Educational Data Mining to Support Programming Learning Using Problem-Solving Data

MD. MOSTAFIZER RAHMAN1, 2, YUTAKA WATANOBE (Member, IEEE) 1, TAKU MATSUMOTO1, RAGE UDAY KIRAN1, KEITA NAKAMURA 1

1Graduate Department of Computer Science and Engineering, The University of Aizu, Aizu-Wakamatsu City, Fukushima, Japan
2Department of Computer Science and Engineering, Dhaka University of Engineering & Technology, Gazipur, Bangladesh

ABSTRACT Computer programming has attracted a lot of attention in the development of information and communication technologies in the real world. Meeting the growing demand for highly skilled programmers in the ICT industry is one of the major challenges. In this point, online judge (OJ) systems enhance programming learning and practice opportunities in addition to classroom-based learning. Consequently, OJ systems have created a large number of problem-solving data (solution codes, logs, and scores) archives that can be valuable raw materials for programming education research. In this paper, we propose an educational data mining framework to support programming learning using unsupervised algorithms. The framework includes the following sequence of steps: (i) problem-solving data collection (logs and scores are collected from the OJ) and preprocessing; (ii) MK-means clustering algorithm is used for data clustering in Euclidean space; (iii) statistical features are extracted from each cluster; (iv) frequent pattern (FP)-growth algorithm is applied to each cluster to mine data patterns and association rules; (v) a set of suggestions are provided on the basis of the extracted features, data patterns, and rules. Different parameters are adjusted to achieve the best results for clustering and association rule mining algorithms. For the experiment, approximately 70,000 real-world problem-solving data from 537 students of a programming course (Algorithm and Data Structures) were used. In addition, synthetic data have leveraged for experiments to demonstrate the performance of MK-means algorithm. The experimental results show that the proposed framework effectively extracts useful features, patterns, and rules from problem-solving data. Moreover, these extracted features, patterns, and rules highlight the weaknesses and the scope of possible improvements in programming learning.

INDEX TERMS Educational Data Mining, Programming Learning, Problem-solving Data, Clustering, Pattern Mining, Rule Mining

I. INTRODUCTION

TODAY’S information and communication technology (ICT) industry demands for highly skilled programmers for further development. The conventional computer programming learning environment is insufficient to prepare highly skilled programmers due to the limited number of exercise classes, limited practice opportunities, and lack of individual tutoring. In addition, most educational institutions, such as schools, colleges, and universities are struggling to build more educational facilities to increase academic activity (e.g., additional exercise classes, practice, and individual tutoring) due to logistical and organizational constraints [1]. The growing number of people in classrooms in educational institutions, the large number of students per class, and some lectures are conducted with more than a thousand participants in the massive open online courses which complicate the individual tutoring process [2]. Furthermore, the growing ratio between students and educators raises the question of how to provide individual support to students to improve their problem-solving skills. Especially, when learning computer programming, students need a lot of practice and individual tutoring to improve their programming knowledge and skills. Computer programming is one of the fundamental courses in ICT discipline. Programming practice and competition can...
play an important role in acquiring good programming skills. More programming practice can help to improve a student’s problem-solving skills. Due to logistical and institutional constraints, the traditional classroom-based programming learning approach is a major obstacle to the development of students’ programming skills.

Research in technology-assisted learning (TAL) has focused on addressing these challenges by exploiting the potential of information and technology (IT). Over the past 40 years, educational researchers have examined the effects of IT on learning outcomes [3]. In the last 5 to 15 years, researchers have been able to demonstrate the importance of TAL systems which are more effective than non-personal tutoring conditions [4]–[6]. Many TAL systems assist students at every step of the problem-solving session rather than just evaluating the final answer. These systems follow a variety of scaffolding strategies to support students’ learning processes [7]. Scaffolding strategies transform learning activities into smaller modules, maximizing the use of tools and structures to support students to gain more knowledge [8]. Scaffolding strategies can be of two types, namely, dynamic and static [9]. In dynamic scaffolding strategies, TAL systems continuously analyze student activity and provide the necessary support based on students’ problems and responses. In contrast, static scaffolding strategies provide static support to students based on the analysis of students’ previous difficulties and responses.

Recently, a new technology known as smart personal assistants (SPAs) has become available to end-users. Some examples of SPAs are Siri (Apple), Alexa (Amazon), and Assistant (Google). These technologies are highly intelligent and interactive, which can be useful for learning [10]. In addition, SPA technology will be powered by more than 870 million devices by 2022 [11]. An SPA is an application of artificial intelligence (AI) that provides assistance (e.g., answering questions, recommendations, executing actions, suggestions, etc.) based on user input (e.g., voice, images, and other types of information) [12]. These systems are hosted by big technology giants to receive voice or text data and produce the relevant output [13]. Despite the many advantages of SPA systems for quick answers, they are difficult to employ for the purpose of programming learning and evaluation.

In contrast, online judge (OJ) systems introduce an alternative programming learning platform, where students/learners can perform their programming learning activities over the year [14]. Note that, an OJ system is a type of TAL system used as a learning tool (e.g., programming, logical implementations, other types of exercises, etc.). Many educational bodies have developed their own OJ or programming assessment systems to provide students with better opportunities to learn programming. A few examples of OJ systems are AOJ [15], UVa [16], URI [17], and Jutge.org [18], which are being used as academic tools for different programming and exercise courses in universities. OJ systems are not only effective for programming courses but also have wide adaptations across different domains [19]. Because of these OJ systems, a large number of problem-solving data (solution codes and logs) are generated every day that can be valuable resources for programming educational research.

The rapid proliferation of e-learning platforms and their use in educational institutions has resulted in a huge amount of data archives. Educational data is evolving and the diversity of these data varies from one e-learning platform to another. Efficiently handling this massive and diverse educational data is a non-trivial and challenging task. Educational data mining (EDM) technique has emerged to address this problem. Due to the diversity, volume, nature, and structure of educational data, conventional data mining algorithms cannot be applied directly. Also, the data preprocessing is one of the most important tasks in the EDM process to achieve better results [20]. In the context of EDM, the usability and applicability of clustering algorithms are reviewed in research [20]. Munshi et al. [21] proposed an education data analytical platform to discover the hidden knowledge from educational big data. In another study [22], a web-based tool for EDM has been developed to support tutoring in higher education. In [23], the performance of clustering algorithms is automatically compared and the most efficient clustering algorithm is recommended based on the dataset. Statistical data analysis is performed on programming logs and test scores to investigate the hidden information of programmers in the study [24]. Furthermore, EDM can be employed to find useful knowledge and information from problem-solving data to support programming learning.

Particularly, learning analytics (LA) visualizes and analyzes educational data to improve the learning process and outcomes. Moreover, LA has a broader coverage of other areas such as EDM, learning sciences, academic analytics, and so on. Mangaroska et al. [25] presented an overview of LA and its application to learning technologies. In [26], an educational LA technology called Web Programming Grading Assistant (WPGA) is used to study the effectiveness of students’ programming learning.

In addition, rule-based recommender systems (RSs) extract useful information spread in large amounts of data. To find the most similar activities from a large amount of data, a rule-based RS is the effective one. Furthermore, a meta-rule based RS provides personalized recommendations to learners [27]. In e-learning environments, rule-based RSs are widely used to provide information to students and educators [28], [29]. Since conventional e-learning systems capture only grades against answer submissions, they are inadequate to evaluate practical-based answers (e.g., programming solutions, circuit design, and evaluation of mathematical equations). Many evaluation metrics are involved when evaluating practical-based answers. The OJ system can assess practical-based answer (e.g., solution codes) and generate a variety of evaluation metrics such as accepted, wrong answer, compile error, run time exceeded, memory limit exceeded, CPU time, and memory usage. So, rule-based RSs can be useful for OJ systems to recommend suitable problems to the users.
Although, extracting hidden information and discovering data patterns and rules from the problem-solving data is a challenging task. In this paper, we have exploited unsupervised algorithms to extract useful information and find data patterns from real-world problem-solving data collected from an OJ system. Upon completion of data preprocessing task, a clustering algorithm is applied to the processed data to cluster it and then statistical features are extracted from each cluster. Next, the frequent pattern (FP)-growth algorithm is applied to explore data patterns and association rules from each cluster. Finally, recommendations are provided for students and educators based on the extracted features, patterns, and rules. In particular, we have focused on finding effective data patterns from problem-solving data that can be useful to support programming learning. This paper makes the following contributions:

- We propose a framework for EDM and data analysis using unsupervised machine learning algorithms. To demonstrate the effectiveness of the proposed framework, experiments are conducted on real-world problem-solving dataset.
- Various features, patterns, and rules are extracted from problem-solving data to support programming learning. Useful recommendations are generated for students and educators on the basis of the extracted features, rules, and data patterns.
- The extracted knowledge (features, rules and patterns) shows the weaknesses and the scope of possible improvements in programming learning.
- The proposed EDM framework can be integrated into e-learning platforms and OJ systems.

The remainder of this paper is organized as follows. Section II covers background and recent related research works. Section III describes data collection and preprocessing. In Section IV, the proposed approach is presented. Section V shows the experimental results. Finally, the discussion and conclusion are given in Sections VI and VII.

II. BACKGROUND AND RELATED LITERATURE

In this section, we present some recent research works that are related to EDM, rule-based RS, clustering techniques, and data pattern and association rule mining (ARM) techniques. In addition, OJ systems and their applications in academia will be presented.

A. EDUCATIONAL DATA MINING

In the last few years, e-learning platforms have become more popular for a variety of reasons and demands, including teacher shortage, unbalanced student-teacher ratio, logistical and infrastructure constraints, high cost of technical and professional courses, dissemination of education to a large number of people, time saving and easy access to many courses [30]. As the use of e-learning systems increases, different types of data are being generated regularly. Some data are structured whereas some are unstructured. Therefore, it is very difficult to retrieve useful information from this huge amount of mixed data archives using traditional statistical algorithms [31]. In [32], an EDM technique such as k-means clustering was used to evaluate students’ activities in an e-learning system and identified students’ interests. It also identified the correlation between the activity in the e-learning system and academic performance. Students’ programming level was evaluated using log data from an OJ system. Xu et al. [33] proposed an analytical model where four factors were considered: Simple (number of easy problems solved correctly), Complex (number of difficult problems solved correctly), Frequency (number of accesses to an OJ system), and Hint (number of times students asked for help). A deep neural network model was trained on the basis of SCFH to classify students into three main groups: “Risky”, “Intermediate”, and “Advanced”.

Moreover, the data preprocessing is one of the important steps for EDM to achieve better results [20]. Hooshary et al. [23] presented the usefulness of clustering techniques in the context of EDM. They proposed an evaluation approach that compares the strength of clustering algorithms on a given dataset and automatically recommends a suitable algorithm for EDM. Unlike existing models, the proposed framework for EDM leverages real-world problem-solving data collected from an OJ system to find data patterns, rules, and features. Finally, some useful recommendations for learners and educators are generated based on the identified features, rules, and data patterns.

B. RULE-BASED RECOMMENDATION SYSTEMS

The volume and variety of content on e-learning platforms are increasing at an unprecedented rate, and at the same time the opportunities for research using the resources of e-learning platforms are also increasing. Recommending relevant and appropriate content to users (e.g., students, instructors, and teachers) is a challenging and tough task for any e-learning platform. Perumal et al. [34] proposed a novel personalized RS to provide appropriate supportive content to users. In their approach, FP-growth algorithm is applied to generate frequent items patterns, and fuzzy logic is used to partition the content into three levels. Recently, some RSs have been using a mixed approach of content-based filtering and collaborative filtering to achieve high-quality results in specific contexts [35]. In addition, most RSs are built with a collaborative, knowledge-based, content-based, and hybrid approaches [36]. Conventional e-learning platforms are insufficient to assess exercise-based content such as programming solution codes automatically, instead they can assess exercise-based contents semi-automatically [37]. Thus, usual RS in e-learning platforms have limited suitability for programming and exercise-based education.

In contrast, OJ systems can automatically evaluate exercise contents such as programming, functional programming, PL/SQL, and circuit design and generate many evaluation parameters (e.g., verdicts, resource usage, errors, submission details, and content details) [18]. In this paper, we use problem-solving data to generate various data patterns,
rules, and features to identify student activities in the OJ system. Also, various useful recommendations are provided for students and educators based on the extracted features, rules and data patterns.

C. OJ SYSTEMS
In recent years, the OJ system has become an effective academic tool for students, teachers, and researchers in terms of programming learning, assessment, and research. Watanobe et al. [38] presented a theory for the development of an OJ system and its internal architecture. An OJ system enables students to conduct programming practices and tests outside the classroom. In addition to assessing programming and functional programming, OJ systems have recently been used to assess complex exercise-based contents such as circuit design and AI [18]. In [39], an automated programming evaluation tool called Edgar was proposed and used to evaluate and analyze a variety of programming tasks. This system does not provide recommendations or any other type of personalized evaluation. In higher education, many universities use OJ systems as an academic tool for programming and other exercises. Examples of OJ systems include Aizu Online Judge (AOJ) [15], [40], UVa (University of Valladolid) online judge [16], Jutge.org (Universitat Politecnica de Catalunya) [18], and URI (Universidade Regional Integrada) [17], which have been in use for a long time. To the best of our knowledge, most of the existing OJ systems use a “learning-by-doing” formula to reinforce students’ programming skills. The huge volume of data stored in these OJ systems helps researchers find shortcomings in students’ programming and identify areas for improvement. As a result, a lot of research is being conducted using these rich resources to identify and solve various programming-related problems [38], [41]–[44].

D. CLUSTERING TECHNIQUES
Clustering techniques are widely used in data analysis and play an important role in the field of data mining. With the diversification of data, many variations of clustering techniques have been developed simultaneously to analyze different types of data. Each clustering technique has its advantages and disadvantages for clustering data. The usability and applicability of clustering techniques in the context of the EDM has been described in a study [20]. To the best of our knowledge, there is no single clustering technique that can handle all types of data including text, numbers, images, and videos. Xu and Tian [45] conducted a comprehensive survey on clustering techniques and discussed their advantages, disadvantages, evaluations, and their complexity. Clustering techniques can be classified into different groups (hierarchy, fuzzy theory, distribution, density, graph theory, grid, fractal theory, and model) based on their working procedures. Here, we present some examples of classical clustering techniques including K-means, SVM, KNN, PAM, CLARA, BIRCH, CURE, FCM, FCS, DBCLASD, GMM, DBSCAN, OPTICS, CLICK, STING, CLIQUE, and COBWEB [45]. There are many improved and extended versions of these classical clustering algorithms. In this paper, a modified K-means (MK-means) clustering algorithm [46] is used for problem-solving data clustering. The algorithm has two important features: (i) selection of the optimal center point, and (ii) detection and removal of noises.

Furthermore, in the K-means clustering algorithm, it is important to select the optimal number of $k$ values because setting an inappropriate $k$ value may change the data characteristics of the clusters. For this purpose, many methods such as Elbow, Dunn Index, Silhouette Coefficient, Gap Statistic, and Canopy are available. In this paper, the Elbow method is used to select the initial number of $k$ clusters in the dataset; the Elbow method is effective in selecting the optimal number of clusters for the partitioning-based clustering techniques [47], [48].

E. PATTERN AND ASSOCIATION RULE MINING TECHNIQUES
ARM has been widely used for data mining purposes. ARM is an unsupervised algorithm and was first introduced in research [49]. There are diverse applications of the ARM technique in various fields such as pattern mining, education, social, medical, census, market-basket, and big data analysis. ARM is an efficient technique for obtaining frequent items from large datasets [50]. Among the many types of ARM algorithms, Apriori and FP-growth algorithms are the most widely used [51], [52]. Comparing Apriori and FP-growth, Apriori requires repeated scanning of the database to form a candidate itemset, whereas the FP-growth algorithm is very fast because it only needs to scan the database twice to complete the process. Many variations and improvements of both algorithms have been introduced in different studies, including hashing technique [53], sampling approach [54], dynamic counting [55], depth-first mining [56], h-mine [57], and tree structures [58].

III. DATA COLLECTION AND PREPROCESSING
In this section, data collection and preprocessing are presented. Problem-solving data (solution evaluation logs and test scores) of the programming course (Algorithm and Data Structures (ALDS)) are collected from the AOJ platform. In addition, the data preprocessing steps for logs and scores as well as creation of transactional database (TDB) are presented.

A. AOJ PLATFORM
The problem-solving data was collected from the AOJ platform [15], [40] for experiments. AOJ is a popular platform for programming practice, contest, and learning around the world. Over the past 15 years, the AOJ system has generated approximately 6.0 million source codes and submission logs. There are approximately 100,000 users registered in the AOJ system, and the collection of problems is about 2,700, which are nicely organized and supported by 18 programming languages. In this paper, problem-solving data of the ALDS programming course at the University of Aizu are used. The
size of the submission log was approximately 70,000, and the scores were collected from approximately 537 students in the ALDS course. The ALDS course consists of 13 topics, and all programming assignments for each topic as well as final coding test are conducted on the AOJ platform. Recently, the source codes of the AOJ system have been used in IBM’s research project “Project CodeNet” [59]. In addition, the submission logs, code archives, and problem sets of the AOJ system have been used for many research and educational purposes [41]–[44].

B. COLLECTION OF SOLUTION EVALUATION LOGS
An OJ system is a web-based service that automatically evaluates solution code; an OJ system evaluates the solution code using a predefined test dataset and returns a judge result with various parameters after testing. This judge result for a solution code is called a solution evaluation log. The judgment parameters for a solution code evaluation include judge identifier (j), users (u), problem name (p), programming language (l), judge’s verdict (v), CPU time (ct), memory usage (mu), code size (cs), solution judgment date (jd), and solution submission date (sd). For better experimentation, a mathematical model represents the judgment parameters in a TDB format.

Let \( U = \{u_1, u_2, \ldots, u_n\}, n \geq 1 \) be the set of users (or students), \( P = \{p_1, p_2, \ldots, p_m\}, m \geq 1 \) be the set of unique problems, and \( J = \{j_1, j_2, \ldots, j_q\}, q \geq 1 \) be the set of judge identifiers. When a user \((u_i)\) submits a solution code to an OJ system, the OJ system generates a verdict, but there are several types of verdicts \((v)\) in the OJ system. Let \( V = \{AC, WA, PE, CE, TLE, RTE, OLE, MLE\} \) be the set of judge verdicts, where \( AC = \) Accepted, \( WA = \) Wrong Answer, \( PE = \) Presentation Error, \( CE = \) Compile Error, \( TLE = \) Time Limit Exceeded, \( RTE = \) Run Time Exceeded, \( OLE = \) Output Limit Exceeded, and \( MLE = \) Memory Limit Exceeded. Let \( L = \{C, C++, python, python2, python3, JAVA, Rust, D, Ruby, Haskell, C\#\} \) be the set of programming languages. Thus, the output of an OJ system can be expressed as \( O_{oj} = \{j_r, u_s, p_t, v_u, ct, l_v, cs, mu, sd, jd\} \), where \( j_r, j, u_s \in U, p_t \in P, v_u \in V, l_v \in L \). Some examples of the solution evaluation logs are produced by the AOJ platform shown in Table 1.

C. COLLECTION OF TEST SCORES
Different types of test scores for the ALDS course were collected from two separate year students (approximately 537) in two separate quarters (Q2, 2018 and Q4, 2019) at the University of Aizu, Japan. There are several tests in this course, including Programming Assignment (ProgA), Algorithm Assignment (AlgoA), Code Validation Score (CVS), Final Coding Test (FCT), and Final Paper-based Test (FPT). Note that, submission of AlgoA is also considered as attendance (ATD). Scores on the FCT and FPT tests effectively determine a student’s programming skills and theoretical knowledge, respectively. Higher scores on both tests (FCT and FPT) indicate better programming skills and theoretical knowledge in the respective course.

Thus, the test (both theory and programming related) scores can be expressed as \( T_{test} = \{U, ATD, AlgoA, ProgA, CVS, FCT, FPT\} \) where \( ATD \in \mathbb{N} \) and \( 0 \leq ATD \leq 120, ProgA \in \mathbb{N} \) and \( 0 \leq ProgA \leq 120, AlgoA \in \mathbb{N} \) and \( 0 \leq AlgoA \leq 100, CVS \in \mathbb{R} \) and \( 0 \leq CVS \leq 1, FPT \in \mathbb{N} \) and \( 0 \leq FPT \leq 120, FCT \in \mathbb{N} \) and \( 0 \leq FCT \leq 120 \). Some examples of \( T_{test} \) are shown in Table 2.

TABLE 1: Examples of solution evaluation logs generated by the AOJ system

| J      | U  | P | V  | L    | ct (1/100 s) | mu (KB) | cs (B) | sd (ms) | jd (ms) |
|--------|----|---|----|------|-------------|---------|-------|--------|--------|
| 926815 | u1 | p1 | AC | C++ | 0          | 1124    | 239   | 1398327386668 | 1398327387170 |
| 936016 | u2 | p3 | WA | C++ | 2          | 1160    | 202   | 1398931090247 | 1398931090608 |
| 1283124| u3 | p7 | CE | C   | 0          | 0       | 294   | 1428630316947 | 1428630317044 |
| 1283153| u4 | p10| PE | C   | 0          | 604     | 322   | 1428630577514 | 1428630577691 |
| 3283150| u5 | p13| WA | Python | 1       | 6588    | 789   | 1544362759147 | 1544362759662 |

D. CREATION OF TDB FOR PATTERN AND RULE MINING
In order to form the TDB for pattern and rule mining, some prominent attributes are selected from both Tables 1 and 2. The selected attributes are \( P, S_{Accu}, V, ProgA, FCT, \) and \( FPT \) and each attribute has a different value. Once data clustering is complete, TDB can help to mine various interesting rules and patterns from each cluster. A new attribute called solution accuracy \((S_{Accu})\) is introduced in TDB, which is defined as follows.

**Definition 1.** The total number of solutions received and accepted by an OJ system from the total number of submissions is called the \( S_{Accu} \).

\[
S_{Accu} = \frac{\sum_{p_n=1}^{p_n} T_{NAS}}{\sum_{p_n=1}^{p_n} T_{NS}}
\]

where \( p_n = \) number of problems, \( T_{NAS} = \) total number of accepted solutions, and \( T_{NS} = \) total number of submissions.

**Example 1.** Let user \( u_1 \) submitted a total of 53 solutions to the OJ system for verdict, and from all submissions user \( u_1 \)
received 43 accepted verdicts. According to Equation 1, the final solution accuracy for user \( u_1 \) is \((43/53) = 81.13\%\).

For better results and explanation, the values of the attributes are divided into separate ranges with unified IDs, as shown in Table 3.

**TABLE 3: Conversion of attribute values to unified IDs**

| ID | Value (%) | Attributes |
|----|-----------|------------|
| 1-57 | \( p_1, p_2, \cdots, p_{57} \) | P |
| 60 | \( \geq 75 \) | SAccu |
| 61 | 60 – 74 | |
| 62 | 45 – 69 | |
| 63 | \( < 45 \) | |
| 70 | AC | V |
| 71 | CE | |
| 72 | MLE | |
| 73 | OLE | |
| 74 | PE | |
| 75 | RLE | |
| 76 | TLE | |
| 77 | WA | |
| 80 | \( \geq 80 \) | ProgA |
| 81 | 65 – 79 | |
| 82 | 45 – 64 | |
| 83 | \( < 45 \) | |
| 90 | \( \geq 80 \) | |
| 91 | 65 – 79 | |
| 92 | 45 – 64 | |
| 93 | \( < 45 \) | |
| 100 | \( \geq 80 \) | FPT |
| 101 | 65 – 79 | |
| 102 | 45 – 64 | |
| 103 | \( < 45 \) | |

Let \( T = \{P, SAccu, V, ProgA, FCT, FPT\} \) be the set of transaction, where \( P = \{ID|ID \in \mathbb{N}, 1 \leq ID \leq 57\} \), \( SAccu = \{ID|ID \in \mathbb{N}, 60 \leq ID \leq 63\} \), \( V = \{ID|ID \in \mathbb{N}, 70 \leq ID \leq 77\} \), \( ProgA = \{ID|ID \in \mathbb{N}, 80 \leq ID \leq 83\} \), \( FCT = \{ID|ID \in \mathbb{N}, 90 \leq ID \leq 93\} \), \( FPT = \{ID|ID \in \mathbb{N}, 100 \leq ID \leq 103\} \). Now, a unified TDB can be generated for each cluster based on the conversion results in Table 3. The TDBs are then used to mine patterns and rules in each cluster; a sample of TDB is shown in Table 4.

**TABLE 4: Sample of a TDB for pattern and rule mining**

| U | P | SAccu | V | ProgA | FCT | FPT |
|---|---|-------|---|-------|-----|-----|
| \( u_1 \) | 1 | 60 | 70 | 80 | 91 | 100 |
| \( u_2 \) | 3 | 60 | 77 | 80 | 90 | 100 |
| \( u_3 \) | 7 | 61 | 71 | 81 | 91 | 100 |
| \( u_4 \) | 10 | 60 | 74 | 80 | 90 | 100 |
| \( u_5 \) | 13 | 60 | 77 | 80 | 90 | 100 |

**IV. PROPOSED APPROACH**

In this section, we describe the proposed EDM approach, which is shown in Figure 1. The proposed approach can be partitioned into three main phases: first, data collection and preparation; second, data clustering, numerical analysis, and feature extraction by applying clustering algorithms to the dataset (Table 2); third, pattern and rule are mined by applying pattern mining algorithms to the TDB (Table 4) of each cluster. In the proposed approach, different algorithms are leveraged in each phase, which are described below.

Let \( T = \{P, SAccu, V, ProgA, FCT, FPT\} \) be the set of transaction, where \( P = \{ID|ID \in \mathbb{N}, 1 \leq ID \leq 57\} \), \( SAccu = \{ID|ID \in \mathbb{N}, 60 \leq ID \leq 63\} \), \( V = \{ID|ID \in \mathbb{N}, 70 \leq ID \leq 77\} \), \( ProgA = \{ID|ID \in \mathbb{N}, 80 \leq ID \leq 83\} \), \( FCT = \{ID|ID \in \mathbb{N}, 90 \leq ID \leq 93\} \), \( FPT = \{ID|ID \in \mathbb{N}, 100 \leq ID \leq 103\} \). Now, a unified TDB can be generated for each cluster based on the conversion results in Table 3. The TDBs are then used to mine patterns and rules in each cluster; a sample of TDB is shown in Table 4.

**FIGURE 1: Overview of the proposed EDM framework**

**A. ELBOW AND MK-MEANS CLUSTERING ALGORITHM**

In our proposed approach, one of the key phases is data clustering. Data analysis, including data patterns, feature extraction, and association rules, is closely related to data clustering, and improper clustering can lead to poor results. Before clustering our multidimensional education data, we analyzed the suitability of different clustering methods and algorithms for our clustering objectives. Each algorithm has its own suitability for a particular application, so it is very difficult to find the best clustering algorithm. However, we looked at the hierarchical clustering method, which has high computational and memory requirements compared to the partitional clustering methods. For large data sets, a hierarchical algorithm can be very expensive. The main differences between hierarchical and partitional methods are computation time, prior assumptions, data sets, and clustering goals/results. We also investigated several partition-based clustering algorithms such as K-means, K-means++, and Fuzzy C-means (FCM). K-means++ is used to select the initial cluster center, FCM is used for soft clustering where a data point belongs to multiple clusters with a certain degree of membership. The applicability of these algorithms is not aligned with our goals of multidimensional data clustering and outlier handling.

For multidimensional data normalization, optimal initial center selection, and outlier handling, we proposed a clustering algorithm based on K-means perception in a study [46], called the modified K-means (MK-means) clustering algorithm. The MK-means algorithm showed the effectiveness of clustering multidimensional data in Euclidean space. Therefore, in the current study, we improved, explained in detail and implemented the MK-means algorithm [46] to achieve better results. In particular, in the MK-means clustering algorithm, the number of \( k \) values plays an important role in the overall data analysis results. Thus, the Elbow method is
used to select the appropriate number of $k$ values for the MK-means clustering algorithm. In general, the Elbow method calculates the value of the sum of squared errors (SSE) in each cluster to determine the most appropriate number of clusters for a dataset. A small SSE value guarantees a good cluster in terms of quality, and SSE is calculated by formula 2.

$$SSE = \sum_{j=1}^{k} \sum_{z \in h_j} d^2(c_j, z)$$

where $k$ represents the initial number of clusters, $z$ represents a data point in cluster $h_j$, $c_j$ is the center data point of cluster $h_j$, and $d$ represents the distance among centers and data points.

On the other hand, the unsupervised clustering algorithm is used to group the most similar data from the large dataset by following some mathematical procedures. The K-means clustering algorithm is a simple and effective algorithm for partitioning data. Despite the effectiveness of K-means clustering, it has some limitations such as (i) determining the number of clusters, and (ii) optimal center selection. MK-means clustering algorithm is used to overcome the above weaknesses of K-means clustering. The MK-means clustering algorithm has introduced a unique idea for normalizing multidimensional data without using dimensionality reduction techniques such as non-negative matrix factorization (NMF), principal component analysis (PCA), and diffusion maps. The MK-means clustering algorithm is used to process the multidimensional education data, select the optimal centroid, and handle outliers. These features make this clustering algorithm more effective than other partition-based clustering algorithms. The MK-means clustering algorithm contains two (02) new modules: (i) an initial center selection module (ICSM) to select the optimal centers, and (ii) an outlier detection module (ODM) to identify and remove irrelevant data points from the dataset. The MK-means algorithm has been applied to a variety of data analysis and applications [24], [29]. The pseudocodes of these two modules ICSM and ODM of the MK-means clustering algorithm are shown in Algorithm 1 and Algorithm 2, respectively.

1) Multidimensional Data Clustering in Euclidean Space

The growing collection of structured, unstructured, and multidimensional data in a variety of media has created challenges in the field of data science. However, clustering of multidimensional data is a tricky and challenging task. Conventional clustering algorithms have limitations in clustering multidimensional data. To alleviate this problem, the MK-means clustering algorithm can efficiently handle multidimensional data in Euclidean space (numerical data) for clustering. Table 5 shows an example of a four-dimensional $(A_1, A_2, A_3, A_4)$ data set. According to the ICSM module of the MK-means clustering algorithm, the distance from the origin $(O)$ is calculated for each data point. Here we describe a mathematical procedure for calculating the distance of multidimensional data points.

![Table 5: An example of 4-dimensional dataset](image)

The mathematical model for multidimensional data normalization for clustering by ICSM (Algorithm 1) is as follows:

Let $Z = \{A_1, A_2, A_3, \cdots, A_n\}$ and $O = \{O_1, O_2, O_3, \cdots, O_n\}$ be the multidimensional data point and the origin, respec-
tively, where \( n \) is the number of dimensions. Now, the distance between \( Z \) and \( O \) can be expressed by Equation 3.

\[
D(O, Z) = \sqrt{(0_1 - A_1)^2 + (0_2 - A_2)^2 + \cdots + (0_n - A_n)^2}
\]

(3)

The generalized formula for the distance calculation between multidimensional data points and the origin is shown in Equation 4.

\[
\text{Distance } D(O, Z) = \sqrt{\sum_{i=1}^{N} (O - Z_{A_i})^2}
\]

(4)

where \( N \) is the number of dimensions, \( O \) is the origin, and \( Z_{A_i} \) is the data in each dimension of a data point.

Here we calculated the distance according to Equation 4 for each data point (in Table 5) as follows:

\[
d_{z_1} = \sqrt{(0 - 14)^2 + (0 - 5)^2 + (0 - 9)^2 + (0 - 12)^2} = 21.12
\]

\[
d_{z_2} = \sqrt{(0 - 7)^2 + (0 - 10)^2 + (0 - 11)^2 + (0 - 21)^2} = 26.66
\]

\[
d_{z_3} = \sqrt{(0 - 15)^2 + (0 - 9)^2 + (0 - 6)^2 + (0 - 10)^2} = 21.02
\]

\[
d_{z_4} = \sqrt{(0 - 20)^2 + (0 - 28)^2 + (0 - 17)^2 + (0 - 13)^2} = 40.52
\]

\[
d_{z_5} = \sqrt{(0 - 18)^2 + (0 - 24)^2 + (0 - 19)^2 + (0 - 15)^2} = 38.55
\]

Furthermore, as per the ICSM module (Algorithm 1), the data points are sorted (in ascending or descending order) based on their calculated distance values. Subsequently, the data points are partitioned according to the specified \( k \) values. In addition, optimal initial centers are selected and other clustering tasks are performed. In the clustering process, ODM is another effective module (Algorithm 2) that can detect and remove the most irrelevant data points from the dataset. In this way, the probability of selecting irrelevant data points as optimal initial centers is reduced. Figure 2 shows an example of two-dimensional data distribution in Euclidean space, where the distance of the data points from the origin can be calculated.

**B. FP-GROWTH ALGORITHM**

In the proposed approach, the FP-growth algorithm is leveraged for mining patterns and association rules. The FP-growth algorithm is faster than other algorithms and has the advantage of generating candidate itemsets after only two repetitions (scans) of the database. The FP-growth growth algorithm [60] forms with two main phases: (i) constructing the FP-tree, and (ii) mining the most FPs from the FP-tree. Let, \( E = \{e_1, e_2, e_3, \cdots, e_m\} \) be the set of all items in the TDBs. A TDB is constructed with the tuple of records \( R = \{r_1, r_2, r_3, \cdots, r_n\} \) where each record \( r_i \) is a subset of \( E \). An association rule is expressed as \( AR = Y \rightarrow Z \), where \( Y, Z \) is a subset of \( E \), \( Y \subseteq E, Z \subseteq E \) and \( Y \cap Z = \phi \). The set of items in \( Y \) is marked as predecessor (i.e., if), on the contrary, \( Z \) is named as successor (i.e., then). Two important terms, support (\( S \)) and confidence (\( C \)) are entailed for an association rule \( AR \), which is written as follows:

\[
S(AR) = \frac{\sigma(Y \cup Z)}{n}
\]

(5)

\[
C(AR) = \frac{\sigma(Y \cup Z)}{\sigma(Y)}
\]

(6)

The pseudocodes for FP-tree construction [60] in Algorithm 3 and FPs mining with the FP-tree [60] in Algorithm 4 are shown below.

**V. EXPERIMENTAL RESULTS**

In this section, we present the various experimental results. First, we visualize the data in clusters and extract the basic statistical features from each cluster. Next, we present data patterns (e.g., frequencies, ranking, number of rules, etc.) from each cluster, and finally, we show the relevant association rules extracted from each cluster.

**A. DETERMINING THE VALUE OF K AND DATA CLUSTERING**

To determine the suitable number (\( k \)) of clusters, we applied the Elbow method to the dataset. First, we select a random number of clusters and compute the \( SSE \) value in each cluster. In Figure 3, we can see that the \( SSE \) value is stable after the number of clusters \( k =4 \). According to the results
Algorithm 3  FP-tree construction

**Input:** Database: $TDB$, Minimum support (MinSup): $M_{sup}$

**Output:** FP-tree: $Tree$

**Repeat/Scan** full $TDB$ once to fetch the frequent items $F_i$ and supports of each item.

**Sort** $F_i$ in descending order (based on support) as $DF_i$

**Create** root node (null) of FP-tree $Tree$ for each transaction $Trns$ in $TDB$

- **Sort** items of each $Trns$ according to the $DF_i$ list.

- **Sorted** frequent items of each $Trns$ be $[s|S]$, where $s$ is a first item and the rest of the item is $S$.

- **if** $T$ has child $n$, $n.item - name = s.item - name$ **then**
  - **Increase** $n$’s count by 1

- **else**
  - Create a new node $n$ and increase $n$’s count by 1, $n$’s parent link be linked to $T$ and its node-link be linked to the nodes with the same item $- name$ via node-link structural path.

- **end**

- **if** $S \neq \Phi$ **then**
  - **Repeat** above If Statement

**end**

Algorithm 4  Mining FP with FP-tree

**Input:** Constructed FP-tree: $Tree$ and MinSup: $M_{sup}$

**Output:** FP

- **if** $Tree$ contains single path $P$ **then**
  - **for** each combination $\gamma$ of the nodes on path $P$ do
    - Create pattern $\gamma \cup \alpha$ with support := $M_{sup}$ of the nodes of $\gamma$
  - **end**

- **else**
  - **for** each item $a_i$ in the header of $Tree$ (in support ascending order) do
    - Create pattern $\gamma := a_i \cup \alpha$ with support := $a_i.support$
    - Construct conditional pattern base of $\gamma$ and $\gamma$’s conditional FP-tree $Tree_\gamma$
    - **if** $Tree_\gamma \neq \Phi$ **then**
      - **Repeat** the process
  - **end**

**end**

of the Elbow method shown in Figure 3, we obtained the appropriate number ($k = 4$) of clusters to perform the main clustering operation using the MK-means algorithm on the dataset (Table 2).

Next, we applied the MK-means and other random initial center selection (RCS) with K-means clustering algorithms to the dataset. The cluster data visualization by the RCS with K-means is shown in Figure 4a and the MK-means clustering algorithm is shown in Figure 4b. The results show that the MK-means clustering algorithm requires 17.33% less number of iterations to build clusters than the RCS with K-means clustering algorithm using our dataset. Furthermore, when we tested the MK-means clustering algorithm on approximately 110,000 synthetic data points, the algorithm performed better in terms of the number of iterations and $SSE$. Figure 4 shows that the data overlaps occurred during clustering when the K-means algorithm with RCS, while the MK-means algorithm clustered the data in a clean manner.
FIGURE 5: Overview of students and their submission logs’ statistics for each cluster

Figure 6 shows the average scores of ProgA, FCT and FPT for all clusters. Cluster 1 scored highest on all tests than the other clusters (2, 3, and 4). The scores decrease linearly from cluster 1 to 4, with the only exception that cluster 4 achieves a higher ProgA score than cluster 3.

FIGURE 6: Avg. scores of programming-related tests

Figure 7 shows the statistics of judge verdicts obtained by the students in each cluster of the ALDS course. Typically, verdicts are made immediately after submitting each solution. From the statistics shown in Figure 7, we can draw the following observations: (i) students in clusters 1 and 4 obtained the most AC verdicts whereas students in cluster 3 got the least AC verdicts; (ii) students in cluster 3 received the most error verdicts; and (iii) students in cluster 1 received the most TLEs among the other clusters.

B. DISCOVERING FREQUENT DATA PATTERNS BASED ON CLUSTERS

In this part of the experiment, the FP-growth algorithm is used to discover the frequent data patterns in each cluster. First, the frequency of different attributes in each cluster is calculated, then, the ranking of the attributes based on frequency is also enumerated. Next, we compute the frequent data patterns for each cluster by varying the minimum support (minSup) value. Note that minSup is used to find frequent itemsets from transactions in the database. For better understanding, we assume minSup = 3 for a TDB. In this case, an item a appears 4 times out of total 8 transactions in TDB, and since item a satisfies the minSup value (a ≥ minSup), item a is called a frequent item. In Figure 8, we enumerate the frequency of each attribute to investigate interesting patterns of attributes in each cluster. Since the values of the attributes are divided into different groups, and each group is represented by a specific ID. So, in each sub-figure of Figure 8, the X-axis represents the ID of the attribute, and the Y-axis represents the frequency.

Several observations can be made from this figure. (i) As shown in Figure 8a, most students are interested in solving the first 30 problems, but out of this common trend, students in cluster 1 attempted to solve all problems. In addition, students in clusters 2 and 3 solved more problems than the other clusters. (ii) In Figure 8b, students in cluster 2 achieved the most AC and WA verdicts among the other clusters based on their submissions. (iii) As shown in Figure 8c, students in all clusters maintain the same pattern for SAccu. Most of the students in cluster 2 achieved high SAccu than those of clusters 1, 3, and 4, at the same time students in clusters 2 and 3 obtained the most number of low SAccu. (iv) For the ProgA scores, as shown in Figure 8d, most of the students in cluster 1 obtained high scores, while most of the students in cluster 3 obtained low scores. (v) As shown in Figure 8e, students in clusters 3 and 4 did not achieve high scores (below 65%) in the FCT, instead most of them received low scores. However, more students from cluster 1 achieved high scores in the FCT than those of cluster 2. (vi) As shown in Figure 8f, most students in clusters 1 and 2 obtained high scores in the FPT, in contrast, more students from clusters 3 and 4 obtained low FPT scores.

Attribute ranking was calculated on the basis of the frequency values in each cluster. Figure 9 shows the frequency distribution of attributes in the TDB. The X-axis shows the ranking of the attributes in order of frequency values, and the Y-axis shows the frequency of the attributes. From this figure, the following observations can be drawn: (i) the distribution of attribute frequency is a long tail (or exponential) distribution; (ii) the frequency of a small number of attributes (especially the top 20 attributes) is high, and the frequency of the remaining attributes is relatively low; (iii) there are similarities in the frequency-based ranking patterns of each cluster. The attributes of clusters 2 and 3 are more frequent (higher frequency) than those of the other clusters.
Cluster 1
Cluster 2
Cluster 3
Cluster 4

0 10 20 30 40

0 10 20 30 40

0 10 20 30 40

0 10 20 30 40

%

%

%
Here, we summarized the frequencies of all clusters and created a ranking curve for the attributes, as shown in Figure 10. This ranking curve is similar to the curves for the individual clusters, reflecting the fact that the distribution of attribute frequencies is a long tail, with the first few attributes achieving high frequencies.

Next, we generated frequent patterns from the TDB. Figure 11 shows the number of frequent patterns available for each cluster at various minimum support (\(\text{minSup}\)) values. In order to generate patterns for each cluster, we diversified the value of \(\text{minSup}\) in the range of 500 to 3500. From this figure, several observations can be made. (i) As the \(\text{minSup}\) value for each cluster increases, the number of frequent patterns decreases. Because of many patterns could not satisfy the increasing \(\text{minSup}\) value. (ii) Clusters 2 and 3 have the highest number of patterns generated with any number of \(\text{minSup}\) values. Cluster 4 has the lowest number of patterns generated with any number of \(\text{minSup}\) values.

\section*{C. ASSOCIATION RULE MINING}

In this part of the experiment, we explore the available association rules based on different \(\text{minSup}\) and minimum confidence (\(\text{minConf}\)) values. Note that \(\text{minConf}\) is used to generate association rules and has no effect on mining frequent patterns from the TDB. \(\text{minConf}\) indicates how frequent an association rule is true in the TDB. For example, if attribute \(Y\) occurs 5 times and attributes \(Y\) and \(Z\) co-occur 3 times in the TDB, the confidence of rules \(Y \rightarrow Z\) is \(3/5 = 0.6\) (or 60%). As shown in Figure 12, we varied the values of \(\text{minConf}\) and \(\text{minSup}\) and showed the number of relevant rules in each cluster. First, we set several values of \(\text{minSup}\), such as 500, 1000, 1500, 2000, 2500, and 3000. For each value of \(\text{minSup}\), we varied the value of \(\text{minConf}\) to 50%, 60%, 70%, 80%, and 90%. As a result, clusters 2 and 3 generated the most association rules based \(\text{minSup}\) and \(\text{minConf}\) values. On the other hand, relatively few rules are generated on the basis of different \(\text{minSup}\) and \(\text{minConf}\) values in clusters 1 and 4.

Next, based on \(\text{minConf}\) and \(\text{minSup}\) values, the as-
association rules are constructed in each cluster. In order to generate association rules, we varied \( \text{minSup} \) values for each cluster because of the unequal number of transactions in clusters. However, the \( \text{minConf} = 90\% \) value was set for each cluster. For clusters 1 and 4, we set \( \text{minSup} = 1500 \). Similarly, for clusters 2 and 3, we set \( \text{minSup} = 2000 \) and \( \text{minSup} = 3000 \), respectively, to construct the association rules. Some of the generated association rules are shown in Table 6.

The association rules in Table 6 show the actual characteristics of the clusters, as well as the involvement of the students in various programming and academic activities. Based on the constructed association rules, the following observations can be made. Students in cluster 1 have higher scores (\( FCT \) and \( FPT \)), solution accuracy (\( S\text{Accu} \)), and the number of accepted (\( AC \)) verdicts. Students in cluster 2 have similar characteristics to those in cluster 1, but many of them have lower scores on the coding test (\( FCT \)). Similarly, we also assessed the characteristics of students in clusters 3 and 4. The association rules for clusters 3 and 4 are related to the low scores (\( PA, FCT, \) and \( FPT \)), solution accuracy (\( S\text{Accu} \)), and the number of accepted (\( AC \)) verdicts.

VI. DISCUSSION

In this section, we discuss the results of our experiments on data clustering, statistical features, data patterns, and association rules. Furthermore, recommendations based on the extracted features, patterns, and rules are discussed.

A. DATA CLUSTERING AND STATISTICAL FEATURES

Knowledge tracing from the vast archives of e-learning platforms, especially OJ systems, is always a challenge. Heterogeneous data and storage require preprocessing of the data to find accurate information. The problem-solving data is a type of complex data that includes submission logs, test scores, and code evaluation results. Table 1 and Table 2 show the submission logs and test scores of a programming course (ALDS), respectively. For pattern and rules mining, a TDB (Table 4) was created based on data preprocessing, new attribute creation (\( S\text{Accu} \)), and segmentation (Table 3). The data in Table 2 are clustered using the MK-means and Elbow methods, and four (04) clusters are created. Figure 4 shows the effectiveness of the MK-means clustering algorithm for clustering multidimensional data in Euclidean space. A number of statistical features are extracted from each cluster.

Figure 5 shows that the lowest number of students (16.01\%) and submission logs (7.36\%) belong to clusters 1 and 4, respectively. In contrast, the highest number of sub-
mission logs (31.59%) and students (33.33%) can be found in cluster 2 than those in clusters 1, 3 and 4. Figure 7 shows the verdicts received by students in each cluster against their submitted solutions. Students in cluster 1 received the highest AC verdicts (40.88%) compared to students in clusters 2, 3, and 4. On the other hand, students in cluster 3 received the highest (67.30%) error verdicts for different types of errors (WA, CE, RE, PE, TLE, MLE, and OLE). In addition, these verdicts (both accepted and errors) provide insight into the programming abilities of the students in each cluster. However, it is important to find more meaningful features among the many statistical hypothesized features that can be useful for improving programming learning. The following statistical features can be generated in each cluster to explore the hidden information from the problem-solving data. We have highlighted some important features: (i) tendency to submit each assignment on a daily basis, (ii) average, standard deviation, and variance of scores in different tests (ProgA, FCT, and FPT), (iii) statistics on additional problem solutions beyond the regular assignments, (iv) the accuracy of solutions in each cluster, and (v) tendency to solve accepted solutions to improve code efficiency.

### B. PATTERN AND ASSOCIATION RULE MINING

Data patterns and associated rules are mined from the problem-solving data of each cluster using the FP-growth algorithm. Based on the data segmentation (Table 3) and mathematical models, a TDB (Table 4) is created to search for patterns and rules. This process is useful for visualizing data patterns in detail. Figure 8 shows the frequencies of the data, where the frequencies of the individual segmented attributes are also calculated. Figure 8 provides some important insight, such as Figure 8a showing that students in cluster 1 attempted and solved all problems, and Figure 8e showing that students in clusters 3 and 4 did not obtain a high score (below 65%) on the final coding test. Thus, the segmentation of the scores provides a deeper insight into the patterns of the data that would not be visible without the segmentation.

The ranking of attributes based on frequencies is shown in Figure 9. A long-tail (or exponential) data distribution is observed, with the first 20 attributes having higher frequencies. Figure 11 shows the number of patterns generated based on the minSup values. Cluster 2 and cluster 3 generated the highest number of patterns for any minSup value. Furthermore, Figure 12 shows the number of association rules for different values of minSup and minConf. It can be seen that cluster 3 has the highest number of association rules, no matter how many values of minSup and minConf are used. In addition, some extracted association rules are listed in Table 6 for each cluster. The whole process of pattern and rule mining exposed the hidden information from the problem-solving data and can be useful for other real-world educational applications.

### C. ANALYSIS AND APPLICATION OF EDM RESULTS TO IMPROVE PROGRAMMING SKILLS

EDM enables to discover hidden features in problem-solving data, and the results can facilitate both students and teachers in many ways. We explored various features of each cluster, including verdicts (accepted and errors) (in Figure 7), average scores (ProgA, FCT, and FPT) (in Figure 6), number of submissions and students (in Figure 5). These characteristics can be useful in identifying students’ weaknesses and strengths in programming in each cluster. For example, cluster 1 received the most accepted verdicts (40.88%) and scores (ProgA, FCT, and FPT) than other clusters (2, 3, and 4). In contrast, clusters 3 and 4 received fewer accepted verdicts (32.70%, 39.19%) and scores (ProgA, FCT, and FPT) than cluster 1. Figure 7 shows that a large number of errors occurred/appeared in the solution codes, which were higher than the accepted verdicts in each cluster. Basically, the different types of error verdicts are caused by the different types of errors/mistakes in the solution code. There is an opportunity to improve students’ programming skills so that they can understand and handle these errors in the solution codes.
Normally, students in the ALDS course can discuss with each other to solve the problem during ProgA, but they cannot discuss during FCT (closed-book programming test). Figures 8d and 8e show that students in clusters 3 and 4 scored higher in ProgA when they were allowed to discuss with others, but they did not score higher in FCT when they were not allowed to discuss. Students in clusters 3 and 4 obtained average scores of 28.46 and 16.55 out of 120 in FCT, respectively, indicating poor programming skills (in Figure 6). Despite the lower number of submissions (in Figure 5), students in cluster 4 received higher accepted verdicts (39.19%) than students in clusters 2 and 3 (in Figure 7). These statistics show the weaknesses of actual programming skills of students in clusters 3 and 4. Therefore, appropriate intervention can help clusters 3 and 4 students to improve their programming skills.

Moreover, association rules show the correlation between attributes and how these attributes depend on each other. Based on different minSup and minConf values, we extracted and analyzed the association rules in each cluster (in Figure 12). Association rules based on higher minSup and minConf values are more important. The most frequent association rules for each combination of minSup and minConf revealed students’ programming and academic activities. Table 6 shows that students in cluster 4 involved with the lower accuracy, ProgA, FCT, FPT, and accepted verdict. Based on this analysis, teachers can take the necessary steps to improve students’ programming skills in each cluster.

**D. LEARNING AND TEACHING STRATEGIES FOR PROGRAMMING**

One of the main objectives of this research is to understand what difficulties students have in solving programming problems, identify the main influencing factors in their programming learning process, and determine what strategies, methods, or technologies can be used in teaching and learning to improve students’ programming skills. In Figure 5, it can be seen that although the number of students in cluster 4 is higher (18.62%) than in cluster 1, the students of cluster 4 submitted the lowest number of solution codes (7.36%) for evaluation compared to the other clusters (1, 2, and 3). These statistics also indicate that students in cluster 4 put less effort in solving the problem. Another important statistic we found in Figure 7 is that students in clusters 1, 2, 3, and 4 received about 51.75%, 51.71%, 52.42%, and 42.22% error verdicts (WA, RE, PE, TLE, MLE, and OLE) respectively in the solution codes that cannot be identified by the compilers or correctly recognized by the students. Understanding and reducing these errors (WA, RE, PE, TLE, MLE, and OLE) in the solution code is also a challenging task. Since students in all clusters rated about 50% of the submitted solutions as incorrect excluding compile error (CE), and these errors involved semantic, mathematical, and logical errors. To improve students’ mathematical and logical skills, they need different types of problems and a suitable practice environment. In addition, some other strategies may be useful, such as rapid response and continuous monitoring of students’ programming activities, programming workshops to address students’ weaknesses.

**E. RECOMMENDATION BASED ON FEATURES**

Based on statistical features, data patterns, and association rules, the performance of students in clusters 1 and 2 indicates that they are motivated and proficient in both programming and theoretical knowledge. Some additional strategies for students in clusters 1 and 2 could include (i) giving these students special treatment, (ii) assigning more complex problems to improve their skills, (iii) working on real-world problem-solving tasks, and (iv) participating in programming contests and workshops. On the other hand, the students in clusters 3 and 4 are relatively weak at programming and other academic tests, and there are possible ways to help them improve their performance. Also, Table 6 shows that most of the frequent association rules are linked with lower scores. Several measures can be taken to improve the performance of these students, such as (i) pay special attention to these students during lectures and observe their responses, (ii) encourage them to solve more problems to improve their programming skills, (iii) carefully monitor their performance, (iv) free discussions can be limited during programming assignments, and (v) additional workshop on closed-book/no-discussion programming practices.

**F. LIMITATIONS**

In this paper, all experimental results were obtained based on problem-solving data collected in a programming course (ALDS). Experimental results may vary for other programming courses’ data. The number of clusters (k) for the MK-means algorithm can vary depending on the data set. Also, different number of patterns and rules can be generated for different minSup and minConf values. Therefore, the proposed EDM framework can yield good or poor results on other data sets.

**VII. CONCLUSION**

In this paper, we proposed an EDM framework for data clustering, patterns, and rules mining using real-world problem-solving data. A mathematical model for data preprocessing, MK-means, and FP-growth algorithms were used to conduct this study. For programming education, OJ systems have been adopted by many institutions as academic tools. As a result, a huge number of programming-related resources (source codes, logs, scores, activities, etc.) are regularly accumulated in OJ systems. In this study, a large amount of real-world problem-solving data collected from the AOJ system was used in the experiments. Problem-solving data preprocessing is one of the main tasks to achieve accurate EDM results. Therefore, a mathematical model for problem-solving data preprocessing is developed. Then, the processed data are clustered using Elbow and MK-means algorithms. Various statistical features, data patterns and rules are ex-
tracted from each cluster based on different threshold values \((K, \text{minConf}, \text{minSup})\). These results can effectively contribute to the improvement of overall programming education. Moreover, based on the experimental results, some pertinent suggestions have been made. Furthermore, the proposed framework can be applied to other practical/exercise courses to demonstrate data patterns, statistical features, and rules. Besides, any third-party applications with similar data resources such as AlgoA, ProgA, FCT, and FPT, can use the proposed approach for EDM and analysis.

In the future, the experimental results of EDM using problem-solving data can be integrated to visualize different LA for programming platforms such as the OJ system. In addition, fuzzy estimation and polynomial approximation methods can be handy to dynamically select the optimal \text{minSup} values based on the dataset. Appropriate \text{minSup} values could help to generate the actual number of frequent elements and association rules from the dataset.

**AVAILABILITY OF PROBLEM-SOLVING DATA**

In this paper, we collected all problem-solving data for experiments from the Aizu Online Judge (AOJ) system. The problem-solving data can be accessed through these links https://onlinejudge.u-aizu.ac.jp and http://developers.u-aizu.ac.jp/index.

**CONFLICT OF INTEREST**

The authors declare that they have no conflicts of interest.

**REFERENCES**

[1] R. Rietsche, K. Duss, J. M. Persch, and M. Soellner. (2018). Design and evaluation of an IT-based formative feedback tool to foster student performance. In 39th international conference on information systems (ICIS) (pp. 1–17). San Francisco, CA, USA.

[2] S. Oeste, K. Lehmann, A. Janson, M. Sollner, and J. M. Leimeister. (2015). Redesigning university large scale lecture: How to activate the learner. 75th academy of management annual meeting. Vancouver: British Columbia.

[3] J. A. Kulik and J. D. Fletcher. (2016). Effectiveness of intelligent tutoring systems. Review of Educational Research, 86, 42–78. https://doi.org/10.3102/0034654315581420.

[4] Ma. W., Adesope, O. O., Nesbit, J. C., Liu, Q. (2014). Intelligent tutoring systems and learning outcomes: A meta-analysis. Journal of Educational Psychology, 106, 901–918. https://doi.org/10.1037/a0037123.

[5] J. C. Nesbit, O. O. Adesope, Q. Liu and W. Ma. “How Effective are Intelligent Tutoring Systems in Computer Science Education?,” 2014 IEEE 14th International Conference on Advanced Learning Technologies, 2014, pp. 99-103.

[6] K. Vannhelm. (2011). The relative effectiveness of human tutoring, intelligent tutoring systems, and other tutoring systems. Educational Psychologist, 46, 197–221. https://doi.org/10.1080/00461520.2011.611369.

[7] A. Janson, M. Sollner, and J. M. Leimeister. (2020). Ladders for Learning: Is Scaffolding the Key to Teaching Problem-Solving in Technology Mediated Learning Contexts? Academy of Management Learning Education, 19(4), 439–468.

[8] M. C. Kim, and M. J. Hannafin. (2011). Scaffolding problem solving in technology-enhanced learning environments (TELEs): Bridging research and theory with practice. Computers Education, 56, 403–417.

[9] T.A. Brush, and J. W. Saye. (2002). A summary of research exploring hard and soft scaffolding for teachers and students using a multimedia supported learning environment. The Journal of Interactive Online Learning, 19(4), 439–468.

[10] R. Knote, A. Janson, and J. M. Leimeister. (2020). Value Co-Creation in Smart Services: A Functional Affordances Perspective on Smart Personal Assistants. Journal of the Association for Information Systems.
UDAY R. KIRAN received the Ph.D. degree in computer science from the International Institute of Information Technology, Hyderabad, Telangana, India. He is currently a Associate Professor with the School of Computer Science and Engineering, The University of Aizu, Japan. He holds the position of a Project Assistant Professor with the Kitsuregawa Laboratory, Institute of Industrial Science, The University of Tokyo, Tokyo, Japan. He holds the position of a Researcher with the Social Big Data Research Collaboration Center, National Institute of Information and Communications Technology, Tokyo, Japan. His current research interests include data mining, parallel computation, air pollution data analytics, traffic congestion data analytics, recommender systems, and ICTs for Agriculture.

KEITA NAKAMURA received Ph.D. in Information Science from Hokkaido University, Japan in 2013. He worked as an Assistant Professor at Department of Information and Computer Engineering, National Institute of Technology, Gunma College, Japan from 2014 to 2016. He is currently an Associate Professor at the University of Aizu, Japan. His research interests include robot performance evaluation method, 3D object reconstruction, deep learning, robot vision, data mining, natural language processing.