attarbawiy
Malaysian Online Journal of Education

Attarbawiy: Malaysian Online Journal of Education Vol. 8, No. 2 (2024), 39-46

# THE EFFECTS OF MACHINE-LEARNING PROGRAMMING SIMULATOR ON UNIVERSITY STUDENTS' ENGAGEMENT IN LEARNING PROGRAMMING

*(Kesan Simulator Pengaturcaraan Pembelajaran Mesin Terhadap Penglibatan Pelajar Universiti)*

***Tansa Trisna Astono Putri[1], Wan Ahmad Jaafar Wan Yahaya[1], Nur Azlina Mohamed Mokmin[1], Sriadhi[2], Muhammad Fadhiel Alie[3]**

Submited: 08-Aug-2024
Accepted: 19-Aug-2024
Revised: 19-Aug-2024
Published: 28-Dec-2024

How to cite:
Putri, T. T. A., Wan Yahaya, W. A. J., Mohamed Mokmin, N. A., Sriadhi, S., & Alie, M. F. (2024). The effects of machine-learning programming simulator on university students' engagement in learning programming: Kesan simulator pengaturcaraan pembelajaran mesin terhadap penglibatan pelajar universiti. ATTARBAWIY: Malaysian Online Journal of Education, 8(2), 39–46. https://doi.org/10.53840/attarbawiy.v8i2.228



[1]Centre for Instructional Technology and Multimedia,
Universiti Sains Malaysia,
11800 Penang, Malaysia.

[2]Information Technology and Computer Education
Study Program,
Universitas Negeri Medan,
Medan, Indonesia.

[3]Information System Study Program
Universitas Indo Global Mandiri,
Palembang, Indonesia.

*Corresponding author's email: wajwy@usm.my

## Abstract

A computer programming course is essential for students pursuing a computer science major. The university offers a course that imparts essential knowledge to undergraduate computer science students. The study emphasizes the need for integrating technology, including gamification and simulation tools, in programming education to enhance student engagement and learning outcomes. A quasi-experimental study was conducted to investigate the impact of a machine learning (ML) programming simulator on student engagement. The study involved 120 university students, divided into two groups: one using the ML programming simulator and the other using a non-ML simulator. The results showed that students using the ML simulator had higher engagement levels compared to those using the non-ML simulator, suggesting that ML-based tools can significantly improve student participation and learning in programming courses. The findings underscore the potential benefits of using advanced technological tools to foster better learning experiences in computer programming education.

**Keywords:** engagement; university students; Machine-Learning simulator; programming

## *Abstrak*

*Kursus pengaturcaraan komputer adalah penting untuk pelajar yang mengikuti jurusan sains komputer. Dalam pembelajaran di universiti menawarkan kursus yang memberikan pengetahuan penting kepada pelajar sarjana sains komputer. Kajian ini menekankan keperluan untuk menyepadukan teknologi, termasuk alat gamifikasi dan simulasi, dalam pendidikan pengaturcaraan untuk meningkatkan penglibatan pelajar dan hasil pembelajaran. Kajian kuasi eksperimen telah dijalankan untuk menyiasat kesan simulator pengaturcaraan pembelajaran mesin (ML) terhadap penglibatan pelajar. Kajian ini melibatkan 120 pelajar universiti, dibahagikan kepada dua kumpulan: satu menggunakan simulator pengaturcaraan ML dan satu lagi menggunakan simulator bukan ML. Keputusan menunjukkan bahawa pelajar yang menggunakan simulator ML mempunyai tahap penglibatan yang lebih tinggi berbanding dengan mereka yang*

Doi: https://doi.org/10.53840/attarbawiy.v8i2.228

*menggunakan simulator bukan ML, menunjukkan bahawa alatan berasaskan ML boleh meningkatkan penyertaan dan pembelajaran pelajar dengan ketara dalam kursus pengaturcaraan. Penemuan ini menggariskan potensi manfaat menggunakan alat teknologi canggih untuk memupuk pengalaman pembelajaran yang lebih baik dalam pendidikan pengaturcaraan komputer.*
***Kata kunci:*** *penglibatan; pelajar universiti; Simulator Pembelajaran Mesin; pengaturcaraan*

## 1.0 INTRODUCTION

Computer programming courses are essential for computer science students, providing foundational knowledge required for success in the field. These courses equip students with the technical skills necessary for careers in software development, entrepreneurship, and other computer-related professions. Students who lack proficiency in programming may struggle to secure employment in roles such as software developers, highlighting the importance of early mastery of these skills (Angeli & Valanides, 2020). As a result, programming is a mandatory requirement for computer science students, ensuring that they acquire not only basic computational skills but also the ability to solve complex problems through code.

Despite the growing demand for skilled programmers globally, employment trends present a mixed picture. The U.S. Bureau of Labor Statistics projects a 10% decline in programming jobs between 2021 and 2031, largely due to automation of routine tasks. However, this decline is offset by the creation of more strategic roles, with an expected annual increase of 9,600 job openings due to retirements and transitions to other sectors. In Indonesia, the demand for programmers is even more pressing, with an estimated need for 600,000 programmers by 2025, while the current workforce stands at only 100,000 (Badan Pengkajian dan Penerapan Teknologi, 2023). This gap underscores the need for enhanced education methods to equip future programmers with the necessary skills to meet industry demands.

Technology plays a vital role in addressing these challenges, particularly in enhancing the learning experience in programming courses. Integrating tools such as gamification, visualization techniques, and assessment platforms can significantly improve student engagement and learning outcomes. However, despite these advancements, acquiring programming skills remains a complex task for novice students. Many struggle with not only learning the syntax of programming languages but also understanding the logic and problem-solving principles behind their code (Baist & Pamungkas, 2017). This leads to rote memorization of code rather than a deeper understanding of the concepts, which ultimately hinders their ability to solve new problems.

Student engagement, defined as the level of focus, curiosity, and enthusiasm in the learning process, is critical for success in programming education. Yet, many students report feeling disengaged and bored, which can lead to higher dropout rates, particularly when faced with the challenges of learning new programming paradigms (Isiaq & Jamil, 2017). Tools like simulation platforms offer a promising solution to this issue, enabling students to visualize and interact with coding structures in ways that promote active learning. For example, platforms like CodeOcean and Proteus have been shown to enhance student engagement and improve learning outcomes by providing timely feedback and facilitating hands-on experimentation (Budi et al., 2021; Staubitz et al., 2016).

Despite these advances, there remains a gap in the development of simulation tools specifically tailored to the PHP programming language. This study aims to address that gap by developing a PHP-specific simulator designed to enhance

Doi: https://doi.org/10.53840/attarbawiy.v8i2.228

student engagement, foster experimentation, and support creativity in programming education.

## 2.0 PROBLEM STATEMENT

Programming is an essential discipline that computer engineering majors must fulfill in order to obtain their degree. Indeed, instructors and institutions have challenges when students do not succeed in their programming courses. According to Alturki (2016), variables such as teaching techniques, instructors' subject-matter experience, students' knowledge and competency in cryptography, students' self-discipline, the learning environment, and student market resources all contribute to low student performance in programming. Questionnaire have revealed that undergraduate students have difficulty with some programming subjects, with looping being identified as the most challenging aspect of the programming course. According to the findings of the questionnaire, as shown in Figure 1, this study indicates that mastering programming grammar is the most challenging aspect of studying programming courses for students. A significant proportion of student respondents, namely 67%, identified it as a challenge when studying programming courses. This pertains to the proficiency of pupils in managing faults. Without a thorough understanding of programming grammar, individuals will be unable to effectively address any issues that may arise in their code. This aligns with the issues that the researcher previously addressed.

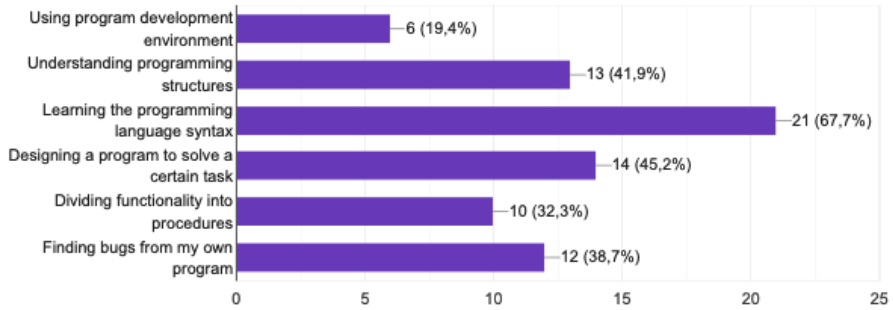


Figure 1 Difficult Issues in Learning Programming Course

Research conducted by Owolabi et al. (2014) has shown that computer anxiety has an impact on computer abilities. A negative correlation exists between mathematics anxiety and academic achievement (Ashcraft & Kirk, 2001). The study discovered that individuals who have mathematics anxiety tend to develop a negative attitude towards computers, which in turn has a notable effect on their performance in computer programming (Owolabi et al., 2014). Students who exhibit higher levels of comfort while engaging in programming-related activities have superior performance compared to those who experience moderate levels of anxiety about programming in any programming course. Consequently, it is very advisable to minimize computer anxiety, programming anxiety, and mathematics anxiety in order to achieve optimal programming outcomes.

While technology offers various tools to teach programming concepts—such as gamification, assessment tools, and visualization—its use in higher education programming courses remains underutilized. Programming is a specialized skill that requires significant practice, yet many college students primarily learn through passive methods like studying analytical literature or closely following their lecturers. This traditional approach often leads to suboptimal outcomes, as students struggle to develop the advanced cognitive abilities needed for programming, such as effective

Doi: https://doi.org/10.53840/attarbawiy.v8i2.228

planning, problem-solving, and logical thinking (Harimurti et al., 2021). Recognizing these challenges, researchers have explored the need for more dynamic and engaging learning environments that actively involve students in the learning process. The focus is on creating relevant learning experiences, providing sufficient support and resources, enabling collaboration and hands-on practice, and fostering a positive learning community to enhance programming mastery.

## 3.0 METHODOLOGY

The researcher conducted this experimental study to investigate the effects of a programming simulator on student engagement in a higher education programming course. The study compared students using the ML programming simulator with those using the noML programming simulator. The design consists of two treatment groups: the ML group and the noML group.

This study employed a quasi-experimental design. A quasi-experimental study design allows the researcher to control the assignment of participants to different groups while simultaneously attempting to determine the reasons for any pre-existing disparities among the groups (Gay et al., 2012; Shadish et al., 2002). Due to the impracticality of subject randomization, a quasi-experimental design was selected. While actual experiments are often regarded as more robust than quasi-experimental studies, the latter might be advantageous in situations when randomization is not feasible. The participants in this study are selected using a random assignment procedure from a group of students in a Programming course. As a result, the present groups, namely the treatment and control groups, differ in terms of their prior skill.

The sample size of this study consists of 120 university students from computer science major. A total of 60 students are given access to the machine learning programming simulator. Another total of 60 students are being introduced to the noML programming simulator. The data describing the four experimental groups used in this investigation may be found in Table 1.

Table 1 Characteristic of Sample

| **Model** | **Number of Students** | **Percentage (%)** |
|---|---|---|
| ML Programming Simulator | 60 | 50% |
| Non ML Programming Simulator | 60 | 50% |
| Total | 120 | 100% |

The research population for this study was selected using a representative strategy in non-probability sampling. The study's population comprised computer science students in Indonesia who were required to take programming classes. Representative sample was employed due to the ease of obtaining and controlling the help of both students enrolled in programming courses and instructors in this study. Three instructors with over a decade of expertise in teaching programming were engaged in this study to authenticate and assess the process of data collecting.

Sampling is the act of selecting a group of individuals from a larger population in a manner that accurately reflects the characteristics of the population (Ashmore et al., 2021). Sampling is a crucial technique in scientific investigations since the target population is often too large to include as participants in the research. An ideal test case is one that is scientifically representative of the target population and sufficiently large to meet the research inquiry.

Doi: https://doi.org/10.53840/attarbawiy.v8i2.228

Medan is selected as a representative sample of the Indonesian population for this investigation. The university is selected by purposive sampling, namely Universitas Negeri Medan, due to its distinctive feature of programming being a compulsory course. In this study, purposive selection is employed to choose the university, whereas random sampling is utilized to select the correspondents from the pool of undergraduate students. This study exclusively focuses on college students. The experiment class is composed of students from the batch years 2021 and 2022. Consequently, the total sample size is around one hundred and twenty students. The selection of undergraduate students for this study is conducted using random sampling.

## 4.0 RESULTS

This study examined the main effects of two independent variables, the Machine Learning (ML) programming simulator and the Non Machine Learning (NoML) programming simulator, on the dependent variable. A one-way analysis of variance (ANOVA) was used to see if there was a statistically significant difference in student participation between the group of interest and the other groups. Before calculating the engagement score, the mean and standard deviation of the scores were calculated for each group. The descriptive information on students' involvement scores across all formats is shown in Table 3.

According to Table 2, the descriptive statistical analysis reveals that 60 students were taught using an ML programming simulator, whereas another 60 students did not receive any instruction using the same simulator. The mean engagement score for a machine learning programming simulator is 4.14, whereas the mean engagement level for a non-machine learning programming simulator is 3.97. The engagement scores of students who utilized an ML programming simulator were higher compared to those who did not, as seen by the presented table.

Table 2 Results of Students' Engagement of NoML Programming Simulator and ML-Programming Simulator

| Mode | | Pretest Engagement Score | Posttest Engagement Score |
|---|---|---|---|
| NoML Programming Simulator | N | 60 | 60 |
| | Mean | 3.35 | 3.97 |
| | Std. Deviation | 0.850 | 0.350 |
| | Std. Error of Mean | 0.109 | 0.045 |
| ML Programming Simulator | N | 60 | 60 |
| | Mean | 3.14 | 4.14 |
| | Std. Deviation | 0.632 | 0.459 |
| | Std. Error of Mean | 0.081 | 0.059 |
| Total | N | 120 | 120 |
| | Mean | 3.26 | 4.06 |
| | Std. Deviation | 0.752 | 0.416 |
| | Std. Error of Mean | 0.068 | 0.038 |

The results of the ANOVA test (Table 3) showed a significant difference in the engagement ratings between the two groups [$F = 5.533$, $p=0.020$, $p < 0.05$]. This indicates that students who were exposed to ML programming simulator therapy exhibited higher levels of engagement compared to those who did not get ML programming simulator treatment.

Doi: https://doi.org/10.53840/attarbawiy.v8i2.228

Table 3 ANOVA Test of Engagement Score

| Source | Type III Sum of Squares | Df Value | Mean Square | F Value | Sig. |
|---|---|---|---|---|---|
| Corrected Model | 0.925a | 1 | 0.925 | 5.533 | 0.020 |
| Intercept | 1978.140 | 1 | 1978.140 | 11836.399 | <0.001 |
| GROUP | 0.925 | 1 | 0.925 | 5.533 | 0.020 |
| Error | 19.721 | 118 | 0.167 | | |
| Total | 1998.786 | 120 | | | |
| Corrected Total | 23.143 | 119 | | | |

According to the output table 4 for the "Independent Samples Test," it may be inferred that based on the observation that the Sig. (2-tailed) value is 0.020, which above the significance limit of 0.05. Therefore, it can be concluded that the level of student engagement with ML programming simulators varies greatly compared to non-ML programming simulators.

Furthermore, the output table above clearly indicates that the value of the "Mean Difference" is 0.1758. This value represents the difference in average student engagement between the ML programming simulator group and the noML programming simulator group. The difference spans from 0.0277 to 0.3233, with a 95% confidence interval.

Table 4 t-Test of Engagement Score

| | t-test for Equality of Means | | | | | | |
|---|---|---|---|---|---|---|---|
| | t | df | Sig. (2-tailed) | Mean Diff. | Std. Error Diff. | 95% Confidence Interval of the Difference | |
| | | | | | | Lower | Upper |
| PosttestEng | 2.352 | 118 | 0.020 | 0.1758 | 0.0746 | 0.0277 | 0.3233 |

## 5.0 DISCUSSION

The study's findings suggest that the use of the ML programming simulator has a substantial impact on students' engagement in a programming course. The study revealed that students who were exposed to the ML programming simulator exhibited higher levels of engagement compared to their peers who did not use the ML programming simulator. Simulators are apparatuses that replicate behaviors and procedures seen in the actual world. Here, the gadget replicates the stages of system occurrences and displays the necessary procedures to successfully carry out this sort of task. There is a wide array of simulation programs and methodologies that frequently differ across distinct fields. Simulation has demonstrated its use in establishing significant and intentional learning settings across several educational domains, especially when confronted with little understanding of assembly and programming (Jamil & Isiaq, 2019).

Previous research have provided evidence for the significance of facilitation, namely the role of lecturers, in the process of engagement. To promote active engagement in the programming course, the researcher focused on the learners rather than the level of participation expected from the teachers. The participation of students and their degree of contentment with the learning process are vital, irrespective of the educational methods utilized. Students' degree of involvement refers to the extent of time and effort they invest in actively participating in the learning process (Bond, 2020; Khlaif et al., 2021). Essentially, it demonstrates the level of student involvement in the classroom. Creating positive and effective learning gains becomes very challenging when students are not actively engaged in the learning environment and express dissatisfaction with it. Prior research conducted by Altun & Serin (2019), Azzi et al. (2020), and Rooney & Nyström (2018) provides compelling evidence that learning style significantly influences educational

institutions. According to Altun and Serin (2019), students perform better and are more engaged when the learning process caters to their chosen learning methods. The design of the online learning environment and the performance of learners can be significantly influenced by the learner's learning style, but only in an online learning setting. The relationship between meaningful learning and student involvement in programming is tightly connected.

The findings of this study indicate that students' participation in and active involvement with simulator-based sessions demonstrate their engagement in significant and meaningful learning. The utilization of a simulator in conjunction with interactive and uncomplicated learning activities fostered a dynamic learning environment that facilitated the collective understanding of programming principles. Conversely, providing students with additional chances to ask questions and assigning them more mentally challenging tasks might enhance their motivation and result in more substantial learning. Moreover, the limited number of responses implies that these associations might potentially alter with a larger sample size, and each individual trait may exert a more or less significant influence on students' learning.

## 6.0 CONCLUSION

This study examines the effects of using a machine learning programming simulator compared to not using one on the learning outcomes of university students in a programming course. This program was developed based on the Allessi and Trollip Developmental Model.

A quasi-experiment was conducted to evaluate the impact of utilizing these applications on engagement for learning materials. The ANOVA study revealed that university students who were exposed to the ML programming simulator and those who were exposed to the noML programming simulator exhibited notable differences in their engagement in the programming course. The study's findings indicate that the ML programming simulator, which utilizes machine learning technology, provides more advantages compared to the noML programming simulator, which merely addresses students' issues based on the most recent input. In summary, the study indicates that the ML programming simulator was a superior choice for both groups of university students compared to the absence of an ML programming simulator.

## 7.0 REFERENCES

Altun, H., & Serin, O. (2019). Determination of learning styles and achievements of talented students in the fields of Science and Mathematics. In Cypriot Journal of Educational Sciences (Vol. 14, Issue 1). www.cjes.eu

Angeli, C., & Valanides, N. (2020). Computers in Human Behavior Developing young children's computational thinking with educational robotics: An interaction effect between gender and scaffolding strategy. Computers in Human Behavior, 105(March 2019), 105954. https://doi.org/10.1016/j.chb.2019.03.018

Ashmore, R., Calinescu, R., & Paterson, C. (2021). Assuring the Machine Learning Lifecycle: Desiderata, Methods, and Challenges. ACM Computing Surveys, 54(5).

Azzi, I., Jeghal, A., Radouane, A., Yahyaouy, A., & Tairi, H. (2020). A robust classification to predict learning styles in adaptive E-learning systems. Education and Information Technologies, 25(1), 437–448. https://doi.org/10.1007/s10639-019-09956-6

Badan Pengkajian dan Penerapan Teknologi. (2023). Kebutuhan Programmer Indonesia.

Baist, A., & Pamungkas, A. S. (2017). Analysis of Student Difficulties in Computer Programming. VOLT: Jurnal Ilmiah Pendidikan Teknik Elektro, 2(2), 81. https://doi.org/10.30870/volt.v2i2.2211

Bond, M. (2020). Facilitating student engagement through the flipped learning approach in K-12: A systematic review. Computers and Education, 151. https://doi.org/10.1016/j.compedu.2020.103819

Budi, A. H. S., Juanda, E. A., Fauzi, D. L. N., Henny, H., & Masek, A. (2021). Implementation of simulation software on vocational high school students in programming and arduino microcontroller subject. Journal of Technical Education and Training, 13(3), 108–114. https://doi.org/10.30880/jtet.2021.13.03.010

Edwards, J., Hart, K., & Warren, C. (2020). A Practical Model of Student Engagement While Programming. SIGCSE 2020 - Proceedings of the 51st ACM Technical Symposium on Computer Science Education, 413–419. https://doi.org/10.1145/3328778.3366863

Harimurti, R., Ekohariadi, Munoto, & Asto Buditjahjanto, I. G. P. (2021). Integrating k-means clustering into automatic programming assessment tool for student performance analysis. Indonesian Journal of Electrical Engineering and Computer Science, 22(3), 1389–1395. https://doi.org/10.11591/ijeecs.v22.i3.pp1389-1395

Jamil, M. G., & Isiaq, S. O. (2019). Teaching technology with technology: approaches to bridging learning and teaching gaps in simulation-based programming education. International Journal of Educational Technology in Higher Education, 16(1). https://doi.org/10.1186/s41239-019-0159-9

Khlaif, Z. N., Salha, S., & Kouraichi, B. (2021). Emergency remote learning during COVID-19 crisis: Students' engagement. In Education and Information Technologies (Vol. 26, Issue 6, pp. 7033–7055). Springer. https://doi.org/10.1007/s10639-021-10566-4

Medvediev, M. (2019). The use of E-olymp internet portal in programming competitions. Olympiads Inf, 13, 201-208.

Rooney, D., & Nyström, S. (2018). Simulation: A complex pedagogical space. In Australasian Journal of Educational Technology (Issue 6).

Zinovieva, I. S., Artemchuk, V. O., Iatsyshyn, A. V., Popov, O. O., Kovach, V. O., Iatsyshyn, A. V., & Radchenko, O. V. (2021, March). The use of online coding platforms as additional distance tools in programming education. In Journal of physics: Conference series (Vol. 1840, No. 1, p. 012029). IOP Publishing.