Using a Hierarchical Clustering Algorithm to Explore the Relationship Between Students' Program Debugging and Learning Performance

Chao Hung Liu and Ting-Chia Hsu

Department of Technology Application and Human Resource, National Taiwan Normal University, Taiwan

ABSTRACT

The programming course poses a significant challenge for students who are just starting to learn a programming language. Many beginners, upon encountering an "ERROR" message from the system, tend to give up on learning. However, there are also students who persist in overcoming difficulties, exerting continued effort to complete their code, and achieving better learning outcomes. Therefore, this study aimed to cluster students based on their behavior during debugging in a programming course. It sought to explore the impact and differences among students in terms of program success and course grades within different debugging frequency clusters.

Keywords

Learning Analytics, Trial and Error, Agglomerative Hierarchical Clustering

1. Introduction

Computer programming courses have long been a significant challenge for students entering the field of information technology. This is because students must express their needs using computer-understandable terms, logic, and thinking, often encountering obstacles in the process (Feurzeig et al., 2011). This challenge is considered a global issue, as both introductory and advanced programming language courses face high dropout rates, creating substantial pressure on students and teachers who may have high expectations for themselves (Luxton,2016). Many students give up when confronted with multiple syntax and logic errors, indicating a potential lack of problem-solving skills and perseverance (Cheah,2020).

In light of these challenges, this study primarily explored students' behavior records during the debugging process of coding, as errors represent obstacles and setbacks. Whether students can progress through these setbacks will be a key factor in their improvement and success. The study was designed to analyze and cluster students' debugging behavior data in programming courses using the Learning Management System. The research was expected to address three main research questions:

- RQ1. Can errors made by students in coding be differentiated into distinct clusters?
- RQ2. Is there a difference in the number of successful program runs (Success_run) among students in different programming error clusters?
- RQ3. Is there a difference in course learning scores (Score) among students in different programming error clusters?

2. Related work

2.1. Learning analytics for online learning

Educational Data Mining (EDM) and Educational Process Mining (EPM) are data science approaches to analyze various Learning Management Systems (LMS)(Bogarín,Cerezo& Romero,2018). This data mining approach is an important part of learning analytics.

LAK-WS 2024: Joint Proceedings of LAK 2024 Workshops, March 18–19, Kyoto, Japan

Particularly with the rapid expansion of Massive Open Online Courses (MOOCs), the interaction between online educational resources and learners is stored in extensive databases, creating educational big data for interpretation by educators (Ruipérez et al., 2022).

In numerous studies on learning analytics, there is often an exploration of dropout rates (failure rates) on online education platforms (Qian,2022). Additionally, researchers have analyzed differences in learning behaviors on a platform (Tong & Zhan, 2023), ultimately aiming to predict students' learning achievements (grades)(Li,Du & Yu,2023). These studies frequently involve classifying learners based on their learning preferences, allowing them to adapt their learning experiences according to the platform's diverse learning paths, recommendation systems, personalized learning strategies, and more.

2.2. Trial and Error in Programming Learning

The learning of programming languages involves various aspects of learning, such as problem-solving skills, computational thinking, and syntax comprehension (Nouri, Zhang, Mannila & Norén, 2020). Importantly, students need to sustain high motivation and genuine engagement in coding to make progress in learning how to program (Silva & Silveira, 2020). From the perspective of beginners, the difficulty lies in the inability to break down large problems into smaller ones. When hearing some specific terms (such as recursion, arrays, etc.), students may understand how they work, but struggle to translate that understanding into actual code (Lister et al., 2004). In this context, it is crucial to encourage students to engage in trial and error, as the experiences gained from mistakes help students engage in self-reflection and can even stimulate strong motivation to find satisfactory answers and iterate through the trial-and-error process (Sicora , 2019).

Since the sixteenth century, people have sought solutions to problems by encountering stimuli that lead to subsequent actions, which may involve continuous trial and error until success or, in some cases, giving up, which results in task failure (Boswell , 1947). Learning is inherently challenging, with difficulties progressing from shallow to deep. For example, students may encounter "Errors" while programming, which could contribute to exacerbating the failure rate in programming courses (Porter, Guzdial, McDowell & Simon, 2013) but can also serve as the driving force for problem solving (Noh & Lee, 2020).

Programming education is often considered to have a high entry threshold, possibly due to insufficient problem-solving skills among students or ineffective use of learning materials (Cheah, 2020). Therefore, in programming courses, continuous trial and error by students is viewed as a positive behavioral performance. This practice signifies students' continuous attempts, whether in syntax or logical reasoning, until they produce results matching their expectations (Ye et al., 2022). Efforts in trial and error also help students enhance their self-efficacy, as they gain confidence in how to deal with errors and develop problem-solving skills (Ahn , Mao , Sung & Black, 2017).

2.3. Agglomerative Hierarchical Clustering in Online Courses

Hierarchical clustering is an unsupervised algorithm that organizes data points into a tree-like structure on a two-dimensional plane. It groups data points and produces a hierarchical structure based on the differences between data points (Alpaydin , 2020). Agglomerative hierarchical clustering is a bottom-up hierarchical clustering method that visualizes the hierarchical structure and underlying data clustering structure (Liu, Xu, Zeng & Ren, 2021). It is also a user-friendly and popular clustering algorithm.

The agglomerative hierarchical clustering process first assigns each object to its own cluster. It then uses distance or similarity measures (e.g. Euclidean distance for quantitative data, Manhattan distance for ordered but not necessarily quantitative data) or more complex methods (e.g. unweighted with arithmetic mean Pair group method (UPGMA) (Oyelade , 2019).

The algorithm proceeds as follows:

Based on N samples, there are initially N clusters, each cluster containing one sample.

Iteratively merges the two closest clusters based on the chosen distance or similarity measure until the number of clusters is reduced to 1 or reaches a user-specified number (Cichosz , 2014).

In each successive iteration, the algorithm merges the closest pair of clusters based on the similarity criterion of features between data points until all data are in one cluster (Sasirekha & Baby, 2013).

Hierarchical clustering helps analyze educational big data, helping researchers identify different student learning styles, achievements, and behaviors, as well as assess individual engagement levels (Hung, Liu, Liang & Su, 2020 ; Trivedi & Patel ,2020 ; Yang, Chen, Flanagan & Ogata, 2022).

3. Method

3.1. Data mining methods

This research incorporates the "Learning Behavior and Learning Strategies" dataset collected by Lu et al.(2022). This dataset predominantly consists of various actions recorded on a Learning Management System (LMS) as students engaged in learning programming. It encompasses a range of data points such as the number of errors generated, instances of code copying, frequency of code execution, and academic grades. The primary focus of the dataset is to capture the learning behaviors and strategies of students while they undertake programming tasks within the LMS environment(Lua et al., 2022).

For the clustering analysis, this study focused on the "viscode.csv" dataset, specifically using the "IndentationError," "NameError," "SyntaxError," and "TypeError" fields. These four types of errors were defined as indicators of programming trial-and-error, representing the problems and difficulties students encountered while running their code. The "Viscode-success_run" and "Score" fields were used as indicators to validate the effectiveness of clustering (Table 1), serving as the basis for addressing Research Questions 2 and 3 in the study.

Table 1
Program trial and error and verification field description table

Program Error Field Name	Program Error Field Introduction	Validation Field Name	Validation Field Introduction
PseudoID	The ID names of each student have been de-identified.	Cluster	This field describes the clusters obtained after grouping the program trial-and-error field.
IndentationError	This field describes when syntax errors occur related to incorrect indentation	Viscode-success_run	This field describes the number of successful program runs in the integrated development environment.
NameError	This field describes when local or global names are not found.	Score	This field describes the final learning score for the course.
SyntaxError	This field describes when the syntax parser encounters a syntax error.		

TypeError	This field describes when an operation or function is applied to inappropriate types of objects.

3.2. Dataset

This study compared actions taken by 452 students in a programming course using an integrated development environment, as recorded by the learning management system in the viscode.csv dataset. The data, preprocessed and de-identified, includes distinct class fields (a-i), fields for interactions with the integrated development environment, debugging attempts, execution counts, success running counts, and grades. After clustering, a new PseudoID field was introduced to represent a unique identifier for each student, also serving as an index after clustering(Ogata et al., 2017). Additionally, a Cluster field was added for conducting inter-group analysis of variances and comparing the correlation between program execution success and learning grades across clusters.

3.3. Designing Clustering Model

This research employs the agglomerative hierarchical clustering method from the Python sklearn.cluster module for systematic trial-and-error clustering. To avoid the skewing of results by any single feature, normalization is performed before clustering. This is critical as it prevents any one data column from exerting undue influence on the clustering outcome and maintains the robustness of the algorithm against outliers, which could be seen as noise.

Following this preparatory step, the clustering process commences. The distance metric adopted is the "Ward" method, designed to minimize the total within-cluster variance. Essentially, at each step, Ward's method selects two clusters to merge in a way that results in the least possible increase in total variance, thus preserving high similarity within the clusters.

Moreover, to ensure a balanced distribution of clusters, silhouette scores are utilized to assess the quality of clustering across different numbers of clusters, ranging from 2 to 9. Referencing Table 2, the study identifies 2 as the optimal number of clusters and proceeds with further data analysis using this configuration.

Table 2
Silhouette coefficient grouping score table

Number of clusters	cluster rating
2	0.58618
3	0.38826
4	0.35471
5	0.36514
6	0.25836
7	0.26924
8	0.22217
9	0.22226

4. Experiment results & discussion

4.1. Can errors made by students in coding be differentiated into distinct clusters?

This study utilized the scipy.cluster.hierarchy library in Python to perform clustering analysis. Through this library, hierarchical clustering results were computed and visualized, as shown in Fig. 1. The chart reveals two distinct clusters with noticeable distances between them. Cluster 0 comprises 414 student records, Cluster 1 includes 38 student records. This study further applied Principal Component Analysis (PCA) to reduce the dimensions of data consisting of student IDs and their trial-and-error behaviors. By transforming the data into a two-dimensional chart, we made it straightforward to compare these behaviors against student performance.

In this study, the Python library seaborn was utilized to create a heatmap (Fig. 2), which presents the average number of trial-and-error attempts by students across different clusters. The heatmap clearly shows that students in Cluster 1 had a higher frequency of programming errors, such as IndentationError, NameError, SyntaxError, and TypeError, compared to those in Cluster 0. Notably, there are significant differences in the occurrences of NameError, SyntaxError, and TypeError between Clusters 1 and 0. Cluster 1 is characterized as the "Frequent Trial-and-Error Group," while Cluster 0 is referred to as the "Regular Trial-and-Error Group" in this study.

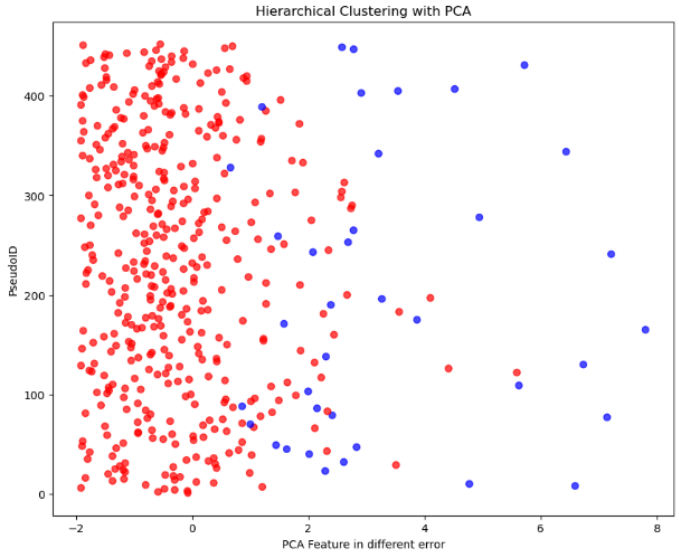

Figure 1: Hierarchical Clustering Dendrogram

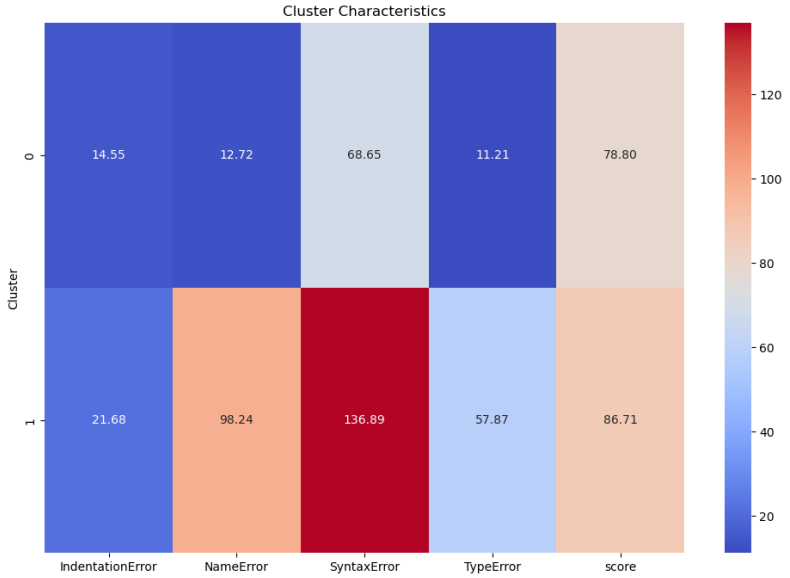

Figure 2: Heat map of program trial and error performance of different clusters

4.2. Is there a difference in the number of successful program runs (Success_run) among students in different programming error clusters?

To address research question 2, the study analyzed the "Success_run" field, revealing through Figure 3 that students in the frequent trial-and-error cluster had higher average successful program runs compared to those in the infrequent trial-and-error cluster. This pattern suggests that students who frequently encountered system errors in the integrated development environment were more persistent, leading to more successful code executions.

The study further employed an independent sample T-test, using IBM SPSS, to investigate statistical differences between the clusters in terms of successful program runs. The findings showed a statistically significant difference ($t = -5.49$, $p < .05$), where the frequent trial-and-error cluster outperformed the regular trial-and-error cluster in successful program executions. This discrepancy likely arises from the frequent trial-and-error students' resilience and continuous engagement with problem-solving and coding adjustments, in contrast to students in the regular trial-and-error cluster who may have experienced reduced motivation and task completion rates after facing setbacks. Consequently, the frequent trial-and-error cluster exhibited a significantly higher number of successful program operations compared to the regular trial-and-error cluster, highlighting their effective learning and problem-solving approach.

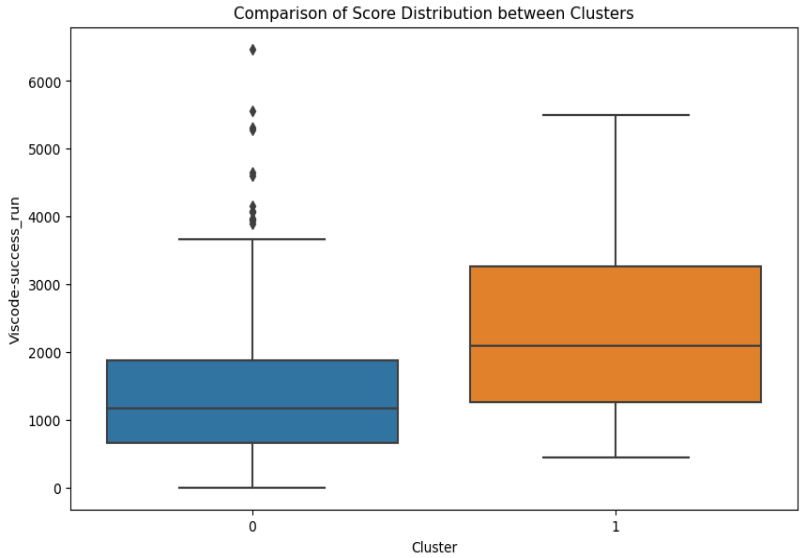

Figure 3: Box plot of program trial and error with different groups of successfully running code]

Table 3

Successfully running code and trial and error clusters were subjected to Independent Samples Test.

Independent Samples Test						
Viscodesuccess_run						
	N	Mean	SD	Sig.	t	*d*
frequent cluster	414	1358.88	1016.89	.023	-5.49*	0.39
regular cluster	38	2323.34	1243.08		-4.64*	

*. The mean difference is significant at the 0.05 level.

4.3. Is there a difference in course learning scores (Score) among students in different programming error clusters?

To explore Research Question 3, the study performed a descriptive analysis of the "score" field across clusters differentiated by trial-and-error frequency. Figures 2 and 4 illustrate that clusters characterized by frequent trial-and-error tend to have higher scores than those with regular trial-and-error patterns. The box plot indicates that scores for the frequent trial-and-error group are predominantly ranged between 80 and 90 points. In contrast, the regular trial-and-error group not only exhibited wider score fluctuations (70-85 points) but also presented numerous outliers significantly below the average, with some scores approaching zero. This suggests potential dropout behaviors among certain students in the regular trial-and-error group.

The independent samples T-test statistics revealed a significant disparity in course performance between the two trial-and-error groups ($t = -2.69$, $p < .05$) as shown in Table 4. This finding resonates with the insights from RQ1 and Figure 1.

The study segmented program-based learning students by their trial-and-error occurrences, noting that a smaller contingent of students (38 in total) fell into the frequent trial-and-error group. Not with standing their group size, these students not only surpassed the regular trial-and-error group in the number of successful program runs but also outscored them. This underscores the significance of persistent trial-and-error efforts in learning; it is imperative for learners to persistently experiment and overcome challenges without capitulation to achieve success(Dong et al., 2019).

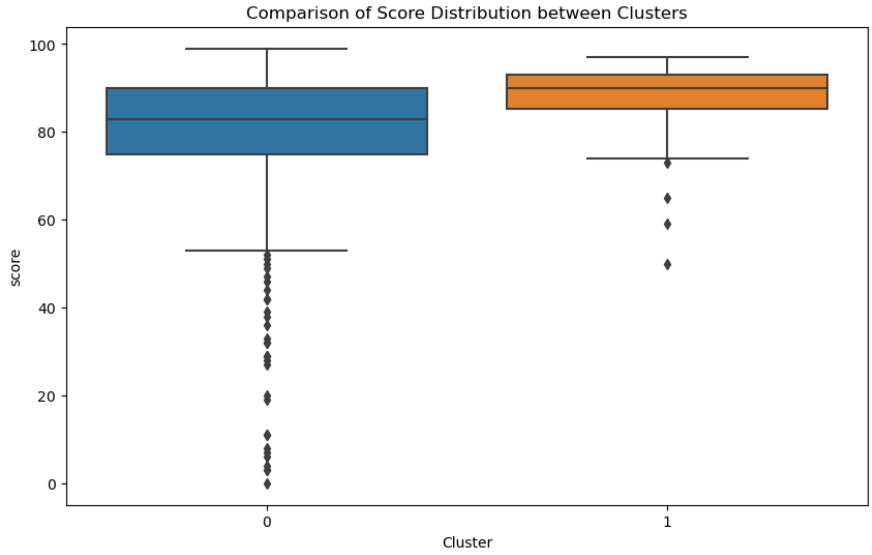

Figure 4: Box plot of program trial and error with different groups of course score

Table 4

Course score and trial and error clusters were subjected to Independent Samples Test.

Independent Samples Test						
	Score					
	N	Mean	SD	Sig.	t	*d*
frequent cluster	414	78.80	17.87	.020	-2.69*	0.26
regular cluster	38	86.71	10.31		-4.19*	

*. The mean difference is significant at the 0.05 level.

5. Conclusion

This study investigated the error patterns students demonstrated while debugging programs, categorizing them into two distinct groups: Cluster 0, the regular trial-and-error cluster, and Cluster 1, the frequent trial-and-error cluster. The analysis revealed that students in the frequent trial-and-error cluster not only had higher average course success rates and final scores compared to their counterparts in the regular trial-and-error cluster but also performed significantly better. These findings suggest that the debugging behaviors of different student groups can affect their learning outcomes. Educators should be cognizant of these differences and strategize appropriate responses to students' programming challenges. Moreover, when the frequency of trial-and-error attempts begins to wane, interventions such as encouragement or providing cues might be necessary to bolster students' motivation and align them with their peers(Xu, Yang, Liu & Jin, 2023).

Nonetheless, it is crucial to acknowledge the limitations of the dataset and context of this study. Given that all participants were novices in programming and enrolled in the same course, the breadth of their acquired knowledge may be limited. This research advocates for the use of more varied datasets and extended observational periods. There is also considerable variation in the amount of time different students dedicate to studying programming. While some engaged with the course material for over seven hours, others may invested mere minutes. As this study lacks precise data on students' study timings, future research could employ time series analysis to discern behavioral shifts among clusters and predict how debugging frequency might influence future learning. Such an approach could yield more precise learning recommendations and enable educators to tailor their focus on the effects of trial-and-error activities on academic achievement.

Acknowledgements

This study is supported in part by the National Science and Technology Council in the Republic of China under contract numbers NSTC 112-2628-H-003-007-.

6. References

[1] Feurzeig, W., Papert, S. A., & Lawler, B. (2011). Programming-languages as a conceptual framework for teaching mathematics. *Interactive Learning Environments*, *19*(5), 487-501. https://doi.org/10.1080/10494820903520040

[2] Luxton-Reilly, A. (2016). Learning to program is easy. Paper presented at the *Proceedings of the 2016 ACM Conference on Innovation and Technology in Computer Science Education*. (pp. 284-289). https://doi.org/10.1145/2899415.2899432

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

[4] Bogarín, A., Cerezo, R., & Romero, C. (2018). A survey on educational process mining. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, *8*(1), e1230. https://doi.org/10.1002/widm.1230

[5] Ruipérez-Valiente, J. A., Staubitz, T., Jenner, M., Halawa, S., Zhang, J., Despujol, I., Rohloff, T. (2022). Large scale analytics of global and regional MOOC providers: Differences in learners' demographics, preferences, and perceptions. *Computers & Education*, *180*, 104426. https://doi.org/10.1016/j.compedu.2021.104426

[6] Qian, Y., Li, C. X., Zou, X. G., Feng, X. B., Xiao, M. H., & Ding, Y. Q. (2022). Research on predicting learning achievement in a flipped classroom based on MOOCs by big data analysis. *Computer Applications in Engineering Education*, *30*(1), 222-234. https://doi.org/10.1002/cae.22452

[7] Tong, Y., & Zhan, Z. (2023). An evaluation model based on procedural behaviors for predicting MOOC learning performance: Students' online learning behavior analytics and algorithms

construction. *Interactive Technology and Smart Education.* https://doi.org/10.1108/ITSE-10-2022-0133

[8] Li, S., Du, J., & Yu, S. (2023). Diversified resource access paths in MOOCs: Insights from network analysis. *Computers & Education*, *204*, 104869. https://doi.org/10.1016/j.compedu.2023.104869

[9] Nouri, J., Zhang, L., Mannila, L., & Norén, E. (2020). Development of computational thinking, digital competence and 21st century skills when learning programming in K-9. *Education Inquiry*, *11*(1), 1-17. https://doi.org/10.1080/20004508.2019.1627844

[10] Silva, J., & Silveira, I. (2020). A systematic review on open educational games for programming learning and teaching. *International Journal of Emerging Technologies in Learning (IJET)*, *15*(9), 156-172. https://doi.org/10.3991/ijet.v15i09.12437

[11] Lister, R., Adams, E. S., Fitzgerald, S., Fone, W., Hamer, J., Lindholm, M., Seppälä, O. (2004). A multi-national study of reading and tracing skills in novice programmers. *ACM SIGCSE Bulletin*, *36*(4), 119-150. https://doi.org/10.1145/1041624.1041673

[12] Sicora, A. (2019). Reflective practice and learning from mistakes in social work student placement. *Social Work Education*, *38*(1), 63-74. https://doi.org/10.1080/02615479.2018.1508567

[13] Boswell, F. P. (1947). Trial and error learning. *Psychological review*, *54*(5), 282. https://doi.org/10.1037/h0058921

[14] Porter, L., Guzdial, M., McDowell, C., & Simon, B. (2013). Success in introductory programming: What works? *Communications of the ACM*, *56*(8), 34-36. https://doi.org/10.1145/2492007.2492020

[15] Noh, J., & Lee, J. (2020). Effects of robotics programming on the computational thinking and creativity of elementary school students. *Educational technology research and development*, *68*, 463-484. https://doi.org/10.1007/s11423-019-09708-w

[16] Ye, Z., Jiang, L., Li, Y., Wang, Z., Zhang, G., & Chen, H. (2022). Analysis of Differences in Self-Regulated Learning Behavior Patterns of Online Learners. *Electronics*, *11*(23), 4013. https://doi.org/10.3390/electronics11234013

[17] Ahn, J.-H., Mao, Y., Sung, W., & Black, J. B. (2017). Supporting debugging skills: Using embodied instructions in children's programming education. Paper presented at the *Society for Information Technology & teacher education international conference.* (pp. 19-26)

[18] Alpaydin, E. (2020). *Introduction to machine learning*: MIT press.

[19] Liu, N., Xu, Z., Zeng, X.-J., & Ren, P. (2021). An agglomerative hierarchical clustering algorithm for linear ordinal rankings. *Information Sciences*, *557*, 170-193. https://doi.org/10.1016/j.ins.2020.12.056

[20] Oyelade, J., Isewon, I., Oladipupo, O., Emebo, O., Omogbadegun, Z., Aromolaran, O., Olawole, O. (2019). Data clustering: Algorithms and its applications. Paper presented at the *2019 19th International Conference on Computational Science and Its Applications (ICCSA)*. (pp. 71-81). IEEE. 10.1109/ICCSA.2019.000-1

[21] Cichosz, P. (2014). *Data mining algorithms: explained using R*: John Wiley & Sons.

[22] Sasirekha, K., & Baby, P. (2013). Agglomerative hierarchical clustering algorithm-a. *International Journal of Scientific and Research Publications*, *83*(3), 83.

[23] Hung, H.-C., Liu, I.-F., Liang, C.-T., & Su, Y.-S. (2020). Applying educational data mining to explore students' learning patterns in the flipped learning approach for coding education. *Symmetry*, *12*(2), 213. https://doi.org/10.3390/sym12020213

[24] Trivedi, S., & Patel, N. (2020). Clustering Students Based on Virtual Learning Engagement, Digital Skills, and E-learning Infrastructure: Applications of K-means, DBSCAN, Hierarchical, and Affinity Propagation Clustering. *Sage Science Review of Educational Technology*, *3*(1), 1-13. https://journals.sagescience.org/index.php/ssret/article/view/6

[25] Yang, A. C., Chen, I. Y., Flanagan, B., & Ogata, H. (2022). How students' self-assessment behavior affects their online learning performance. *Computers and Education: Artificial Intelligence*, *3*, 100058. https://doi.org/10.1016/j.caeai.2022.100058

[26] Lua, O. H., Huang, A. Y., Flanaganc, B., Ogata, H., & Yang, S. J. A. (2022) *Quality Data Set for Data Challenge: Featuring 160 Students' Learning Behaviors and Learning Strategies in a Programming Course*. Proceedings of the 30th International Conference on Computers in Education. Kuala Lumpur City,Malaysia.

[27] Ogata, H., Oi, M., Mohri, K., Okubo, F., Shimada, A., Yamada, M., Hirokawa, S. (2017). Learning analytics for e-book-based educational big data in higher education. *Smart sensors at the IoT frontier*, 327-350. https://doi.org/10.1007/978-3-319-55345-0_13

[28] Dong, Y., Marwan, S., Catete, V., Price, T., & Barnes, T. (2019). Defining tinkering behavior in open-ended block-based programming assignments. Paper presented at the *Proceedings of the 50th ACM Technical Symposium on Computer Science Education*. (pp. 1204-1210). https://doi.org/10.1145/3287324.3287437

[29] Xu, W., Yang, L. Y., Liu, X., & Jin, P. N. (2023). Examining the effects of different forms of teacher feedback intervention for learners' cognitive and emotional interaction in online collaborative discussion: A visualization method for process mining based on text automatic analysis. *Education and Information Technologies*, 1-27. https://doi.org/10.1080/10494820903520040