged to submit their solutions through the AOJ system. Students are allowed to consult with each other, teachers, and teaching assistants to solve problems during *PA*. In contrast, *CoT* is conducted in exercise rooms with a separate workstation for each student, providing a process by which each student's actual programming capabilities can be verified. Note that it is strictly forbidden for a student to consult with other students during the *CoT*. Similarly, the *PT* is a closed-book test that is given to check the true level of each student's theoretical understanding. The test scores distribution can be expressed as $T_{score} = \{UID, AT, AA, PA, CVal, PT, CoT, T, Prac\}$, where $AT \in \mathbb{N}$ and $0 \leq AT \leq 13$, $AA \in \mathbb{N}$ and $0 \leq AA \leq 100$, $PA \in \mathbb{N}$ and $0 \leq PA \leq 120$, $CVal \in \mathbb{R}$ and $0 \leq CVal \leq 1$, $PT \in \mathbb{N}$ and $0 \leq PT \leq 120$, $CoT \in \mathbb{N}$ and $0 \leq CoT \leq 120$, $T \in \mathbb{N}$ and $0 \leq T \leq 120$, $Prac \in \mathbb{N}$ and $0 \leq Prac \leq 120$. To better evaluate the students of the ALDS1 course

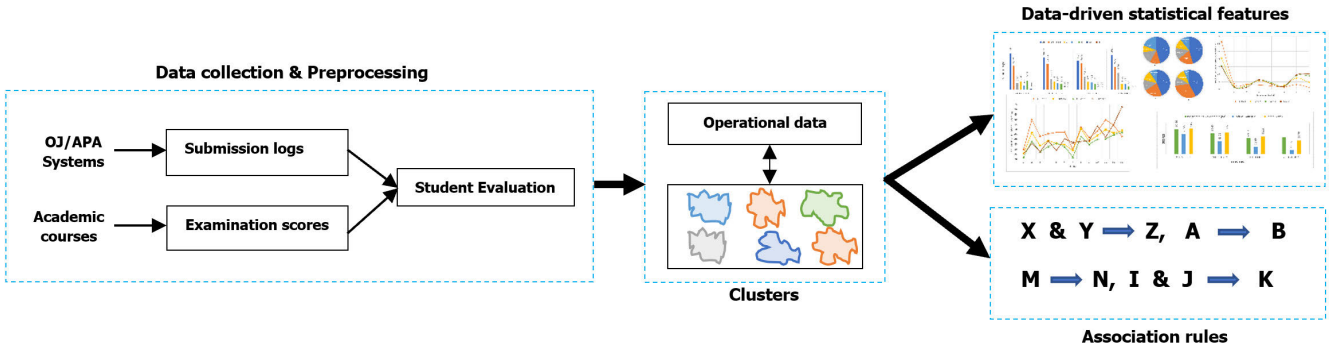

FIGURE 1. The overall framework of the data-driven approach.

by considering the importance of theoretical and practical knowledge, the equations (1), (2), and (3) are developed for the Theory (T), Practical ($Prac$) and Final Score (FS) calculations, respectively, based on the different test scores. Note that the equations for the ALDS1 course are approved by the course coodinator.

$$T = \sqrt{AA \times PT} \tag{1}$$

$$Prac = \sqrt{(PA \times CoT)} \times CVal \tag{2}$$

$$FS = min(100, \lfloor \frac{AT+1}{10} \rfloor \times \sqrt{(T \times Prac)}) \tag{3}$$

For explanation, the student evaluation process is compared using the following two scenarios: (*i*) the conventional case and (*ii*) the proposed case (based on the equations).

1) CONVENTIONAL CASE

In this case, the final results are usually generated by averaging $Prac$ and T scores. For example, if student $s1$ gets 10 points on the $Prac$ test and 90 points on the T test, the final result of $s1$ using the conventional method is $(10 + 90)/2 = 50$.

2) PROPOSED CASE

In this case, $Prac$ and T scores are given equal priority to generate final results, so the equations $(1 - 3)$ are introduced to emphasize both the $Prac$ and T scores. Let us assume that if student $s1$ gets 10 points on the $Prac$ test and 90 points on the T test, the final result of $s1$ using our equations will be $\sqrt{10 \times 90} = 30$. We observed that the proposed evaluation method considers both the $Prac$ and T scores, although there is no balance between the $Prac$ and T scores when calculating the final result using the conventional method.

For statistical feature extraction and ARM, Tables 2 and 3 are joined ($O_{logs} \bowtie T_{scores}$) to produce the operational data, as shown in Table 4. In addition to the existing attributes, a new attribute (*Accuracy*) has been added to the operational data.

Definition 2: The number of accepted solutions out of total submissions is called the solution *Accuracy* of users.

$$Accuracy(Accu) = \frac{\sum_{i=1}^{p_n} TAS_i}{\sum_{i=1}^{p_n} TS_i} \tag{4}$$

where p_n = number of problems, TAS = total accepted solutions, and TS = total submissions.

Example 2: Let u_5 be a user who has submitted a total of 39 solutions to the AOJ system, of which, a total of 28 have been accepted. Then, the solution accuracy of the user u_5 is $(28/39) = 71\%$, according to (4).

Another important term is trial and error ($T\&E$), we use the $T\&E$ method to estimate a programmer's ability to solve problems. In this study, the following definition is adopted for the $T\&E$ method.

Definition 3: A number of repeated attempts are taken until a problem is successfully solved; this process is called trial and error ($T\&E$).

$$T\&E = \frac{\sum_{j=1}^{p_n} TS_j}{\sum_{j=1}^{p_n} TAS_j} \tag{5}$$

where p_n = number of problems, TS = total submissions, and TAS = total accepted solutions.

Example 3: Suppose that u_{10} is a user who has received a total of 25 accepted (AC) verdicts from the AOJ for 5 problems, but has taken a total of 129 attempts ($T\&E$) to achieve it. Then, the average $T\&E$ of user u_{10} for each solved problem is $(129/25) = 5.16$, according to (5).

IV. APPROACH

Figure 1 shows the framework of our proposed educational data-driven approach. We employed the framework to a real-world dataset to extract the hidden features and association rules of students to explore the importance of practical skills. Experimental data are collected from AOJ system and ALDS1 programming course, respectively. The proposed approach consists of four main steps: (*i*) data collections and preprocessing, (*ii*) data clustering, (*iii*) statistical hidden features extraction from clusters, and (*iv*) association rules mining from clusters. A modified k-means clustering algorithm is applied for data clustering, where the elbow method used to select the optimal k values for the k-means. Furthermore, the FP-growth ARM algorithm is leveraged to extract the association rules from each cluster. The methods and algorithms used for the proposed approach are discussed below.

TABLE 4. Sample operational data distributions by joining submission logs (Table 2) and evaluation scores (Table 3).

UID	P	Verd	Accu (%)	mu	cs	CVal	PA	CoT	PT
u_1	p_1	AC	70	1164	116	1	90	75	80
u_2	p_2	WA	65	1072	125	0.5	85	70	85
u_3	p_7	RTE	84	1124	239	0.5	100	90	75
u_1	p_2	TLE	70	1064	96	1	90	75	80
u_5	p_1	AC	88	1143	209	0.5	70	60	75

A. ELBOW METHOD

The elbow method is a proven technique to determine the optimal number of clusters k for the k-means algorithm. It uses the sum of squared errors (SSE) of each cluster to calculate the optimal number of clusters. The SSE is calculated by the equation (6).

$$SSE = \sum_{i=1}^{k} \sum_{x \in C_i} dist^2(m_i, x) \tag{6}$$

where k is a number of clusters, x is a data point in cluster C_i, and m_i is the center of cluster C_i.

The elbow method reduces unnecessary clustering in the dataset, where a small SSE value indicates a better cluster. Normally, increasing the value of k automatically decreases the SSE value. When the SSE value is drastically decreased, that point is caught as the ideal number (k) of clusters for k-means. The elbow method was applied to our dataset to obtain the optimal k value ($k = 4$) for the k-means clustering algorithm, as shown in Figure 2.

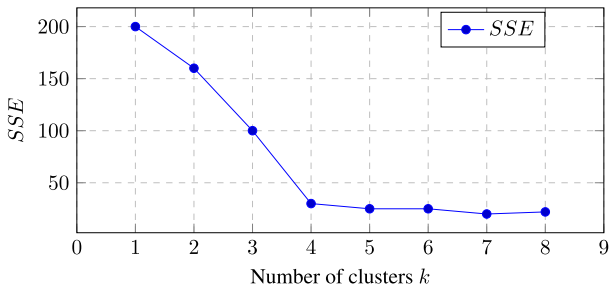

FIGURE 2. Elbow method for optimal k value selection.

B. MODIFIED K-MEANS CLUSTERING ALGORITHM

Usually, the k-means clustering algorithm randomly chooses the initial centroid, so it is possible to select an irrelevant data point as the initial centroid. In addition, conventional k-means algorithms cannot detect and remove outliers from the dataset. Consequently, the results may have a negative impact on the overall clustering process and results. To address these problems, the modified k-means clustering algorithm [30] integrates two important modules, (*i*) optimal initial centroid selection and (*ii*) outlier detection and removal. To the best of our knowledge, this is a unique modification of the k-means clustering algorithm, and these two modules makes the algorithm more efficient, robust, and scalable. This algorithm takes approximately 17.33% fewer iterations to construct clusters for our dataset than other random initial centroid-selection algorithms. The first module is initial centroid selection module (ICSM) which leverages to (*i*) select optimal centroids and (*ii*) build clusters with the most similar data. The pseudocode of ICSM is provided in Algorithm 1.

Algorithm 1 Initial Centroid Selection Module (ICSM)

[h] **Define**: Distance: D, Origin: $O(0, 0)$, Cluster Number: K
Input: Dataset: $X = \{x_1, x_2, x_3, \ldots\ldots\ldots, x_n\}$
Output: Optimal initial centroids C_n=[]
for $x_j \in X$ **do**
  $D \longleftarrow distance(x_j, O)$
end
for $d_i \in D$ **do**
  Apply sorting on D
  $D \longleftarrow d_1, d_2, d_3, \ldots, d_n$
end
if $K \leq |X|$ **then**
  Divide sorted data D into K subsets
  $s_1 \subseteq D, s_2 \subseteq D, s_3 \subseteq D, s_4 \subseteq D, \ldots, s_k \subseteq D$
end
while $k \leq K$ **do**
  Calculate **Mean** value of each subset
  $M_k = \frac{\sum_{x_x \in S_k}}{|S_k|}$
  for $x_j \in S_k$ **do**
    $C_n \longleftarrow mindistance(M_k, x_{j \in S_k})$
  end
end

The second module is the outlier detection module (ODM), which is used to (*i*) detect outliers (irrelevant/insignificant data point), (*ii*) remove them from the datasets, and (*iii*) improve the overall cluster quality. The pseudocode of the ODM is presented in Algorithm 2.

C. FP-GROWTH ALGORITHM

In the field of data mining, the Apriori, Eclat, and FP-growth algorithms are the most commonly used [58]. The FP-growth algorithm is much more efficient and faster than Apriori because the Apriori algorithm repeatedly scans the database, whereas the FP-growth algorithm only scans twice to complete the process. The FP-growth algorithm basically consists of two (2) main steps [37], namely (*i*) construction of the FP-tree and (*ii*) FP mining based on the FP-tree. Let $L = \{l_1, l_2, l_3, l_4, \ldots, l_d\}$ be the set of all items in the database. The databases are built based on a set of tuples/transactions

Algorithm 2 Outlier Detection Module (ODM)

Define: Number of Cluster: K, Cluster: C
Input: Dataset: $X = \{x_1, x_2, x_3, \ldots\ldots\ldots, x_n\}$, Distance: D
Output: Outliers: O, SSE
Run **ICSM** and Calculate **min-max average (MMA)** using *sorted* distance $d \in D$
$MMA = \frac{d_{min}+d_{max}}{2}$
while $k \leq K$ **do**
  for $x_i \in X_{C_k}$ **do**
    if $distance(x_i, center_{C_k}) > MMA$ **then**
      Remove x_i from the cluster C_k
      $O \longleftarrow x_i$
      Recalculate SSE
    end
  end
end

$T = \{t_1, t_2, t_3, t_4, \ldots, t_N\}$, where each transaction t_i is a subset of $L (t_i \subseteq L)$. The formula of an association rule can be written as $R = X \longrightarrow Y$, where X, Y is a subset of L ($X \subseteq L, Y \subseteq L$) and $X \cap Y = \phi$. The set of items in X is often called the preceding (*if*), and the set of items in Y is called the subsequent (*then*). Mathematically, the support count for item set X is expressed as $\varsigma(X) = |\{t_i | X \subseteq t_i, t_i \in T\}|$, where $|.|$ denotes the number of elements in a set. The minimum support (*minSup*) and minimum confidence (*minConf*) are two important terms that are used to create association rules. The *minSup* threshold is used to find item frequencies in a database, whereas the *minConf* threshold value is applied to these frequent items to construct the association rules. The support (*Sup*) and confidence (*Con*) are represented by the equations (7) and (8), respectively.

$$Sup(X \longrightarrow Y) = \frac{\varsigma(X \cup Y)}{N} \tag{7}$$

$$Con(X \longrightarrow Y) = \frac{\varsigma(X \cup Y)}{\varsigma(X)} \tag{8}$$

where $N =$ total number of transactions.

V. EXPERIMENTAL RESULTS

In this section, the experimental results are presented. We first cluster students based on their submission logs and scores, and then extracted the hidden features from each cluster. The association rules are generated from each cluster using the FP-growth algorithm to validate the features. Finally, all the correlated features are accumulated for discussion.

A. CLUSTERING THE DATA

According to the proposed framework (Figure 1), the modified k-means clustering algorithm is applied to the Table 3 for the clustering process. Before clustering begins, the elbow method is applied to the same data to generate the optimal number ($k = 4$) of clusters, as shown in Figure 2. Now, four clusters have been formed, named clusters P, Q, R, and S. Note that multidimensional data (Table 3) are clustered. To visualize the data distribution of each cluster, we applied principal component analysis (PCA) technique to multidimensional clustered data to convert it into a two-dimensional (2D) shape. For this reason, the first two components (PCA 1 and PCA 2) of the PCA that explain the majority of the variance in the data are used for the 2D visualization. The visualized clusters are shown in Figure 3.

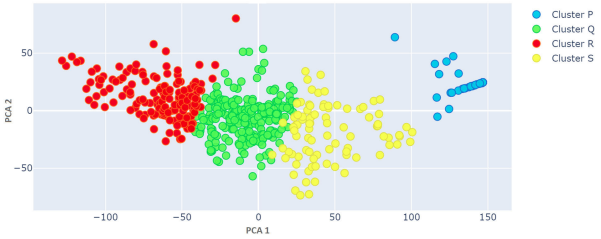

FIGURE 3. 2D visualization of the clusters.

First, the preliminary statistical information related to each cluster, (*i*) the number of students per cluster and (*ii*) the total number of problems solved by the students in each cluster, is presented in Table 5. We found that approximately 33.33% of the students are in cluster Q, which is the largest, and approximately 16.01% of the students are in cluster P, which is the smallest. On the other hand, the students in cluster Q generated the largest submission log of 22,110, and the students in cluster S produced the smallest submission log of 5,153.

TABLE 5. Preliminary statistical information of each cluster.

Cluster	Submission logs	Students (%)
P	13099	16.01
Q	22110	33.33
R	17274	32.02
S	5153	18.62

B. EXTRACTING HIDDEN FEATURES

In this section, different features of students are extracted from clusters P, Q, R, and S. We calculated the solution verdicts (considering problems A, B, C, and D) in each cluster, as denoted in Table 6. Each submission log contains at least one judge verdict out of many (*AC*, *WA*, *CE*, *etc.*). Therefore, each verdict determined the ultimate result of a submitted solution. A few observations can be illustrated from the Table 6: (*i*) clusters P and S have the highest *AC* rates, (*ii*) the students of cluster R achieved the lowest *AC* rates, and (*iii*) the students of cluster R received higher error verdicts than those in other clusters.

Also, we enumerated problem-wise statistics of the submitted solutions to find out how many submissions belong to each problem such as A, B, C, and D in each cluster, as presented in Table 7. A few observations can be drawn from the Table 7: (*i*) Students of cluster P submitted the fewest solutions to problem A, at 30.99%, compared to clusters Q, R, and S. (*ii*) Cluster P students submitted the highest number

TABLE 6. Overview of the judge verdicts of ALDS1 course.

Verdict	Cluster P	Cluster Q	Cluster R	Cluster S
AC (%)	40.88	36.34	32.70	39.19
WA(%)	26.71	28.92	29.46	25.69
CE (%)	7.37	11.95	14.88	18.59
RTE (%)	8.81	8.80	8.71	6.29
PrE (%)	4.76	7.02	7.94	6.46
TLE (%)	10.19	6.48	6.11	3.80
MLE (%)	1.20	0.44	0.16	0.00
OLE (%)	0.03	0.04	0.04	0.00

of solutions for problems C and D, at 32.82% and 10.55%, respectively, compared to clusters Q, R, and S. (*iii*) Students of clusters R and S submitted the fewest solutions for problems C and D, compared to clusters P and Q, respectively.

TABLE 7. Overview of the submission statistics for each type of problem.

Problem	Cluster P	Cluster Q	Cluster R	Cluster S
A	30.99%	39.17%	48.73%	49.19%
B	25.65%	30.28%	32.19%	29.09%
C	32.82%	25.04%	16.02%	18.11%
D	10.55%	5.51%	3.06%	3.61%

In contrast, the error verdicts of each cluster are also calculated. The segmentation of error verdicts received by the students in each cluster are shown in Figure 4. For that, the error verdicts are divided into five (05) categories based on the error types in codes such as (*i*) *WA*, (*ii*) *CE*, (*iii*) *RTE*, (*iv*) *PrE*, and (*v*) Resource Limitation (*TLE*, *MLE*, *OLE*) i.e., *RL*. Detailed error statistics for each cluster are presented in Figure 4.

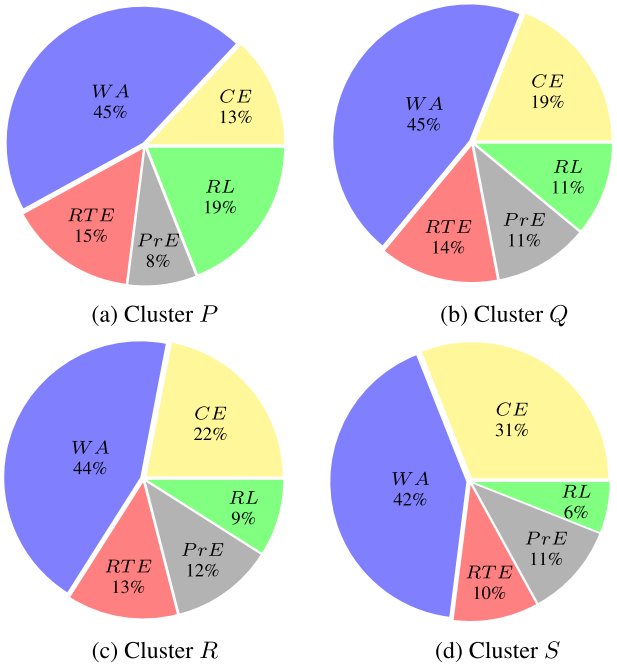

FIGURE 4. Segmentation of error verdicts received by the students.

Now, the solution *accuracy* and *T&E* are calculated for each cluster, as enumerated in Table 8. A few observations can be drawn: (*i*) the students of clusters P and Q have more *T&E* as well as higher solution *accuracy*, and (*ii*) both the solution *accuracy* and *T&E* of cluster R are lower than those of other clusters. Note that the solution *accuracy* and *T&E* are calculated by equations (4) and (5), respectively.

TABLE 8. Cluster-wise solution accuracy and problem-solving *T&E*.

Cluster	Avg. solution accuracy (%)	Avg. T&E
P	55.71	12.40
Q	48.45	10.14
R	43.15	9.52
S	45.18	9.86

Next, the average score and standard deviation (σ) for each cluster are calculated, as shown in Table 9 and the comparative views presented in Figure 5. Standard deviation (σ) is used to measure the variation of values in a cluster. Thus, a low value of σ indicates that the values are likely close to the mean (average). The following observations can be drawn: (*i*) the *CoT*, *PA*, and *PT* scores of cluster P are much higher than those of clusters Q, R, and S; (*ii*) the *PA*, *CoT*, and *PT* scores of cluster Q are higher than those of clusters R and S; (*iii*) the *CoT* score of cluster R is comparatively lower than the other *PA* and *PT* scores of this cluster; (*iv*) the *CoT* score of cluster S is also much lower than those of clusters P, Q, and R.

TABLE 9. Overview of the average scores and standard deviation (σ) in each cluster.

Cluster	PA	σ-PA	CoT	σ-CoT	PT	σ-PT
P	97.46	13.95	78.55	16.70	98.79	11.67
Q	81.43	15.97	48.21	12.61	84.51	15.15
R	60.92	17.80	28.46	10.68	68.26	16.60
S	65.19	34.96	16.55	14.48	53.79	30.05

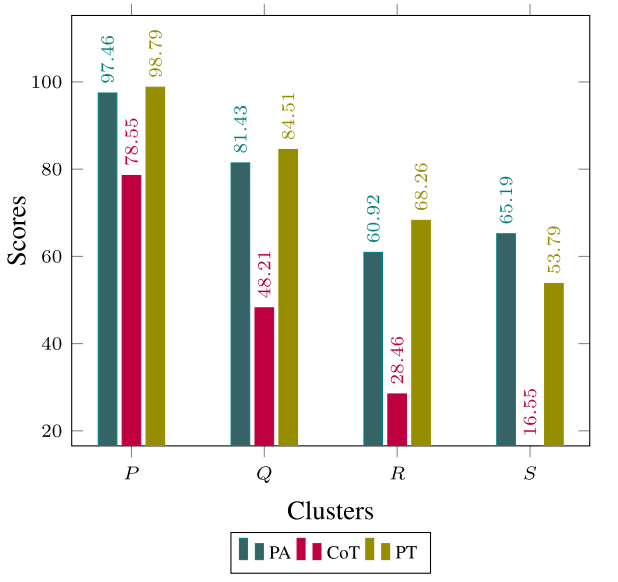

FIGURE 5. Comparison of scores in different tests.

We found more interesting features from the clusters. For example, students solved numerous additional problems beyond their regular exercise assignments through the AOJ platform, solely for their own interests and amusement. The cluster-wise extra problem solution statistics are listed in Table 10. The following observations can be drawn from the Table 10: (*i*) the students of cluster *P* solved a huge number of problems beyond their regular exercise assignments, which clearly indicates their enthusiasm for programming, and (*ii*) the students in other clusters (*Q*, *R*, and *S*) did not solve a significant number of extra problems.

TABLE 10. Statistics of extra problem solutions.

Cluster	Average number of extra problem solutions
P	56.14
Q	0.68
R	2.31
S	0.00

The tendency to submit each assignment in the ALDS1 course is analyzed for more information. There are a few rules to submit each *PA* task through the AOJ platform: (*i*) problems *A* and *B* must be solved by a certain predetermined deadline, where students usually have eight (08) days to submit each assignment, and (*ii*) problems *C* and *D* can be submitted by the end of the semester. One of our goals is to observe students' submission trends for each topic, how they submitted solutions to problems *A* and *B* within the allotted time, because problems *A* and *B* are mandatory for scoring. The average submission trend among all clusters over a period of time (08 days) is shown in Figure 6.

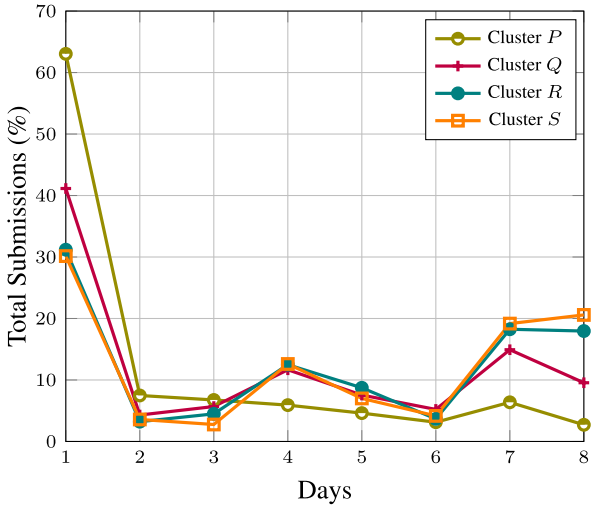

FIGURE 6. Tendency to submit assignments within the allotted period (08 days).

The following observations can be illustrated from the Figure 6: (*i*) the students of cluster *P* tried very hard to solve and submit their assignments (problems *A* and *B*) on the very first day of the submission period, (*ii*) the students of clusters *R* and *S* made less effort in submitting assignments during the first few days of the submission period compared to clusters *P* and *Q*, and (*iii*) more students from clusters *R* and *S* submitted their assignments on the last day (8^{th}) of the submission deadline than students from clusters *P* and *Q*.

Sometimes students submitted their solutions after the deadlines. The topic-wise accepted (*AC*) solution rate and average accepted (*AC*) rate for all clusters are calculated and listed in Table 11. Moreover, a visual comparison between all topics for all clusters is presented in Figure 7. A few observations can be found: (*i*) the students of cluster *P* received the highest acceptance against all their assignment submissions and (*ii*) the students of cluster *R* obtained the lowest acceptance rate compared to those in clusters *P*, *Q*, and *S*.

TABLE 11. Topic-wise average accepted (*AC*) solution rate.

Topic	Accepted Solution (%)			
	Cluster P	Cluster Q	Cluster R	Cluster S
1	29.70	25.60	22.04	25.24
2	60.03	46.71	38.19	35.73
3	43.10	33.65	27.37	27.57
4	45.56	38.41	32.71	38.43
5	47.10	35.81	36.12	28.69
6	47.35	34.62	32.62	40.51
7	29.87	28.74	22.08	36.57
8	52.04	50.55	42.62	37.77
9	41.78	38.10	35.07	38.67
10	48.11	43.75	40.10	53.68
11	59.70	49.82	40.38	42.27
12	54.15	46.29	44.59	55.56
13	42.95	48.71	47.25	72.95
Average	**46.24**	**40.06**	**35.47**	**41.05**

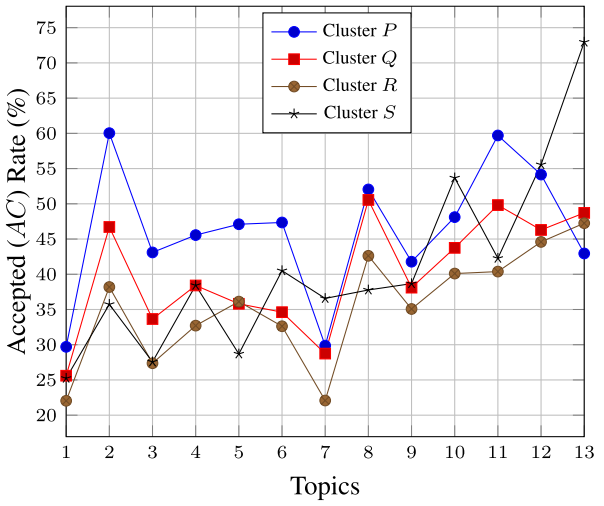

FIGURE 7. Comparison of topic-wise accepted (*AC*) rate.

We also analyzed the data across all clusters to find the assignment submission trends on the last (8^{th}) day of the allotted time. A comparative analysis of the tendency to submit assignments on the last day of each topic is presented in Figure 8. The average submission rate on the last day is also calculated across all topics, as shown in Table 12. It can be observed that among all clusters, (*i*) students of

clusters *R* and *S* submitted most solutions on the last day and (*ii*) students of cluster *P* submitted the fewest solutions on the last day.

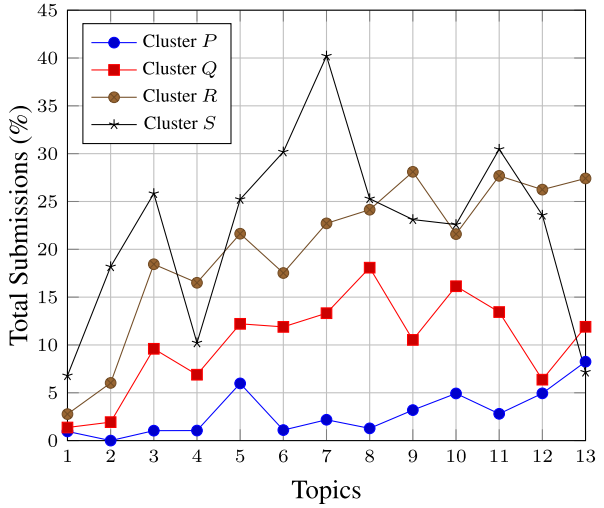

FIGURE 8. Tendency to submit assignments on the last day.

TABLE 12. Average submission rate on the last day.

Cluster	Average submission rate (%)
P	2.90
Q	10.28
R	20.06
S	22.22

To obtain more interesting hidden features that are not plainly visible in the dataset, we analyzed the data of each cluster and found that many students repeatedly solved problems (already accepted) for optimization in terms of memory usage, CPU time, code refactoring, etc. The repetition tendency (only for accepted problems) of students in each cluster is calculated. The ALDS1 course has thirteen topics, each with four problems (*A*, *B*, *C* and *D*), for a total of 52 unique problems (total problems $TP = 52$). We determined how many students in each cluster repeatedly solved 25%, 50%, and 75% of the *TP*, as enumerated in Table 13. The students participation (maximum and minimum) from each cluster are enumerated as follows: (*i*) 92.86% students of cluster *P* repeatedly solved 25% of the *TP* whereas 28.21% of students of cluster *S* participated, which is the lowest; (*ii*) for 50% *TP* repetition, 44.05% and 24.16% of students from clusters *P* and *Q* participated, respectively; and (*iii*) for 75% *TP* repetition, 9.52% students participated from cluster *P*, which is the largest and no students (0%) participated from clusters *R* and *S*.

Here, the *CVal* scores are calculated for each cluster, with average scores of 1, 0.96, 0.96, and 0.44 for clusters *P*, *Q*, *R*, and *S*, respectively. The students in cluster *S* received an average *CVal* score of 0.44, indicating poor coding skills. According to the Definition 1, they likely copied someone else's code to solve the assignments. In contrast, the students of clusters *P*, *Q*, and *R* obtained higher *CVal* scores.

TABLE 13. Repetition tendency of accepted problems.

Problems	Number of students (%)			
	Cluster P	Cluster Q	Cluster R	Cluster S
25% of TP	92.86	80.90	63.16	28.21
50% of TP	44.05	24.16	5.85	5.13
75% of TP	9.52	3.37	0	0

C. ASSOCIATION RULE MINING

The FP-growth algorithm is applied to the clustered data to find the association rules, which help identify the actual relationship between programming skills and academic performance. In addition, the association rules are used to verify the extracted statistical features of each cluster.

TABLE 14. Dictionary for the set attributes.

catId	Categorization of values	Attributes
1-59	$prob_1, prob_2, prob_3 \cdots prob_{57}$	Problems ($Prob$)
60	$\geq 75\%$	Accuracy ($Accu$)
61	60%-74%	
62	45%-59%	
63	<45%	
70	Accepted (AC)	Verdicts ($Verd$)
71	Compile Error (CE)	
72	Memory Limit Exceeded (MLE)	
73	Output Limit Exceeded (OLE)	
74	Presentation Error (PrE)	
75	Run Time Error (RTE)	
76	Time Limit Exceeded (TLE)	
77	Wrong Answer (WA)	
80	$\geq 80\%$	Programming Assignment(PA)
81	65%-79%	
82	45%-64%	
83	< 45%	
90	$\geq 80\%$	Coding Test (CoT)
91	65%-79%	
92	45%-64%	
93	< 45%	
100	$\geq 80\%$	Paper-based Test (PT)
101	65%-79%	
102	45%-64%	
103	< 45%	

Before the FP-growth algorithm is applied, we prepare each cluster's data in a uniform data format. Therefore, the prominent attributes such as *Prob*, *Accu*, *Verd*, *PA*, *CoT*, and *PT* are selected for ARM. Let $W = \{\{Prob\}, \{Accu\}, \{Verd\}, \{PA\}, \{CoT\}, \{PT\}\}$ be a set of attributes, where $Prob = \{\text{catId} \mid \text{catId} \in \mathbb{N}, 1 \leq \text{catId} \leq 59\}$, $Accu = \{\text{catId} \mid \text{catId} \in \mathbb{N}, 60 \leq \text{catId} \leq 69\}$, $Verd = \{\text{catId} \mid \text{catId} \in \mathbb{N}, 70 \leq \text{catId} \leq 79\}$, $PA = \{\text{catId} \mid \text{catId} \in \mathbb{N}, 80 \leq \text{catId} \leq 89\}$, $CoT = \{\text{catId} \mid \text{catId} \in \mathbb{N}, 90 \leq \text{catId} \leq 99\}$, and $PT = \{\text{catId} \mid \text{catId} \in \mathbb{N}, 100 \leq \text{catId} \leq 109\}$. Thus, the sets *Prob*, *Accu*, *Verd*, *PA*, *CoT*, and *PT* are a subset of *W*, i.e., $Prob \subseteq W, Accu \subseteq W, Verd \subseteq W, PA \subseteq W, CoT \subseteq W, PT \subseteq W$. The values of the elements in each set have been converted into uniform categorical IDs (catId) according to the definition in Table 14. After the cluster data is converted into uniform catId, the sample data formats of the tuples are as

follows: $W_1 = \{29, 60, 70, 81, 90, 100\}$, $W_2 = \{17, 61, 71, 80, 92, 102\}$, and $W_3 = \{32, 58, 70, 81, 91, 101\}$.

Interesting and relevant association rules are obtained from each cluster by setting the optimal minimum support (*minSup*) and confidence (*minConf*) threshold values. For cluster *P*, we set *minSup* = 1500 and *minConf* = 90%. Consequently, the frequent rules shown in Table 15 are obtained.

TABLE 15. Association rules for the students of cluster *P*.

Rules
R1: $PT \geq 80\%$ && $Verd == AC$ && $PA \geq 80\% \rightarrow Accu(\geq 75\%)$
R2: $PT(65\% - 79\%)$ && $Verd == AC \rightarrow Accu(\geq 75\%)$
R3: $PT \geq 80\%$ && $Verd == AC \rightarrow Accu(\geq 75\%)$
R4: $Verd == AC$ && $PA \geq 80\% \rightarrow Accu(\geq 75\%)$
R5: $Verd == AC$ && $CoT(45\% - 64\%) \rightarrow Accu(\geq 75\%)$

In Table 16, the association rules extracted from cluster *Q* using the values *minSup* = 2000 and *minConf* = 90% are listed.

TABLE 16. Association rules for the students of cluster *Q*.

Rules
R1: $PT(65\% - 79\%)$ && $Verd == AC \rightarrow Accu(\geq 75\%)$
R2: $PT(45\% - 64\%)$ && $Verd == AC \rightarrow Accu(\geq 75\%)$
R3: $Verd == AC$ && $PA(65\% - 79\%)$ && $CoT < 45\% \rightarrow Accu(\geq 75\%)$
R4: $PT(65\% - 79\%)$ && $Verd == AC$ && $CoT < 45\% \rightarrow Accu(\geq 75\%)$
R5: $Verd == AC$ && $CoT < 45\% \rightarrow Accu(\geq 75\%)$
R6: $Verd == AC$ && $CoT(45\% - 64\%) \rightarrow Accu(\geq 75\%)$
R7: $Verd == AC$ && $PA(65\% - 79\%) \rightarrow Accu(\geq 75\%)$
R8: $Verd == AC$ && $PA(45\% - 64\%) \rightarrow Accu(\geq 75\%)$
R9: $Verd == CE \rightarrow Accu(< 45\%)$

Similarly, valuable rules are also extracted from cluster *R* when *minSup* = 3000 and *minConf* = 90%. The generated rules are listed in Table 17.

TABLE 17. Association rules for the students of cluster *R*.

Rules
R1: $PT(45\% - 64\%)$ && $Accu < 45\% \rightarrow CoT < 45\%$
R2: $PT(65\% - 79\%) \rightarrow CoT < 45\%$
R3: $Verd == AC \rightarrow Accu \geq 75\%$ && $CoT < 45\%$
R4: $Accu < 45\% \rightarrow CoT < 45\%$
R5: $PT(45\% - 64\%)$ && $Accu \geq 75\% \rightarrow CoT < 45\%$
R6: $Accu \geq 75\%$ && $Verd == AC \rightarrow CoT < 45\%$
R7: $Accu < 45\%$ && $PA(45\% - 64\%) \rightarrow CoT < 45\%$
R8: $Accu < 45\%$ && $Verd == WA \rightarrow CoT < 45\%$
R9: $Accu < 45\%$ && $PA < 45\% \rightarrow CoT < 45\%$
R10: $PA < 45\% \rightarrow CoT < 45\%$
R11: $Verd == AC$ && $CoT < 45\% \rightarrow Accu(\geq 75\%)$
R12: $Accu \geq 75\%$ && $PA(45\% - 64\%) \rightarrow CoT < 45\%$
R13: $PT(45\% - 64\%)$ && $PA(45\% - 64\%) \rightarrow CoT < 45\%$
R14: $PT < 45\% \rightarrow CoT < 45\%$

Finally, rules are generated for cluster *S* when we set *minSup* = 1500 and *minConf* = 90%, as shown in Table 18.

The following observations can be obtained based on the association rules from different clusters: (*i*) in cluster *P*, association rules are involved with higher *accuracy*, *PA*, *PT*, and *CoT*, as well as the most frequent accepted (*AC*) verdicts; (*ii*) students of cluster *Q* showed with higher *accuracy*, *PA*, and *PT* but lower scores in *CoT*; (*iii*) students of cluster *R* tended to have lower scores in *CoT*, *PT*, and *PA*, as well as infrequent *AC* verdicts; and (*iv*) cluster *S* students showed lower scores in *CoT*, *PT*, and *PA*, as well as lower *accuracy*.

TABLE 18. Association rules for the students of cluster *S*.

Rules
R1: $Accu >= 75\% \rightarrow CoT < 45\%$
R2: $Verd == AC \rightarrow CoT < 45\%$
R3: $PT < 45\%$ && $Accu < 45\%$ && $PA < 45\% \rightarrow CoT < 45\%$
R4: $PT < 45\%$ && $Accu < 45\% \rightarrow CoT < 45\%$
R5: $PT < 45\%$ && $PA < 45\% \rightarrow CoT < 45\%$
R6: $Accu \geq 75\%$ && $Verd == AC \rightarrow CoT < 45\%$
R7: $PA < 45\%$ && $CoT < 45\% \rightarrow PT < 45\%$
R8: $Accu < 45\%$ && $PA < 45\% \rightarrow CoT < 45\%$
R9: $Verd == AC$ && $CoT < 45\% \rightarrow Accu(\geq 75\%)$
R10: $Accu < 45\% \rightarrow CoT < 45\%$
R11: $Accu < 45\%$ && $PA < 45\%$ && $CoT < 45\% \rightarrow PT < 45\%$

D. ACCUMULATION OF CORRELATED FEATURES

Many significant features are generated from each cluster by employing the proposed framework. These features are deeply correlated to each other and meaningful. These correlated features and rules are accumulated for each cluster.

Students of the cluster *P* (*i*) took an average problem-solving *T&E* of 12.40 (Table 8), (*ii*) solved an average of 56.14 extra problems beyond their academic assignments (Table 10), (*iii*) repeated more accepted (*AC*) solutions for optimization than those of other clusters (Table 13), (*iv*) submitted their assignments on the very first day more than those in other clusters (Figure 6), and (*v*) had the lowest average last-day submission rate of approximately 2.90% than clusters *Q*, *R*, and *S* (Table 12). These features are interdependent and deeply correlated to each other. The features mentioned above have interesting meanings; overall, these features indicate that students in this cluster are committed to programming, which has a positive impact on their programming and academic performance.

Cluster *P* also had an overall *AC* rate (considering problems *A*, *B*, *C*, and *D*) of 40.88% (Table 6), topic-wise *AC* rate (considering problems *A* and *B*) of 46.24% (Table 11), average solution *accuracy* of 55.71% (Table 8), and higher *CVal* score of 1. These higher success rates in programming enabled higher scores in *PA*, *CoT*, and *PT* that are also validated by the association rules (Table 15).

In cluster *Q* the overall *AC* rate considering all problems is 36.34% (Table 6) and the topic-wise average *AC* rate (considering problems *A* and *B*) is 40.06% (Table 11), which are lower than those of clusters *P* and *S*. The students of cluster *Q* consistently maintained a high *AC* rate throughout the thirteen topics (Figure 7) but solved minimal extra problems beyond their academic assignments (Table 10). They had a

lower tendency to submit solutions on the last day than students of clusters *R* and *S* instead submitted their assignments early (Table 12). The students of cluster *Q* obtained higher scores in *PA* and *PT* than in *CoT* (Table 9), as shown by the association rules (Table 16). Most of the features involved high values, indicating that they put a great effort into programming. However, the lower attempt to solve additional problems likely affected the *CoT* scores in this cluster.

Students of cluster *R* (*i*) took an average of 9.52 attempts/trials to solve problems, (*ii*) did not solve many additional problems outside of regular academic assignments (Table 10), (*iii*) submitted their assignments on the deadline or the day before (Figure 6), and (*iv*) rarely repeated the *AC* problems more than once (Table 13). These features are related to their various programming activities and indicate that they did not put much effort into programming. These students also received the highest error (*WA*, *CE*, *RTE*, *TLE*, etc.) verdicts of 67.30% and the lowest *AC* rate of 32.70% compared to clusters *P*, *Q*, and *S*. Their topic-wise *AC* rate is not coherent across all thirteen topics. Students in cluster *R* obtained good scores in *PA* (60.92) and *PT* (68.26), but lower scores in *CoT* (28.46) (Table 9). The association rules showed that students were involved with lower scores and infrequent *AC* verdicts. Note that the coding test (*CoT*) is used to verify the students' core programming skills. Thus, less effort in programming negatively affects this *CoT* score.

Students of cluster *S* (*i*) undertook an average of 9.86 attempts/trials to solve problems (Table 8) (*ii*) solved no additional problems (Table 10), (*iii*) had the highest rate of last-day submission for each assignment with an average of 22.22% solutions submitted on the last day (Figure 6 and Table 12) compared to clusters *P*, *Q*, and *R*, and (*iv*) obtained very low *CVal* score of 0.44. Furthermore, an insignificant number of students attempted to repeat the *AC* problems more than once (Table 13). Collectively, these features indicate that students in cluster *S* did not perform well in programming. Most features are negatively prioritized. Consequently, students in this cluster obtained the lowest scores in *CoT* (16.55), which is alarming for actual coding performance. Most of the association rules are connected with lower *CoT* scores (Table 18). In addition, we found an interesting correlation: most of the students submitted their solutions on the last day, but achieved higher *AC* rates and *accuracy*. This trend differs from that of clusters *P* and *Q*.

VI. DISCUSSION

In this study, many hidden features are obtained by employing the proposed framework, where modified k-means is applied for data clustering and then FP-growth is applied to the clustered data to discover the association rules. Interesting features and behaviors are observed that are not readily apparent in the base dataset. After applying the elbow and k-means algorithms to the dataset, four (04) clusters are found. Different features and rules are extracted from each cluster considering the different conditions presented in the experimental results section. Next, we discuss the features and the resulting explanations, recommendations, assessments, practical applications, and limitations.

TABLE 19. Main features.

No.	Features	Values
a	Trial and error ($T\&E$)	$H/M/L$
b	Extra problem solutions beyond academic assignments	$H/M/L$
c	Tendency to submit programming assignments	$E/M/D$
d	Topic-wise average last day submission	$H/M/L$
e	Number of students who repeatedly solved accepted (AC) problems for optimization	$H/M/L$
f	Accepted (AC) rate (both topic-wise and problems)	$H/M/L$
g	Overall accuracy ($Accu$)	$H/M/L$
h	Scores in programming assignment (PA)	$H/M/L$
i	Scores in coding test (CoT)	$H/M/L$
j	Scores in paper-based test (PT)	$H/M/L$

For a better understanding, ten main features are listed in Table 19 with three indicator values: higher (H), medium (M), and lower (L). Feature *c*, which indicates when the assignments are submitted within the alloted time, uses indicator values of early (E), mid-time (M), and delay (D).

A. ANALYSIS AND RECOMMENDATIONS

In the summary graph of the main features shown in Figure 9, we observe that the students of cluster *P* performed extraordinarily well in different programming activities and academic tests. Importantly, most students in this cluster are highly enthusiastic about programming, with more than 62% of total solutions (Figure 6) submitted on the very first day of all assignments. They also solved an average of 56.14 extra problems in addition to their academic assignments. The tendency to submit solutions on the last day is approximately 2.90% which is the lowest compared to clusters *Q*, *R*, and *S* (Table 12). For solution optimization, a large number of students repeated their *AC* solutions (Table 13). In addition, these students achieved higher *AC* rates of 46.24% for problems *A* and *B* (Table 11), *accuracy* of 55.71% (Table 8), and scores on various tests of 81.22%, 65.46%, and 82.33% for *PA*, *CoT*, and *PT*, respectively than those of clusters *Q*, *R*, and *S* (Table 9), as reflected by the association rules (Table 15). In contrast, the total error verdict is analyzed from this cluster, with approximately 45% error due to *WA*, 19% due to resource limitations (*TLE*, *MLE*, *OLE*), and 15% due to *RTE* (Figure 4).

Note that, to develop students' programming skills and ensure the efficiency of the source codes, several constraints are set for problems such as input and output limits/numbers, space and time complexity. In this case, a solution code must satisfy the set of constraints to be accepted, otherwise it receives error verdicts such as *TLE* and *MLE*. Figure 4 shows that students in clusters *Q*, *R*, and *S* received 10.87%, 9.32%, and 6.25% errors due to *TLE* and *MLE*, respectively. In contrast, the students of cluster *P* received about 19.28% errors due to *TLE* and *MLE*, which is the highest compared to clusters *Q*, *R*, and *S*. However, students in cluster *P* took about 12.40 attempts ($T\&E$) to solve a problem, which is

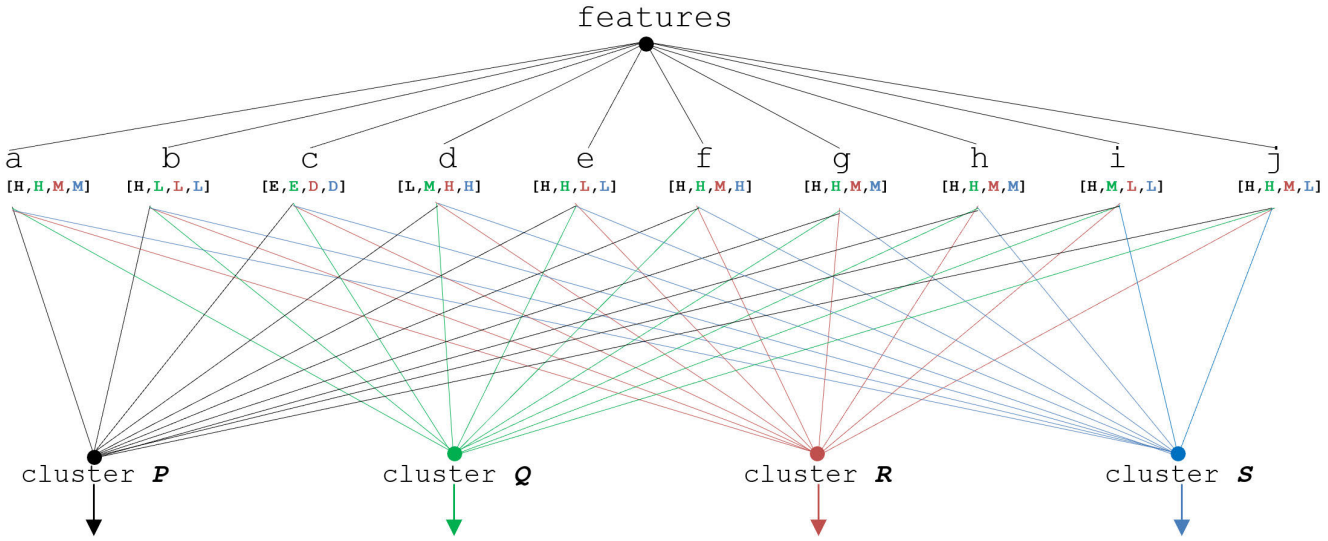

FIGURE 9. A summary graph of the main features.

higher than students in clusters *Q*, *R*, and *S* (Table 8). In general, problems *C* and *D* are comparatively more difficult and contain tough constraints than problems *A* and *B*. Students in cluster *P* submitted the highest percentage of solutions, about 43.36%, for problems *C* and *D* compared to clusters *Q*, *R*, and *S* (Table 7). Moreover, each student in cluster *P* solved an average of 56.14 additional problems, which is significantly higher than students in clusters *Q*, *R*, and *S* (Table 10).

Students in cluster *P* attempted many additional and challenging problems, resulting in a high percentage of errors in *TLE* and *MLE*. Usually, complex algorithm-based problems contain various tough constraints, and sometimes it is very difficult to deal with these kinds of constraints alone without prerequisite knowledge. Our analysis shows that students in cluster *P* have a high tendency to take on difficult problems independently (Table 7), and have achieved significant success in solving problems with tough constraints (Table 6). Besides, students in this cluster still have the opportunity to further improve their programming skills in dealing tough constraint-based problems. Based on the overall empirical and analytical results, we can summarize that the students of cluster *P* are highly skilled and enthusiastic about programming and perform well on academic tests.

Similarly, students in cluster *Q* achieved higher values in most features, as shown in Figure 9. More than 40% of their assignments were submitted on the very first day (Figure 6), with higher *accuracy*, *AC* rate, repetition tendency, and scores in *PA* and *PT*. The error verdicts of this cluster have been analyzed approximately 45% of the errors occur due to *WA*, 19% due to *CE*, 11% due to *PrE*, 11% due to resource limitation (*TLE*, *MLE*, *OLE*), and 14% due to *RTE* (Figure 4). These students can be understood by analyzing the reasons for each type of error. In addition to these positive features, we found some flaws. The students in cluster *Q* achieved medium (*M*) scores in *CoT* and did not solve a significant number of problems outside of their regular assignments. Usually, *CoT* is used to verify actual programming ability; a medium score in *CoT* means students need to pay more attention in programming. Accordingly, to improve programming skills, students can practice more outside of their academic workload.

Considering all the results and analysis, we determined the following recommendations for clusters *P* and *Q*: (*i*) special attention to these students can further improve their skills and knowledge; (*ii*) more difficult problems can be assigned to these students because they find general assignments are very easy; and (*iii*) they can be involved in real-world problem-solving tasks.

For the students of cluster *R*, the rate of last-day submission was approximately 20.06% (Figure 6) which indicates a tendency to delay submission, and they show inconsistent acceptance (*AC*) for all topics (Figure 7). Moreover, this cluster had the lowest acceptance (*AC*) and *accuracy* rates among all clusters; very few students repeated their accepted solutions for optimization and solved extra problems. Students of this cluster obtained the highest error rate (67.30%) among all the clusters. The extracted association rules show that most of these students achieve lower *CoT* scores, *accuracy*, *AC* rate, and *WA* verdicts. The students in this cluster scored much higher in *PA* (60.92) than *CoT* (28.46). During *PA*, students can consult with others to solve problems. This may allow some students to solve problems with the help of other students without understanding the problems properly. In contrast, students are not allowed to consult/talk with others during *CoT*, in which students of this cluster rarely obtain good scores. The average *CoT* score is 28.46 out of 120. Considering all the features, it is concluded that (*i*) students may solve assignments with the help of

others without understanding the problems and this cluster (*ii*) lacks actual programming skills, and (*iii*) has less effort in programming.

Figure 9 shows that students of cluster *S* achieved lower values in most features. Their last-day submission rate is 22.22%, which is the highest among all clusters. They achieved lower scores in coding and paper-based examinations (*CoT* and *PT*), but obtained relatively high scores in *PA*. Similarly, they had fewer *T&E* attempts but achieved higher *accuracy* and *AC* rate. Most of the association rules are involved with lower scores (*CoT* and *PT*) and *accuracy*, as well. The following observations can be obtained from the extracted features: (*i*) while a large number of solutions were submitted on the last day, there may have been some students who waited for other solutions to become available; this is justified by the *CVal* score. (*ii*) There is an unusual trend where students obtained lower scores in *CoT* while achieving higher *AC* rate, *accuracy*, and *PA* scores; this suggests (*iii*) a lack of actual programming skills and (*iv*) less effort in programming, and that (*v*) the students may solve their assignments through collaboration with others.

After analyzing the features and association rules from different perspectives, some deficiencies have been identified in the programming and academic fields for the students of clusters *R* and *S*. Accordingly, we provide some recommendations that may help improve students' programming skills and academic performance: (*i*) special assistance can be provided in the development of algorithms and mathematical logic; (*ii*) encourage students to solve problems with self-knowledge and understanding; (*iii*) students can participate in different programming activities, such as competitions, programming lectures, and workshops; and (*iv*) teachers should give these students additional attention and support in theory and exercise classes and observe their responses.

B. OVERALL ASSESSMENTS AND PRACTICAL APPLICATIONS

Considering all the empirical results and analysis, we see that the students of cluster *P* obtained the highest acceptance rate of 40.88% for all problems (*A*, *B*, *C*, and *D*) (Table 6), average solution accuracy of 55.71% (Table 8), solved the average additional problem of 56.14 (Table 10), a faster propensity to submit assignments early (Figure 6), topic-wise accepted solution rate of 46.24 for problems (*A* and *B*) (Table 11), lowest submission rate on the last day of 2.90% (Table 12), highest number of repetitions (Table 13), highest *PA*, *CoT*, and *PT* scores of 81.22%, 65.46%, and 82.33%, respectively (Table 9). All the features indicate that the students of cluster *P* invested great efforts in programming-related tasks. In addition, the summary graph (Figure 9) of the features shows that the students of cluster *P* are involved in better indicators in all the features. Similarly, students in cluster *Q* received an acceptance rate of 36.34% for all problems (*A*, *B*, *C*, and *D*) (Table 6), solution accuracy of 48.45% (Table 8), high tendency to submit assignments early (Figure 6), the topic-wise acceptance rate of 40.06% for problems (*A* and *B*) (Table 11), last-day submission rate of 10.28% (Table 12), and *PA*, *CoT*, and *PA* scores of 67.86%, 40.18%, and 70.43% respectively (Table 9). As shown in Figure 9, most of the features are associated with good indicators. It can be seen that the students of cluster *Q* also performed well in programming.

On the other hand, students in cluster *S* did not solve any additional problems (Table 10), had a less repetition tendency (Table 13), a higher last day submission rate of 22.22% compared to clusters *P*, *Q*, and *R* (Table 12 and Figure 6), and received the lowest *CVal* score of 0.44 compared to clusters *P*, *Q*, and *R*. Besides, students scored 54.33%, 13.79%, and 44.83% on *PA*, *CoT*, and *PT*, respectively, which is very poor compared to clusters *P*, *Q*, and *R* (Table 9). The overall results show that the students in this cluster did not perform well in programming. In addition, the summary graph (Figure 9) of the features and association rules (Table 18) show that they were involved with lower indicators in most features.

From the above results, it can be seen that the students of clusters *P* and *Q* made a good effort in programming and obtained good results in various tests, while the students of cluster *S* made less effort and therefore achieved poor results in various tests. So, we can conclude that if students (especially in ICT-related disciplines) perform well in practical applications (e.g., programming, logical implementation) then they are also likely to perform well in different academic activities, including tests. In addition, the current research provided some recommendations for students based on the identified features and flaws. Teachers, instructors, and faculty advisors can use these analytical results and recommendations to improve students' programming and academic performance levels. In addition, our proposed framework, experiments, and overall analytical results can be applied to other related courses/disciplines.

The ultimate goal of this research is to support and improve student learning by identifying their weaknesses and strengths. For this purpose, a real-world dataset from a programming course was used. The proposed framework included EDM techniques and LA to find invisible knowledge from the e-learning data. The knowledge was then analyzed and visualized from various perspectives. The results of these analyses highlight the weaknesses and strengths of the students and improve their learning. The proposed research can be suitable for practical applications for the following reasons: (*i*) the proposed research can provide a useful direction, that is, how to deal with e-learning data, (*ii*) e-learning data processing has always been a challenging task, in this regard, the proposed research shows the way of handling real-world e-learning data. As the proposed research has already processed OJ (e-learning) data for EDM and LA, (*iii*) the process of data analysis and its results can be helpful for other related courses to improve students' learning, and (*iv*) the proposed framework can be integrated with existing e-learning platforms for EDM and LA purposes.

C. LIMITATIONS

The proposed framework is leveraged for data clustering, and then the hidden features and association rules are extracted from each cluster. The results are generated based on a dataset comprising submission logs and scores collected from the AOJ system; they may vary for other datasets due to noise or irrelevant data. The number of association rules may vary depending on the threshold values of *minSup* and *minConf*. The value of k for the modified k-means clustering algorithm may differ based on the dataset. Therefore, the proposed framework can produce better or worse results for other datasets.

VII. CONCLUSION AND FUTURE WORK

In this research, a novel framework for exploring the effects of practical skills on academic performance was proposed. Subsequently, a programming course was selected as a sample course for experiments and analyses. By employing the framework, many meaningful and significant features were extracted from the dataset. The extracted features are deeply correlated to the students' behavior. The analytical results showed that better practical (e.g., programming) skills have a positive effect on academic performance. Moreover, the interaction and interdependence between practical skills and academic performance are presented based on the experimental results. Thus, we have concluded that if a student of an ICT or engineering discipline performs well in practical assignments (e.g., programming, logical implementation, PL/SQL, etc.), then they are likely to perform well in other academic activities. The overall approach of this research is applicable to other fields such as education, educational data mining, data analytics, and behavior analysis. In future work, we will consider an automated recommender system that can guide students to improve their practical skills. Moreover, other types of datasets in addition to programming logs and scores will be included.

AVAILABILITY OF DATA AND MATERIALS

In the present research, all the experimental data are collected from AOJ system and class performance scores of a course (ALDS1). Source code submission logs are accessed by these two (02) web applications http://developers.u-aizu.ac.jp/index and https://onlinejudge.u-aizu.ac.jp.

CONFLICT OF INTEREST

The authors declare that they have no conflicts of interest.

ETHICS APPROVAL

This research was approved by the Research Ethics Examination Boards, The University of Aizu, Japan.

REFERENCES

[1] A. Vee, "Understanding computer programming as a literacy," *Literacy Composition Stud.*, vol. 1, no. 2, pp. 42–64, Nov. 2013, doi: 10.21623/1.1.2.4.

[2] L. E. Margulieux, B. B. Morrison, and A. Decker, "Reducing withdrawal and failure rates in introductory programming with subgoal labeled worked examples," *Int. J. STEM Educ.*, vol. 7, no. 1, pp. 1–16, May 2020, doi: 10.1186/s40594-020-00222-7.

[3] S. Wasik, M. Antczak, J. Badura, A. Laskowski, and T. Sternal, "A survey on online judge systems and their applications," *ACM Comput. Surv.*, vol. 51, no. 1, pp. 1–34, Apr. 2018, doi: 10.1145/3143560.

[4] R. Yera and L. Martínez, "A recommendation approach for programming online judges supported by data preprocessing techniques," *Appl. Intell.*, vol. 47, pp. 277–290, Mar. 2017, doi: 10.1007/s10489-016-0892-x.

[5] S. Manzoor, "Common mistakes in online and real-time contests," *ACM Mag. Students*, vol. 14, no. 4, pp. 10–16, Jun. 2008, doi: 10.1145/1375972.1375976.

[6] M. A. Revilla, S. Manzoor, and R. Liu, "Competitive learning in informatics: The UVA online judge experience," *Olympiads Informat.*, vol. 2, no. 10, pp. 131–148, 2008.

[7] F. Okubo, T. Yamashita, and A. Shimada, "Students' performance prediction using data of multiple courses by recurrent neural network," in *Proc. 25th Int. Conf. Comput. Educ. (ICCE)*, Christchurch, New Zealand, Dec. 2017, pp. 439–444.

[8] J. Petit, S. Roura, J. Carmona, J. Cortadella, J. Duch, O. Gimnez, A. Mani, J. Mas, E. Rodrguez-Carbonell, E. Rubio, and E. de San Pedro, "Jutge.org: Characteristics and experiences," *IEEE Trans. Learn. Technol.*, vol. 11, no. 3, pp. 321–333, Jul./Sep. 2018, doi: 10.1109/TLT.2017.2723389.

[9] J. L. Bez, N. A. Tonin, and P. R. Rodegheri, "URI online judge academic: A tool for algorithms and programming classes," in *Proc. 9th Int. Conf. Comput. Sci. Educ.*, Vancouver, BC, Canada, Aug. 2014, pp. 149–152.

[10] R. Romli, S. Sulaiman, and K. Z. Zamli, "Improving automated programming assessments: User experience evaluation using fast-generator," *Proc. Comput. Sci.*, vol. 72, pp. 186–193, Jan. 2015, doi: 10.1016/j.procs.2015.12.120.

[11] C. A. Higgins, G. Gray, P. Symeonidis, and A. Tsintsifas, "Automated assessment and experiences of teaching programming," *J. Educ. Resour. Comput.*, vol. 5, no. 3, pp. 1–21, Sep. 2005, doi: 10.1145/1163405.1163410.

[12] I. Mekterovic, L. Brkic, B. Milasinovic, and M. Baranovic, "Building a comprehensive automated programming assessment system," *IEEE Access*, vol. 8, pp. 81154–81172, 2020, doi: 10.1109/ACCESS.2020.2990980.

[13] A. Kosowski, M. Malafiejski, and T. Noinski, "Application of an online judge & contester system in academic tuition," in *Proc. 6th Int. Conf. Adv. Web Based Learn. (ICWL)*, Edinburgh, U.K., Aug. 2007, pp. 343–354.

[14] N. A. Rashid, L. W. Lim, O. S. Eng, T. H. Ping, Z. Zainol, and O. Majid, "A framework of an automatic assessment system for learning programming," in *Advanced Computer and Communication Engineering Technology* (Lecture Notes in Electrical Engineering), vol. 362. Cham, Switzerland: Springer, Dec. 2016, pp. 967–977, doi: 10.1007/978-3-319-24584-3_82.

[15] M. M. Rahman, Y. Watanobe, and K. Nakamura, "Source code assessment and classification based on estimated error probability using attentive LSTM language model and its application in programming education," *Appl. Sci.*, vol. 10, no. 8, p. 2973, Apr. 2020, doi: 10.3390/app10082973.

[16] M. M. Rahman, Y. Watanobe, and K. Nakamura, "A neural network based intelligent support model for program code completion," *Sci. Program.*, vol. 2020, pp. 1–18, Jul. 2020, doi: 10.1155/2020/7426461.

[17] V. Hegde and H. S. S. Rao, "A framework to analyze performance of Student's in programming language using educational data mining," in *Proc. IEEE Int. Conf. Comput. Intell. Comput. Res. (ICCIC)*, Dec. 2017, pp. 1–4, doi: 10.1109/ICCIC.2017.8524244.

[18] K. L.-M. Ang, F. L. Ge, and K. P. Seng, "Big educational data & analytics: Survey, architecture and challenges," *IEEE Access*, vol. 8, pp. 116392–116414, 2020, doi: 10.1109/ACCESS.2020.2994561.

[19] J. Knobbout and E. Van Der Stappen, "Where is the learning in learning analytics? A systematic literature review on the operationalization of learning-related constructs in the evaluation of learning analytics interventions," *IEEE Trans. Learn. Technol.*, vol. 13, no. 3, pp. 631–645, Jul. 2020, doi: 10.1109/TLT.2020.2999970.

[20] Y. Maher, S. M. Moussa, and M. E. Khalifa, "Learners on focus: Visualizing analytics through an integrated model for learning analytics in adaptive gamified e-learning," *IEEE Access*, vol. 8, pp. 197597–197616, 2020, doi: 10.1109/ACCESS.2020.3034284.

[21] J. L. F. Aleman, "Automated assessment in a programming tools course," *IEEE Trans. Educ.*, vol. 54, no. 4, pp. 576–581, Nov. 2011, doi: 10.1109/TE.2010.2098442.

[22] R. E. Francisco and A. P. Ambrosio, "Mining an online judge system to support introductory computer programming teaching," in *Proc. 8th Int. Conf. Educ. Data Mining*, Madrid, Spain, Jun. 2015, pp. 1–6.

[23] F. Restrepo-Calle, J. J. R. Echeverry, and F. A. González, "Continuous assessment in a computer programming course supported by a software tool," *Comput. Appl. Eng. Educ.*, vol. 27, no. 1, pp. 80–89, Sep. 2018, doi: 10.1002/cae.22058.
[24] X. Lu, D. Zheng, and L. Liu, "Data driven analysis on the effect of online judge system," in *Proc. IEEE Int. Conf. Internet Things (iThings) IEEE Green Comput. Commun. (GreenCom) IEEE Cyber, Phys. Social Comput. (CPSCom) IEEE Smart Data (SmartData)*, Exeter, U.K., Jun. 2017, pp. 573–577.
[25] R. Y. Toledo, Y. C. Mota, and L. Martínez, "A recommender system for programming online judges using fuzzy information modeling," *Information*, vol. 5, no. 2, pp. 1–17, Apr. 2018, doi: 10.3390/informatics5020017.
[26] C. M. Intisar, Y. Watanobe, M. Poudel, and S. Bhalla, "Classification of programming problems based on topic modeling," in *Proc. 7th Int. Conf. Inf. Educ. Technol.*, Aizu-Wakamatsu, Japan, Mar. 2019, pp. 275–283.
[27] A. R. Anaya and J. Boticario, "A data mining approach to reveal representative collaboration indicators in open collaboration frameworks," in *Proc. 2nd Int. Conf. Educ. Data Mining (EDM)*, Córdoba, Spain, Jul. 2009, pp. 210–219.
[28] S. Kausar, X. Huahu, I. Hussain, W. Zhu, and M. Zahid, "Integration of data mining clustering approach in the personalized e-learning system," *IEEE Access*, vol. 6, pp. 72724–72734, 2018, doi: 10.1109/ACCESS.2018.2882240.
[29] E. S. Tabanao, M. M. T. Rodrigo, and M. C. Jadud, "Predicting at-risk novice Java programmers through the analysis of online protocols," in *Proc. 7th Int. Workshop Comput. Educ. Res.*, New York, NY, USA, Aug. 2011, pp. 58–92.
[30] M. M. Rahman, Y. Watanobe, and K. Nakamura, "An efficient approach for selecting initial centroid and outlier detection of data clustering," in *Proc. 18th Int. Conf. Intell. Softw. Methodol., Tools, Techn. (SOMET)*, Kuching, Malaysia, Sep. 2019, pp. 616–628.
[31] R. Agrawal and R. Srikant, "Fast algorithms for mining association rules in large databases," in *Proc. 20th Int. Conf. Very Large Data Bases*, San Francisco, CA, USA, Sep. 1994, pp. 487–499.
[32] H. Toivonen, "Sampling large databases for association rules," in *Proc. 22nd Int. Conf. Very Large Data Bases*, San Francisco, CA, USA, Sep. 1996, pp. 134–145.
[33] J. S. Park, M.-S. Chen, and P. S. Yu, "An effective hash-based algorithm for mining association rules," *ACM SIGMOD Rec.*, vol. 24, no. 2, pp. 175–186, May 1995, doi: 10.1145/568271.223813.
[34] S. Brin, R. Motwani, J. D. Ullman, and S. Tsur, "Dynamic itemset counting and implication rules for market basket data," *ACM SIGMOD Rec.*, vol. 26, no. 2, pp. 255–264, Jun. 1997, doi: 10.1145/253262.253325.
[35] A. Savasere, E. Omiecinski, and S. B. Navathe, "An efficient algorithm for mining association rules in large databases," in *Proc. 21th Int. Conf. Very Large Data Bases*, San Francisco, CA, USA, Sep. 1995, pp. 432–444.
[36] D. W. Cheung, J. Han, V. T. Ng, and C. Y. Wong, "Maintenance of discovered association rules in large databases: An incremental updating technique," in *Proc. 12th Int. Conf. Data Eng.*, New Orleans, LA, USA, 1996, pp. 106–114, doi: 10.1109/ICDE.1996.492094.
[37] J. Han, J. Pei, and Y. Yin, "Mining frequent patterns without candidate generation," in *Proc. ACM SIGMOD Int. Conf. Manage. Data (SIGMOD)*, New York, NY, USA, May 2000, pp. 1–12.
[38] D. Ai, H. Pan, X. Li, Y. Gao, and D. He, "Association rule mining algorithms on high-dimensional datasets," *Artif. Life Robot.*, vol. 23, no. 3, pp. 420–427, May 2018, doi: 10.1007/s10015-018-0437-y.
[39] J. Pei, J. Han, H. Lu, S. Nishio, S. Tang, and D. Yang, "H-Mine: Fast and space-preserving frequent pattern mining in large databases," *IIE Trans.*, vol. 39, no. 6, pp. 593–605, Mar. 2007, doi: 10.1080/07408170600897460.
[40] R. C. Agarwal, C. C. Aggarwal, and V. V. V. Prasad, "A tree projection algorithm for generation of frequent item sets," *J. Parallel Distrib. Comput.*, vol. 61, no. 3, pp. 350–371, Mar. 2001, doi: 10.1006/jpdc.2000.1693.
[41] J. Liu, Y. Pan, K. Wang, and J. Han, "Mining frequent item sets by opportunistic projection," in *Proc. 8th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Edmonton, AB, Canada, Jul. 2002, pp. 229–238.
[42] G. Grahne and J. Zhu, "Fast algorithms for frequent itemset mining using FP-trees," *IEEE Trans. Knowl. Data Eng.*, vol. 17, no. 10, pp. 1347–1362, Oct. 2005, doi: 10.1109/TKDE.2005.166.
[43] M. J. Zaki, "Scalable algorithms for association mining," *IEEE Trans. Knowl. Data Eng.*, vol. 12, no. 3, pp. 372–390, May/Jun. 2000, doi: 10.1109/69.846291.
[44] A. Lile, "Analyzing e-learning systems using educational data mining techniques," *Medit. J. Social Sci.*, vol. 2, no. 3, pp. 403–419, Sep. 2011, doi: 10.5901/mjss.2011.v2n3p403.
[45] E. Fernandes, M. Holanda, M. Victorino, V. Borges, R. Carvalho, and G. V. Erven, "Educational data mining: Predictive analysis of academic performance of public school students in the capital of Brazil," *J. Bus. Res.*, vol. 94, pp. 335–343, Jan. 2019, doi: 10.1016/j.jbusres.2018.02.012.
[46] I. E. Livieris, K. Drakopoulou, V. T. Tampakas, T. A. Mikropoulos, and P. Pintelas, "Predicting secondary school students' performance utilizing a semi-supervised learning approach," *J. Educ. Comput. Res.*, vol. 57, no. 2, pp. 448–470, Jan. 2018, doi: 10.1177/0735633117752614.
[47] S. A. Salloum, M. Alshurideh, A. Elnagar, and K. Shaalan, "Mining in educational data: Review and future directions," in *Proceedings of the International Conference on Artificial Intelligence and Computer Vision (AICV2020)* (Advances in Intelligent Systems and Computing), vol. 1153, A. E. Hassanien, A. Azar, T. Gaber, D. Oliva, and F. Tolba, Eds. Cham, Switzerland: Springer, 2020, doi: 10.1007/978-3-030-44289-7_9.
[48] T. P. Tran and D. Meacheam, "Enhancing learners' experience through extending learning systems," *IEEE Trans. Learn. Technol.*, vol. 13, no. 3, pp. 540–551, Jul./Sep. 2020, doi: 10.1109/TLT.2020.2989333.
[49] O. Viberg, M. Hatakka, O. Bälter, and A. Mavroudi, "The current landscape of learning analytics in higher education," *Comput. Hum. Behav.*, vol. 89, pp. 98–110, Dec. 2018, doi: 10.1016/j.chb.2018.07.027.
[50] L.-K. Lee, S. K. S. Cheung, and L.-F. Kwok, "Learning analytics: Current trends and innovative practices," *J. Comput. Educ.*, vol. 7, no. 1, pp. 1–6, Feb. 2020, doi: 10.1007/s40692-020-00155-8.
[51] F. Dunke and S. Nickel, "A data-driven methodology for the automated configuration of online algorithms," *Decis. Support Syst.*, vol. 137, Oct. 2020, Art. no. 113343, doi: 10.1016/j.dss.2020.113343.
[52] Y. Watanobe. *Aizu Online Judge*. Accessed: May 19, 2020. [Online]. Available: https://onlinejudge.u-aizu.ac.jp
[53] *Aizu Online Judge: Developers Site (API)*. Accessed: Dec. 19, 2019. [Online]. Available: http://developers.u-aizu.ac.jp/index
[54] T. Saito and Y. Watanobe, "Learning path recommendation system for programming education based on neural networks," *Int. J. Distance Educ. Technol.*, vol. 18, no. 1, pp. 36–64, Jan. 2020, doi: 10.4018/IJDET.2020010103.
[55] Y. Watanobe, C. M. Intisar, R. Cortez, and A. Vazhenin, "Next-generation programming learning platform: Architecture and challenges," in *Proc. 2nd ACM Chapter Conf. Educ. Technol., Lang. Tech. Commun.*, Aizu-Wakamatsu, Japan, Nov. 2020, pp. 1–11.
[56] M. M. Rahman, Y. Watanobe, and K. Nakamura, "A bidirectional LSTM language model for code evaluation and repair," *Symmetry*, vol. 13, no. 2, p. 247, Feb. 2021, doi: 10.3390/sym13020247.
[57] International Business Machines (IBM). (May 2021). *Project CodeNet*. [Online]. Available: https://github.com/IBM/Project_CodeNet
[58] P. Wang, C. An, and L. Wang, "An improved algorithm for mining association rule in relational database," in *Proc. Int. Conf. Mach. Learn. Cybern.*, Lanzhou, China, Jul. 2014, pp. 247–252.

MD. MOSTAFIZER RAHMAN received the B.Sc. degree in engineering from the Department of Computer Science and Engineering, Hajee Mohammad Danesh Science and Technology University, Dinajpur, Bangladesh, in 2009, and the M. Sc. degree in engineering from the Department of Computer Science and Engineering, Dhaka University of Engineering & Technology, Gazipur, Bangladesh, in 2014. He is currently pursuing the Ph.D. degree with the Database Systems Laboratory, Department of Computer and Information Systems, The University of Aizu, Aizuwakamatsu, Fukushima, Japan. He is also working (on study leave) at the Dhaka University of Engineering & Technology. His research interests include machine learning, deep learning, machine learning application in programming, programming education, data mining, and big data analytics.

YUTAKA WATANOBE (Member, IEEE) received the master's and Ph.D. degrees from The University of Aizu, Japan, in 2004 and 2007, respectively. He was a Research Fellow of the Japan Society for the Promotion of Science (JSPS), The University of Aizu, in 2007. He was a Coach of several ACM-ICPC World Final teams. He is currently a Senior Associate Professor with the School of Computer Science and Engineering, The University of Aizu. He is a Key Member of the Aizu Online Judge (AOJ) System. His research interests include visual programming language, programming education, data mining, e-learning systems, filmification of methods, and cloud robotics.

RAGE UDAY KIRAN received the Ph.D. degree in computer science from the International Institute of Information Technology, Hyderabad, Telangana, India. He was a Project Assistant Professor with the Kitsuregawa Laboratory, Institute of Industrial Science, The University of Tokyo, Tokyo, Japan. He was a Researcher with the Social Big Data Research Collaboration Center, National Institute of Information and Communications Technology, Tokyo. He is currently an Associate Professor with the School of Computer Science and Engineering, The University of Aizu, Japan. His current research interests include data mining, parallel computation, air pollution data analytics, traffic congestion data analytics, recommender systems, and ICTs for agriculture.

TRUONG CONG THANG (Senior Member, IEEE) received the B.E. degree from the Hanoi University of Science and Technology, Vietnam, in 1997, and the Ph.D. degree from KAIST, South Korea, in 2006. From 1997 to 2000, he was a Network Engineer with the Vietnam Post and Telecommunications (VNPT). Since 2002, he has been an Active Member of Korean and Japanese delegations to standard meetings of ISO/IEC and ITU-T. From 2007 to 2011, he was a member of Research Staff with the Electronics and Telecommunications Research Institute (ETRI), South Korea. Since 2011, he has also been an Associate Professor with The University of Aizu, Japan. His research interests include multimedia networking, image/video processing, content adaptation, IPTV, and MPEG/ITU standards.

INCHEON PAIK (Senior Member, IEEE) received the master's and Ph.D. degrees in electronics engineering from Korea University, in 1987 and 1992, respectively. He is currently a Full Professor with The University of Aizu, Japan. His research interests include semantic web, web services and their composition, web data mining, big data analytics, deep learning, awareness computing, and agents on semantic web. He served in several conferences as the chair. He also serves as an Editor for the journals of *JIPS* and *IEICE*.