Leveraging Data Analytics to Investigate the Effectiveness of Flipped Classroom Models: A Case Study of Practical Programming Teaching

Huy Tran[1,2], Tien Vu-Van[1,2], Thanh-Van Le[1,2], Hoang-Anh Pham[1,2] and Nguyen Huynh-Tuong[3]

[1]Ho Chi Minh City University of Technology (HCMUT), Vietnam
[2]Vietnam National University Ho Chi Minh City (VNU-HCM), Vietnam
[3]Industrial University of Ho Chi Minh City (IUH), Vietnam

anhpham@hcmut.edu.vn (Corresponding Author)

https://doi.org/10.34190/ejel.22.9.3597

Abstract: Educators face multiple challenges when teaching programming, such as the intricate nature of programming knowledge, the choice of effective teaching methods, and the diverse abilities of learners. Traditional teaching methods often fail to address these challenges, leading to higher dropout rates and lower student grades. This paper explores a study on the effectiveness of the flipped classroom model as a strategy to enhance student engagement in a practical programming course. In addition, learning data was analyzed to examine the relationship between pre-class preparation and in-class learning outcomes. The study's results indicate that the flipped classroom model significantly enhances student engagement and performance. Students who diligently completed pre-class assignments and dedicated more time to study demonstrated improved performance in subsequent in-class exercises. The results emphasize the potential of the flipped classroom model as a successful teaching method for increasing student involvement, encouraging self-directed learning, and ultimately improving the overall educational experience in programming courses.

Keywords: Flipped classroom, Practical programming course, Learning analytics, Correlation analysis, Naive Bayes classification

1. Introduction

Teaching programming is increasingly important in today's technology-driven world, where programming skills are highly valued across various industries. However, programming is a challenging task (Sobral, 2021). Teaching programming is to develop higher-order thinking skills, including logical reasoning, algorithmic problem-solving, and computational thinking (Fessakis, Gouli and Mavroudi, 2013; Kalelioğlu, 2015). After receiving theoretical instruction in the classroom, students are encouraged to participate in practical programming sessions to familiarize themselves with programming. During these hands-on sessions, students engage in various fundamental exercises to strengthen their understanding of programming concepts and enhance their problem-solving abilities.

In this study, we implemented hands-on coding exercises using Moodle, an open-source learning management system (LMS) (Gamage, Ayres and Behrend, 2022) that allows educational institutions to deliver courses and manage online learning activities. By utilizing Moodle, instructors can create online courses to complement their offline teaching by posting study materials, pre-recorded videos, quizzes, and more. The quizzes implemented on the Moodle system mainly include multiple-choice, short-answer, and mapping questions. To provide a programming learning environment emphasizing hands-on coding, we employed the CodeRunner plugin (Lobb and Harlow, 2016) to deploy programming exercises within the Moodle system.

In previous semesters, the teaching approach involved assigning weekly exercises for students to practice both in class and at home. During classroom sessions, students would work on designated exercises under the instructor's guidance. Any unfinished exercises would be assigned as homework for further practice. However, this method posed challenges as students often lacked preparation, and it was difficult for the teacher to ensure consistent quality for all students. There were instances where the teacher had to re-teach the theory throughout the entire session before students could attempt the exercises. The excessive review of theory diminished the time available for hands-on practice, rendering the practical teaching process less effective. Implementing the flipped classroom model creates an additional learning environment that allows students to allocate more time to the subject while gathering valuable insights from their preparation at home. This approach enhances the efficiency of classroom teaching and learning.

Cite this article: Tran, H. et al. 2024. "Leveraging Data Analytics to Investigate the Effectiveness of Flipped Classroom Models: A Case Study of Practical Programming Teaching", *Electronic Journal of e-Learning*, 22(9), pp 144-162, https://doi.org/10.34190/ejel.22.9.3597

This study presents a new method for teaching practical programming. The proposed method is based on the fundamental idea of the flipped classroom teaching model, in which students study new content independently outside of class and use class time to engage in interactive activities such as discussions, problem-solving, and hands-on exercises (Abeysekera and Dawson, 2014). The flipped classroom is a reversed delivery model where typical lecture and homework elements are converted (Davis, 2016). Before class, students must review guidelines, watch short video lectures, complete quizzes, and participate in forum discussions (Han and Klein, 2019). Meanwhile, the class time is dedicated to exercises, projects, in-depth inquiries about the pre-watched lectures, and engaging in hands-on activities (Liu, Wang and Izadpanah, 2023). At the same time, the instructor assesses the learners' application of knowledge.

In the proposed experimental model, we introduced pre-class materials, including online resources and basic hands-on exercises, for learners to complete before attending the practical sessions. Learners were additionally assigned more difficult and complex exercises during the in-class practical sessions than at home. The outcomes obtained from this experiment were then used to evaluate the feasibility and effectiveness of the proposed teaching model.

This research examines explicitly two main areas: (1) the influence of pre-class preparation on students' preparedness and preparation in class and (2) the relationship between pre-class and in-class learning activities, utilizing learning analytics to gain a more comprehensive understanding of student engagement and performance patterns.

The subsequent sections of this paper are structured in the following manner. Section 2 summarizes related works to clarify our study's framework. Then, Section 3 describes the proposed design and implementation. The effectiveness of flipped classroom implementation in programming courses is discussed in Section 4. Section 5 presents an additional learning analytics approach that examines the relationship between pre-class and in-class activities. This approach enhances the effectiveness of the flipped classroom model in improving student performance. Sections 6 and 7 conclude our study by presenting limitations, remarks, and future directions.

2. Related Works

2.1 Challenges and Innovations in Teaching Programming

Teaching programming presents many challenges for educators, encompassing the nature and complexity of programming knowledge, the wide selection of pedagogical methods, and the heterogeneity of learner abilities (Kadar, et al., 2022). Core concepts such as iteration, language-specific constructs, and program design are abstract, making them difficult for students to grasp (Rouhani, et al., 2022). Moreover, programming technologies' dynamic and ever-evolving nature requires educators to continuously update their knowledge and instructional materials (Caviativa, et al., 2022). For instance, in web programming courses, the rapid pace of technological change and the diversity of tools and topics demand that educators frequently revise their course content and adapt their teaching methods (López-Pimentel, et al., 2021). The challenges are further complicated due to the need to address technical skills, such as algorithmic logic and programming language syntax, and non-technical skills, such as problem-solving and teamwork (Santos, et al., 2020). The diversity of students, ranging from those with no programming experience to those with extensive development backgrounds, exacerbates the difficulty of tailoring instruction to meet all learners' needs. This issue is particularly pronounced in computer science and data analytics programs, where students' varying levels of prior experience lead to disparate performance outcomes in assignments and overall course success (Ahmaderaghi, et al., 2024).

As a result, conventional teaching approaches frequently need to effectively tackle these difficulties, leading to many students dropping out and performing poorly, particularly among non-computer science students who find programming challenging (Groher, et al., 2021). The failure rates in introductory programming courses can be remarkably high, occasionally reaching as high as 60%, highlighting students' challenges in acquiring essential skills (Shahamiri, 2019). Moreover, traditional lecture-oriented teaching approaches frequently need to be more effective in capturing students' attention, resulting in diminished motivation and participation, especially in programming courses. This passive learning mode does not foster critical thinking or active involvement, essential for mastering programming (Dietrich and Evans, 2022). Research suggests that conventional lectures effectively capture the attention of only approximately 65% of students, resulting in many students needing to be more engaged (Shah, et al., 2024).

In order to tackle these problems, novel pedagogical approaches, such as problem-based learning and the incorporation of programming challenges, have demonstrated the potential to enhance student engagement and improve learning results by fostering a collaborative and competitive atmosphere (Santos, et al., 2020;

Martins, et al., 2020). However, these methods require careful implementation and continuous assessment to ensure effectiveness (Santos, et al., 2020). One practical approach is gamification, which has been shown to significantly enhance student motivation and engagement (Papadakis and Kalogiannakis, 2019). By integrating game design elements such as points, badges, avatars, and leaderboards, educators can create a dynamic and competitive learning environment that encourages active participation and sustained interest in programming courses (Papadakis, 2020). In another research, web-based applications combining automated programming assessment with gamification concepts can also provide immediate and accurate feedback, enhancing students' willingness to participate and learn from their mistakes (Hellín, et al., 2022).

To effectively teach programming, adopting a holistic approach that considers students' varying needs and the ever-changing nature of the field is necessary to incorporate innovative teaching methods to establish a stimulating and successful learning environment.

2.2 Flipped Classroom: Application in Teaching Programming

The flipped classroom model, which reverses traditional teaching by delivering instructional content outside the classroom and focusing on interactive activities during class, has gained popularity in teaching programming. This approach enhances student learning and participation via in-class activities like problem-solving, discussions, and collaborative projects. By shifting the focus from passive information reception to active knowledge application, the flipped classroom allows students to access lecture materials, such as videos or readings, before class, freeing up classroom time for more personalized and interactive learning experiences (Rivera and Flores, 2024). This method promotes a deeper understanding of the material as students arrive prepared to engage in activities that reinforce their learning.

The flipped classroom model is particularly beneficial for accommodating diverse learning styles. It allows students to learn at their own pace outside of class and review materials as needed to ensure comprehension (Sánchez-Cuervo, 2024). Technology integration is essential in this approach, providing digital tools and platforms that deliver pre-class content and facilitate in-class activities.

In programming education, the flipped classroom has been shown to enhance student engagement, understanding, and the application of complex concepts. Research indicates that this model can significantly improve students' problem-solving skills and ability to apply theoretical knowledge in practical scenarios (Zhou, 2024). The flipped classroom fosters active learning and computational thinking- skills crucial for mastering programming languages and concepts- by allowing students to engage with lecture materials at their own pace, followed by in-class activities focused on coding exercises and collaborative projects.

Despite its benefits, the implementation of the flipped classroom model presents challenges. The success of this approach heavily depends on the quality of pre-class materials and the design of in-class activities. Poorly designed materials can lead to confusion and disengagement among students, diminishing the advantages of the flipped model (Malkoc, et al., 2024). Additionally, effective implementation requires reliable access to digital resources and technological support, which can be a barrier in some educational settings. Another challenge is the need for students to be self-motivated and disciplined in engaging with pre-class materials (Jin-gang, et al., 2024). Without this intrinsic motivation, students may come to class unprepared, hindering the effectiveness of in-class activities.

Nevertheless, the flipped classroom model has shown promise in programming education, often leading to higher levels of student satisfaction and engagement, which can result in better learning outcomes and retention of programming skills. The model also fosters a collaborative learning environment, encouraging peer-to-peer learning and support, which are beneficial in programming education, where problem-solving usually involves teamwork (Kraml, 2024). Thus, despite the challenges associated with the flipped classroom model, its application in teaching programming has generally yielded positive results, providing a more interactive, student-centered learning experience. The effectiveness of this method relies on careful and thorough planning and implementation, guaranteeing that the components before and during class are carefully crafted and smoothly incorporated. This research aims to apply methods to improve participation in pre-class activities and assess the efficacy of the flipped classroom approach in teaching practical programming courses.

2.3 Learning Analytics

Educational Data Mining (EDM) and Learning Analytics (LA) are rapidly evolving fields that leverage data-driven approaches to enhance educational outcomes. These disciplines analyze educational data to improve learning processes, personalize education, and optimize educational systems.

Educational Data Mining involves applying data mining techniques to educational data to discover patterns and insights that inform educational practices. Data analysis techniques, such as classification, clustering, and association rule mining, are used to examine data from educational environments like learning management systems and online courses. For instance, Pliuskuvienė et al. (2024) highlight using EDM to identify at-risk students and tailor interventions to improve their academic performance. Similarly, Bellaj et al. examine the use of EDM to predict student success and improve curriculum design by analyzing student interaction data (Bellaj, et al., 2024).

Learning Analytics, on the other hand, focuses on measuring, collecting, analyzing, and reporting data about learners and their contexts to comprehend and enhance the learning process and the settings in which it takes place. Cerezo et al. (2024) emphasize the role of LA in providing real-time feedback to students and educators, thereby facilitating adaptive learning environments that respond to the specific needs of each learner. Liu et al. (2023) support this approach, stating that LA can assist educators in making informed decisions by visualizing learning patterns and identifying areas for improvement.

In a study focused on programming students, various techniques, including k-nearest neighbors (kNN), decision trees, logistic regression, and neural networks, were used to predict students likely to drop out. Among these techniques, kNN achieved the highest accuracy in classifying at-risk students (Pliuskuvienė, et al., 2024). Furthermore, various machine learning algorithms such as Naïve Bayes, support vector machine, random forest, and extreme gradient boost have been employed to forecast academic accomplishments. The primary objective has been to enhance data quality by eliminating noise and fine-tuning hyperparameters (Bellaj, et al., 2024). Predictive models are essential for early intervention, enabling educators to offer timely assistance to students who may be encountering difficulties.

Figure 1 illustrates a comprehensive framework that integrates innovative teaching methods and learning analytics approaches to address the intricate challenges of teaching programming. The framework delineates three fundamental obstacles in programming education: the complex nature of programming knowledge, limited student engagement, and the heterogeneity of student skills. Traditional teaching methods frequently need help effectively tackling these problems, resulting in a higher percentage of students dropping out and subpar academic performance, particularly among students with limited or no previous programming knowledge. This research highlights the significance of implementing various instructional methods to address the above-mentioned difficulties, including gamification, flipped classrooms, and problem-based learning. These methods promote an interactive and cooperative learning setting while supporting multiple learning preferences and skill levels.

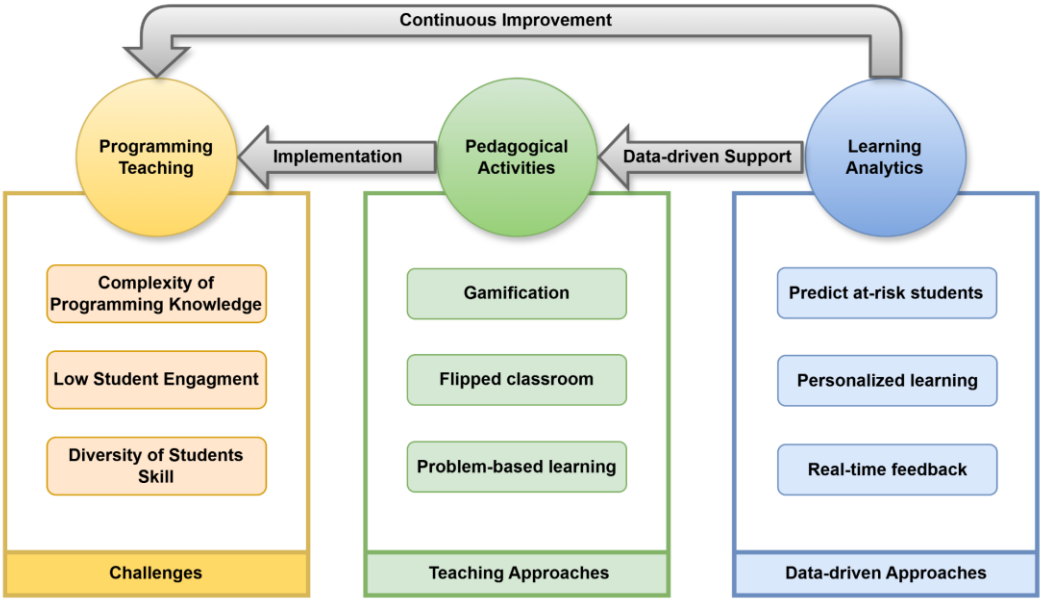

Figure 1: A framework integrating teaching methods and learning analytics for programming education

In addition, the proposed framework incorporates LA and EDM as essential elements to improve and customize the learning experience. Through the utilization of data-driven methodologies, such as predictive modeling for the identification of students at risk, personalized learning pathways, and real-time feedback mechanisms,

educators can make well-informed choices that directly influence student performance and engagement. Moreover, the support offered by learning analytics facilitates the ongoing enhancement of instructional approaches and materials, thereby enhancing the accessibility of programming education despite the ever-changing nature of programming technologies. The integrated framework promotes a continuous improvement culture in programming education, intending to improve student learning outcomes and reduce the achievement gap between students with varying prior experience and aptitude levels.

3. Methodology

3.1 Implementation of the Flipped Classroom Model

To suggest adopting the flipped classroom model in practical programming courses, we analyzed various teaching models, such as the traditional, general, and flipped classroom models, specifically for practical programming courses. This examination allowed us to identify the benefits and drawbacks associated with each approach. Table 1 provides a concise overview of the comparisons, explicitly highlighting the support provided to learners based on two specific criteria: student engagement and student self-study.

Table 1: Comparison of the traditional classroom, general flipped classroom, and flipped classroom for a practical programming course

Model vs. Criteria	Student Engagement	Student Self-Study
Traditional Classroom	May face challenges in maintaining student engagement throughout the class (Singh, et al., 2024).	Relatively low self-study as students mostly follow instructor-led activities (Estaji and Jonaidi-Jafari, 2022).
General Flipped Classroom	Besides in-class activities requiring active participation, student engagement increases through pre-class materials (White, et al., 2017).	Cultivate student self-regulation by requiring pre-class preparation and independent practice (Silverajah, et al., 2022).
Flipped Classroom for a Practical Programming Course	Encourage active student participation through hands-on coding questions and real-world projects in pre-class activities.	Develop intense student self-study and self-directed learning skills through independent coding practice and project work.

Traditional programming classes typically emphasize in-class instruction, where students receive direct guidance and knowledge from the instructor. Although this approach may accelerate the learning process, it also possesses significant limitations. For example, students who need a complete comprehension of the subject matter during class may encounter difficulties with their homework, and only those who have a high level of discipline tend to make adequate preparations for the next session. This model frequently emphasizes delivering content rather than addressing individual learners' unique abilities and learning styles (Li, et al., 2014). As a result, instructors need help keeping students actively involved in the class (Singh, et al., 2024), and more emphasis on self-directed learning leads to a low level of independent study as students mainly rely on instructor-led activities (Estaji and Jonaidi-Jafari, 2022).

On the other hand, the Flipped Classroom model, especially for practical programming courses, provides a more dynamic approach by organizing learning into two primary elements: pre-class and in-class activities. Pre-class activities transfer the obligation of learning to the students, encouraging active participation and enabling them to learn at their preferred pace. To promote independent learning and motivation, we suggest implementing pre-class assignments that gradually escalate in complexity, commencing with easy assignments and advancing to more challenging ones. Engaging in these tasks promotes student readiness before class, which is essential for enhancing the efficacy of the learning process. Programming courses often incorporate pre-class activities that involve practical coding exercises and real-world projects. These activities improve problem-solving abilities and foster a strong sense of self-study and self-directed learning by encouraging independent coding practice and project work.

In-class activities in the Flipped Classroom emphasize interactive and collaborative exercises, such as team-based learning, presentations, and quizzes, which build on the knowledge students have acquired during their pre-class preparation. We recommend dividing in-class sessions into two phases for programming education: an initial review of pre-class work, including live coding and error correction, followed by hands-on practice with more advanced exercises. The instructor should guide these sessions through active learning methods like pair programming and group discussions, making the class livelier as each student brings different perspectives and

levels of understanding (White, et al., 2017). Moreover, the Flipped Classroom cultivates student self-regulation by requiring pre-class preparation and independent practice. It encourages learners to proactively search for additional learning resources that align with their preferences and styles (Silverajah, et al., 2022).

Therefore, we propose implementing the flipped classroom model for practical programming courses, highlighting its effectiveness based on two criteria: student engagement and self-study. In helpful programming courses, student engagement is demonstrated through participation in hands-on coding exercises and real-world projects (Pears, 2010). At the same time, self-study is reflected in the student's ability to develop their skills independently. These criteria, such as participation rates and self-development, can be quantitatively assessed using data collected from Moodle.

This study integrates the Flipped Classroom model into practical programming involving two exercises: PreLab and InLab.

- **PreLab**: Before the in-class session, students must complete the exercises in the PreLab. The PreLab exercises typically consist of easy-to-moderate difficulty-level tasks, aiming to help students grasp the key concepts from the theory and facilitate self-learning.
- **InLab**: The in-class activities consist of two components. First, the PreLab exercises will be discussed and reviewed. The instructor will gather feedback from students regarding the PreLab tasks and provide general guidance to the entire class. Then, students will practice individually with the InLab exercises under the instructor's guidance. The InLab exercises will be organized from moderate to advanced difficulty and aim to enhance students' programming skills after they have been sharpened through the PreLab exercises.

Integrating PreLab and InLab activities improves student engagement and promotes more significant interaction between teachers and students. Instructors can better understand each student's capabilities by utilizing LA approaches, such as real-time predictive analytics. This implementation enables customized instructional approaches that more effectively cater to individual needs, ensuring that the pre-class and in-class components are well-aligned with learners' current comprehension and advancement. Providing individualized assistance can significantly enhance academic achievements in programming instruction.

3.2 Study Design

This study aims to incorporate the Flipped Classroom model into Programming Fundamentals courses to increase student engagement and use the learning data obtained from activities conducted before and during class. The Flipped Classroom approach positively impacts student learning engagement, particularly in pre-class activities. The influence of pre-class exercises on student performance is demonstrated by the improved academic outcomes of students who engage in active participation as opposed to those who do not. The gathered educational data can also be employed to analyze and improve instructional methodologies. A learning analytics methodology is utilized to forecast the in-class scores of students by analyzing their pre-class outcomes, thereby providing additional evidence of the efficacy of the Flipped Classroom model.

The study was conducted in the Programming Fundamentals course during the second semester of the 2021-2022 academic year. A question repository consisting of 99 questions was created to enhance the teaching of practical programming in the course. These questions cover fundamental programming topics such as Strings, Arrays, Functions, and Pointers and are distributed across four practice sessions. Using a voting process, the instructors responsible for the practical sessions categorized the questions into three difficulty levels—easy, medium, and hard. Questions rated as easy to medium were compiled into a set for Prelab activities, while those rated as medium to hard were used for Inlab exercises. The questions were delivered in an online judge format, requiring students to solve programming problems related to the course topics. Student submissions were automatically graded using test cases in CodeRunner, a plugin implemented on the Moodle LMS platform.

Figure 2 illustrates the process of the proposed approach to a programming course, divided into two phases: PreLab and InLab. In the PreLab phase, students begin by engaging with preparatory materials, including reading material like slides or books, videos, and code examples. The work on exercises classified as Easy and Medium in difficulty, with their submission, is graded by an Automated Grading System (AGS). Student's results are recorded in a system for later review by the lecturer. On the lecturers' side, they are responsible for selecting and categorizing exercises by difficulty level, utilizing Learning Analytic tools to assist in this process, like classification tools. Lecturers also review the student's results to gain insights into the student's understanding and performance.

In the InLab phase, students start by listening to a general provided by the lecturer before moving on to more challenging Medium and Hard difficulty exercises. AGS grades these submissions for exercises. On the lecturers' side, they can use Learning Analytic tools to predict the InLab scores right after the PreLab exercises closed. The prediction results help lecturers understand the distribution of scores and the student's overall ability. This analysis enhances the interaction between lecturers and students, allowing the lecturer to provide a more tailored and effective guide during the InLab session.

According to Figure 2, the evaluation needs to be conducted from two perspectives: Pedagogical Activities and Learning Analytics. From the Pedagogical Activities perspective, the evaluation can be based on the feasibility of implementing the teaching approach, the readiness of students to participate in the PreLab or the impact of the PreLab on the course as reflected in the scores. From the Learning Analytics perspective, the evaluation can be based on the prediction's feasibility or accuracy in students' exercise data.

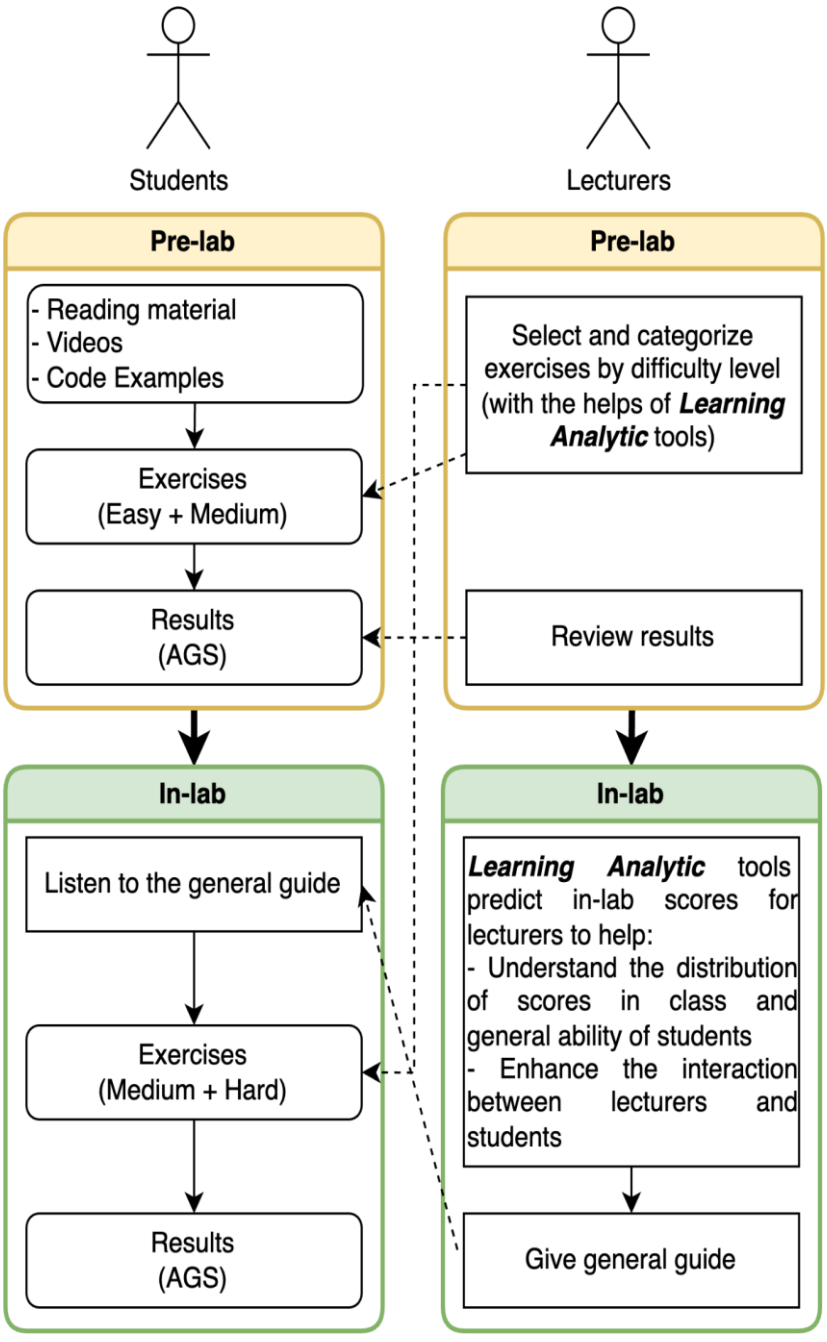

Figure 2: The process of the proposed approach for applying flipped classroom into a programming course

3.3 Data Accumulation

The proposed model was applied in the Programming Fundamentals course in the second semester of the academic year 2022-2023 at our university. The course includes four (4) PreLab and four (4) InLab lessons. This research was conducted with 786 first-year undergraduate students who participated in the course. The data collected from the Moodle system consists of attempts made by students. Based on the Moodle design, a lab exercise contains multiple questions, and each question can have multiple attempts from students. In each attempt, students submit and try numerous times until they achieve the desired result. As a result, in a lab exercise, the minimum number of attempts students must carry out to complete the lab is equal to the number of questions. For example, if PreLab 1 has ten questions, the minimum number of attempts required for students to complete the lab is 10. If the number of attempts exceeds 10, students have submitted multiple times for some questions in the lab. Students can make many attempts for each question, so the scoring for a question is based on the highest-scoring attempt. Therefore, the score of a lab exercise for each student is the average score of the questions in the lab. If a question does not have any attempts by a student, that question will be scored as 0 points.

The results of the lab exercises are compiled and presented in Table 2. Regarding the number of students participating in the practical exercises, out of a total of 786 students, PreLab 1 had the highest student interactions, followed by InLab 1 with 757 (96.43%) and 746 (95.03%) students, respectively. On the other hand, PreLab 4 had the lowest number of participations (688 students - accounting for 87.64%), followed by PreLab 2. It can also be seen that PreLab 2 achieved the highest mean score (9.63), while the lowest mean score went to InLab 1 (8.19).

Table 2: Summary of lab results

	PreLab 1	InLab 1	PreLab 2	InLab 2	PreLab 3	InLab 3	PreLab 4	InLab 4
Count	757	746	688	732	708	733	676	706
Mean	8.97	8.19	9.63	9.12	9.07	9.07	9.35	9.15
Std	2.34	2.87	1.38	2.09	2.21	2.18	1.93	2.13
Min	0	0	0	0	0	0	0	0
25%	9.40	7.77	10.0	9.75	9.60	9.58	10.0	9.88
50%	9.90	9.70	10.0	10.0	10.0	10.0	10.0	10.0
75%	10.0	10.0	10.0	10.0	10.0	10.0	10.0	10.0
Max	10.0	10.0	10.0	10.0	10.0	10.0	10.0	10.0

Additionally, Figure 3 shows each lab's score distribution, revealing a trend that students either achieve perfect scores or abandon the assignments altogether. Scores are heavily concentrated at the 10-point mark, with a significant number also clustered at 0. The remaining score categories are practically empty. This suggests a binary approach to these practical programming assignments – students strive for the highest marks or give up. Interestingly, despite not following a standard or uniform distribution, our data provides valuable insight into how students typically perform on these tasks.

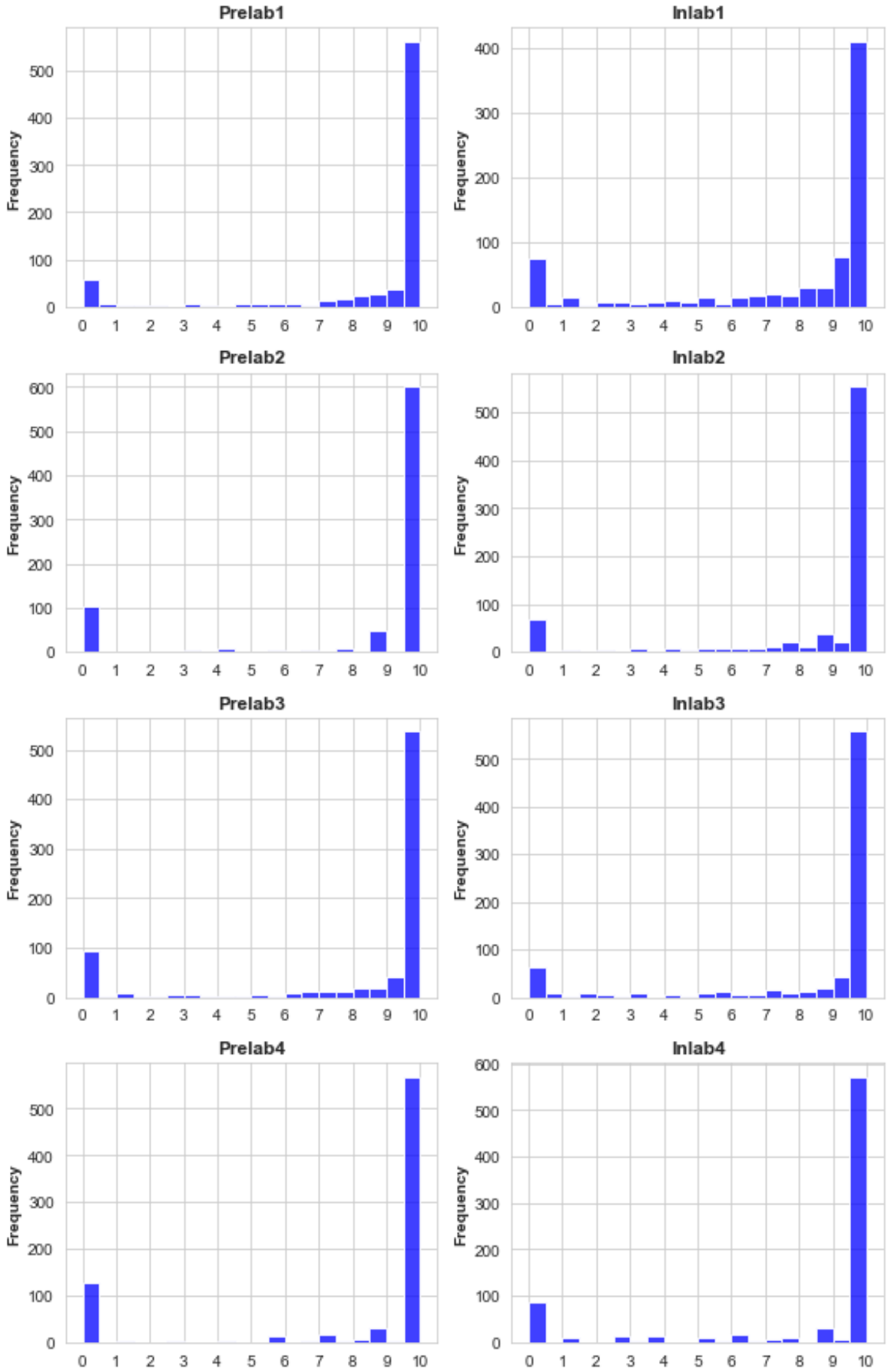

Figure 3: The lab score distribution

4. Effectiveness of the Flipped Classroom Model Implementation

4.1 Student Consensus

Consensus among most students is demonstrated by the ratio of students participating in PreLab questions to the total number of students in the course. Students only need to answer one PreLab question to be considered participating. The value of this ratio aims at 1.0, indicating a high consensus on the new learning method.

In other words, this consensus indicates the learners' engagement. The number of participants partially reflects the effectiveness of implementing this teaching approach. The experimental results obtained for PreLabs 1, 2, 3, and 4 are 0.96, 0.87, 0.90, and 0.86, respectively, which indicates that students still widely accepted the new teaching method even though it was implemented for the first time.

4.2 Ability to Measure Learners' Studying Time

Measuring the study duration of learners cannot be accurately recorded through the system. The actual study duration will be greater than the time recorded in the system. The latter is calculated from the first submission to the final submission. The average study duration t_j for assignment j and the average study duration r for a lab are determined as follows:

$$t_j = \frac{1}{n}\sum_{i=1}^{n} d_i$$

$$r = \frac{1}{T}\sum_{j=1}^{T} t_j$$

where n is the number of attempts of a student for assignment j; $d_1, d_2, \dots, d_j, \dots, d_n$ are the durations of the student's attempts for assignment j; and T is the allowed study duration for a lab.

If T is constant, a higher study duration recorded in the system reflects a higher level of learner investment. This indicates that the student spends more time and effort on the assignments or labs. Table 3 summarizes the study duration that students must allocate for each lab assignment, where the total allowed study duration is the same for all PreLabs and all InLabs.

Table 3: Student's time spent for each lab lesson

	Related PreLab	Related InLab
Lab 1	7 hours	8 hours
Lab 2	3 hours	4 hours
Lab 3	7 hours	5 hours
Lab 4	4 hours	4 hours

4.3 Ability to Collect Illustrative Data for Learner Development

This ability is related to student self-study. Using appropriate statistical data analysis tools, we can observe the development of learners over time. By examining the number of submissions for each question, comparing related questions (which may have different levels of difficulty and occur at different stages: PreLab or InLab), and using a timeline diagram, we can visualize the development of learners. Figure 4 illustrates the growth process of solving a question in an assignment for a sample student.

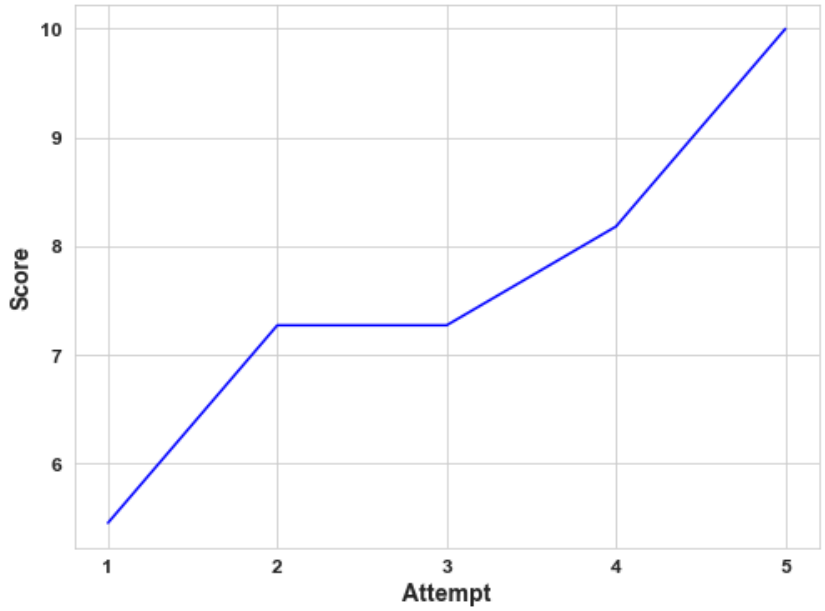

Figure 4: The growth process of a learner solving a question

We can also observe the growth index, which is the difference between the best and initial performance. Figure 5 illustrates the distribution of this growth index in a sample assignment question. The case with a value of 0 represents the group of students whose highest score corresponds to their first submission, excluding those who submitted only once. The case with a value of 10 represents the group of students who initially scored 0 but completed the question with a score of 10.

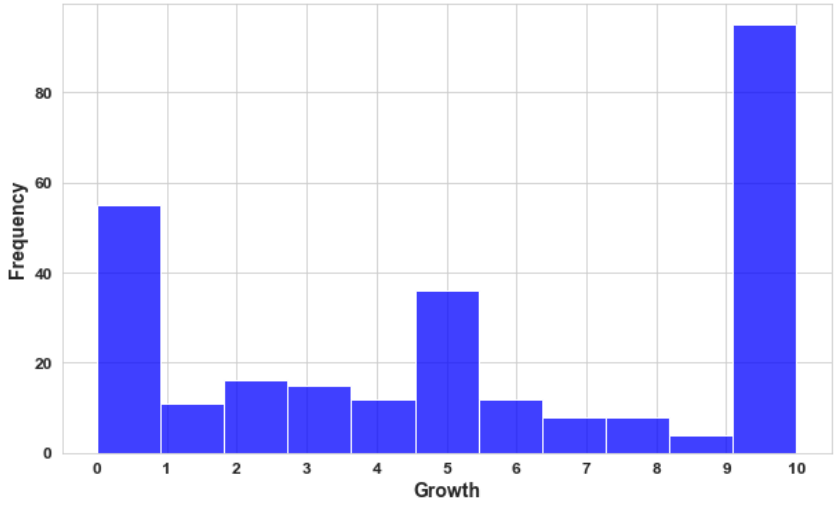

Figure 5: The histogram of growth index of a question

4.4 The Impact of Completing PreLab Before Doing InLab

We compared two groups of students based on the score distribution of their InLab assignments: (1) the first group includes students who did not complete PreLab, and (2) the second group includes students who completed PreLab. Figure 6 shows the score distributions of the InLab assignments for the two groups, in which the label of **InLab i (without PreLab i)** indicates that the students did not complete PreLab i before doing InLab i, and **InLab i (with PreLab i)** means that students completed PreLab i before doing InLab i. For example, if PreLab 1 consists of 10 questions, some students complete only one question, while others complete nine. We set a threshold to determine the completion of PreLab, where students have to complete all the questions in the PreLab assignment. With this threshold, the percentage of students in each group for each lab assignment is summarized in Table 4.

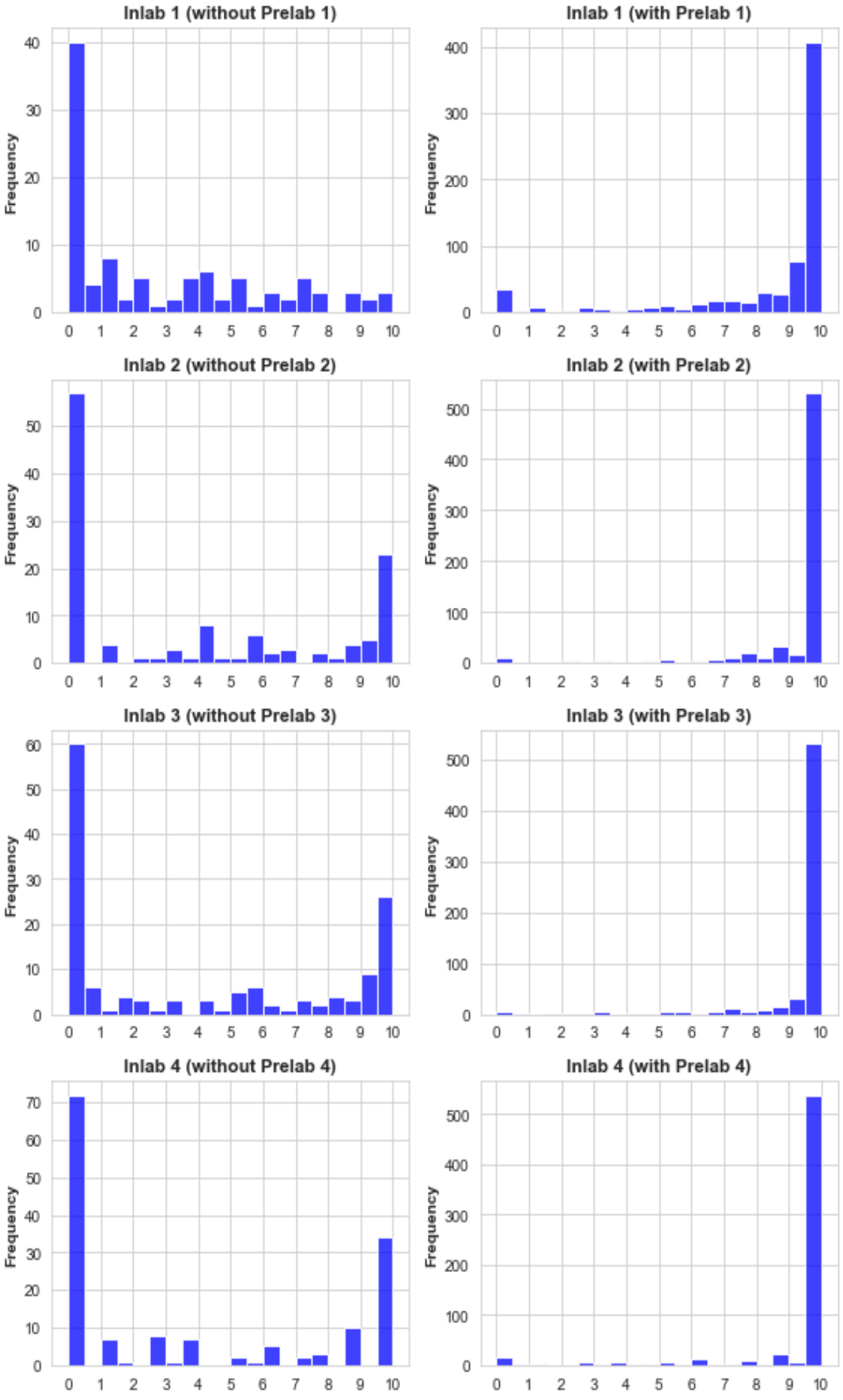

Figure 6: The score distribution of InLab assignments

Table 4: Percentage of students in each group completing each lab assignment

	% Completed	% Not Completed
PreLab 1	87%	13%
PreLab 2	84%	16%
PreLab 3	82%	18%
PreLab 4	81%	19%

As shown in Figure 6, the distributions on the left side have more students with lower scores than those on the right, which indicates a higher probability of failure for the students who did not complete PreLab. Additionally, more than 500 students achieved scores between 9.5 and 10, while the number of students with scores below 6 is low. Therefore, completing PreLab has a positive impact on the InLab results. Additionally, the score distribution of InLab assignments for students who did not complete PreLab shows instability in their performance, which is reflected in the scattered distribution of InLab scores ranging from 0 to 10 for students who did not complete PreLab. In most PreLab assignments, a significant portion of the histogram for InLab 1 (without PreLab 1) shows a score of 0.

To obtain more detailed information, as shown in Table 5, we examine the evaluation of three groups (the group of students who completed PreLab, the group of students who partially completed PreLab, and the group of students who did not do PreLab) in three cases of InLab results ("< 5", "≥ 5", and "≥ 9").

Table 5: InLab results in details

		% Students completed PreLab	% Students partially completed PreLab	% Students did not do PreLab
InLab 1	< 5	10.23 %	73.53 %	89.66 %
	≥ 5	89.77 %	26.47 %	10.34 %
	≥ 9	70.76 %	4.90%	3.45 %
InLab 2	< 5	3.92 %	61.79 %	68.37 %
	≥ 5	96.08 %	38.21 %	31.63 %
	≥ 9	82.65 %	22.76 %	21.43 %
InLab 3	< 5	3.42 %	57.34 %	75.64 %
	≥ 5	96.58 %	42.66 %	24.36 %
	≥ 9	87.71 %	24.48 %	12.82 %
InLab 4	< 5	5.21 %	62.75 %	72.73 %
	≥ 5	94.79 %	37.25 %	27.27 %
	≥ 9	85.94 %	22.22 %	17.27 %

Evaluation of groups with InLab results (≥ 5): For students who fully completed the PreLab assignments, the majority (over 89%) of students achieved a passing score (≥ 5) in all 4 InLabs, and the majority of students (over 70%) scored 9 or 10 in all 4 InLabs.

Regarding students who partially completed the PreLab assignments, only about 26.47% to 42.26% of students achieved a passing score (≥ 5) in all 4 InLabs, and only about 4.90% to 24.48% of students scored 9 or 10 in all 4 InLabs. For students who did not do the PreLab assignments, the percentage of students who achieved a passing score (≥ 5) in all 4 InLabs is low (ranging from 10.34% to 31.63%), and similarly, the percentage of students who scored 9 or 10 in all 4 InLabs ranges from 3.45% to 21.43%.

Evaluation of groups with InLab results (< 5): For students who fully completed the PreLab assignments, very few (less than 10%) students received a failing score (< 5) in all 4 InLabs. Regarding students who partially completed the PreLab assignments, over half (over 57%) of students received a failing score (< 5) in all 4 InLabs. For students who did not do the PreLab assignments, the percentage of students who failed the InLab assignments accounted for over 68.37% in all 4 InLabs.

In conclusion, the data analysis reveals the significant impact of completing PreLab exercises on students' performance in the InLab exercises. Students who fully completed PreLab demonstrated consistently higher scores in the InLab tests, while those who did not complete PreLab showed unstable and lower scores. These findings highlight the importance of engaging in PreLab exercises as a crucial step in achieving better results and improving practical skills in the course.

5. Effectiveness of Student's Performance Prediction

After using statistical methods to observe the flipped classroom's effectiveness, we continued to analyze the collected data to clarify the relationship between PreLab and InLab. Specifically, two data analysis methods—correlation coefficient measurement and Naive Bayes classification—were used in this study. Since PreLab is conducted before InLab, we will predict the outcomes of InLab based on the results of PreLab.

The process of students completing assignments during a semester is recorded in a data matrix. The columns of the data matrix represent the following fields as shown in Table 6, including **PreLab Result** (PreLab Score), **PreLab Attempts** (the number of attempts a student submitted answers for this PreLab), **PreLab Questions** (the number of questions in PreLab), **PreLab Growth** (the progress level of a learner calculated by averaging question growths). A question growth is the difference between the max score and the one achieved from the first attempt and the **InLab Result** (InLab Score).

Instead of observing each lab exercise separately, we concatenate the four data matrices for each lab exercise into a single unified data matrix. Then, we adopt the correlation analysis and the predictive effect as below.

5.1 Correlation Analysis

Table 6 displays the Pearson correlation coefficients (Benesty, et al., 2009) that examine the relationships between different factors in PreLab and InLab activities. Significantly, a robust positive correlation was found between the results of the PreLab and InLab activities (r = 0.698), indicating that students who achieved higher scores in the PreLab tasks generally excelled in the subsequent InLab tasks. This relationship can be attributed to several reasons. Firstly, PreLab activities often serve as foundational exercises that equip students with the essential knowledge and skills needed for InLab tasks. Students who actively and comprehensively participate in these pre-class activities will likely develop a more profound comprehension of the concepts, resulting in improved performance in in-class activities (Förster, et al., 2022). Furthermore, according to reference (Liu, et al., 2024), achieving success in pre-class activities can boost students' self-assurance, leading to a more optimistic attitude towards in-class assignments.

Table 6: Correlation analysis between factors of PreLab and InLab

	PreLab Result	PreLab Attempts	PreLab Questions	PreLab Growth	InLab Result
PreLab Result	1	0.739	0.885	0.221	0.698
PreLab Attempts		1	0.876	0.590	0.481
PreLab Questions			1	0.262	0.596
PreLab Growth				1	0.113
InLab Result					1

Furthermore, significant positive correlations were observed between the number of attempts made during the PreLab phase and the results obtained during the InLab phase (r = 0.481). Similarly, significant positive correlations were observed between the number of PreLab questions completed and the results obtained during

the InLab phase (r = 0.596). This suggests that more attempts and PreLab questions are associated with better performance in InLab activities. This positive correlation indicates that active engagement with PreLab materials and dedicating adequate time to learning may lead to better preparation for InLab sessions. Repetitive learning can enhance comprehension, facilitating the students' application of knowledge during the InLab sessions.

On the other hand, the correlation between the growth observed during the PreLab phase and the results obtained during the InLab phase was relatively weak (r = 0.113), suggesting a less substantial relationship. This finding implies that growth measurement should more accurately capture the factors contributing to success in InLab activities. A plausible rationale is that students who excel academically frequently obtain perfect scores on their initial try. In contrast, students with lower performance may choose to completely omit specific questions, leading to a growth value of zero, which does not offer significant insights. This suggests that a more deliberate and strategic approach to designing PreLab exercises is required. This approach should involve presenting more demanding tasks to high-achieving students and offering extra assistance to lower-achieving students. This approach is consistent with the authors' findings in (Malkoc, et al., 2024). The findings indicate that performance in PreLab activities can predict performance in subsequent InLab activities. This highlights the significance of actively participating in PreLab activities to improve learning outcomes.

5.2 Predictive Effects

These findings suggest that PreLab performance can be a predictive factor for subsequent InLab performance, emphasizing the importance of engaging with PreLab activities to enhance learning outcomes. Previous studies have used various methods for prediction. In this study, Naive Bayes Classification is utilized to examine the predictive effect of PreLab results on InLab performance. By employing the Naive Bayes Classification, we can assign categorical predictions to the InLab scores based on the PreLab results. This approach has been proven effective in different domains and has shown promising results in predicting various outcomes (Blanquero, et al., 2021; Rabie, et al., 2022; Farhana, 2021; Vishwakarma and Kesswani, 2023. Naive Bayes Classification was applied to observe the predictive effect of PreLab results on InLab performance as follows:

$$P_{InLab-Result \mid PreLab} = \frac{P_1 \times P_2 \times P_3 \times P_4 \times P_{InLab-Result}}{P_{PreLab}}$$

where

- $P_1 = P_{PreLab-Result \mid InLab-Result}$
- $P_2 = P_{PreLab-Attempts \mid InLab-Result}$
- $P_3 = P_{PreLab-Questions \mid InLab-Result}$
- $P_4 = P_{PreLab-Growth \mid InLab-Result}$

A PreLab is defined as a vector that includes PreLab results, PreLab attempts, PreLab questions, and PreLab growths. Since the predicted outcomes of the Naive Bayes Classifier are categorical, we transformed the InLab scores into five classes: Class 0 (scores from 0 to less than 4), Class 1 (scores from 0 to less than 5), Class 2 (scores from 5.5 to less than 7), Class 3 (scores from 7 to 8.5), and Class 4 (scores 8.5 and above). With a dataset size of 3144, we divided the dataset into two sets: the training set (2512 instances) and the test set (632 instances).

The Naive Bayes Classifier's classification results are presented in Table 7. More precisely, we assessed the model's performance on the test set by measuring precision, recall, and F1-score metrics. Class 0 demonstrates a moderate level of precision (0.60) and recall (0.69), leading to an F1-score of 0.64. On the other hand, Class 1 and Class 2 exhibit a precision, recall, and F1-score of 0.00, suggesting that the classifier failed to make any accurate predictions for these particular classes. Class 3 exhibits a relatively low level of precision (0.27) and recall (0.09), leading to a low F1-score of 0.14. Class 4 exhibits exceptional performance with a high level of accuracy (0.85), sensitivity (0.94), and F1-score (0.90). These findings align with the fact that Class 0 and Class 4 have the highest and second-highest levels of support in the dataset, respectively. Programming learners typically iterate on their work until they reach the highest possible score, resulting in the majority of learners' scores being categorized as either Class 0 or Class 4. The predictive model demonstrated an accuracy of 0.81 and an F1-score of 0.77, indicating a consistently dependable performance.

The results emphasize the strong correlation between PreLab performance and subsequent InLab outcomes, highlighting that actively participating in PreLab activities can significantly improve learning outcomes. The data

suggests that learners who engage in regular programming practice tend to review and improve their work until they achieve the highest scores possible. As a result, scores are clustered in Classes 0 and 4.

Table 7: Naive Bayes Classification results

Class	Precision	Recall	F1-score	Support
0	0.60	0.69	0.64	77
1	0.00	0.00	0.00	18
2	0.00	0.00	0.00	23
3	0.27	0.09	0.14	32
4	0.85	0.94	0.90	482
Accuracy				0.81
F1-score				0.77

6. Limitations

The findings of this study should be interpreted with caution due to several limitations that need to be considered.

- First, implementing the flipped classroom model in this research primarily demonstrates its feasibility in a practical setting but needs more integration with learning analytics (LA). LA support is essential for maximizing the utilization of data-driven insights to enhance the flipped classroom process, as it may restrict the comprehension of how this model impacts student engagement and self-study. Subsequent investigations should prioritize the integration of LA to offer a more all-encompassing strategy for enhancing the flipped classroom experience.
- Second, the study's use of the Naive Bayes algorithm for predicting InLab performance based on all PreLab activities collectively may not fully capture the nuances of student learning progression. A more precise method would involve making predictions in real-time, such as using PreLab 1 to predict InLab 1, PreLab 2 to predict InLab 2, and so forth. This would allow more timely and targeted interventions, potentially improving the accuracy and effectiveness of the predictions. Further research should aim to refine the predictive model to enhance its application in real-world educational settings.
- Third, this study's data collection was limited to a single semester, which restricts the ability to assess the long-term impacts of the flipped classroom model on student engagement and learning outcomes. Evaluating the effects over multiple semesters would provide a more robust understanding of the approach's sustainability and consistency. Future research should incorporate a longer duration of data collection in order to assess the long-lasting impact of this instructional approach accurately.

7. Conclusion

This paper explores the effectiveness of implementing the flipped classroom model in a practical programming course. We compared the traditional classroom model, the general flipped classroom model, and the flipped classroom model tailored explicitly for teaching practical programming regarding two critical criteria: student engagement and student performance.

The effectiveness of the flipped classroom model in the practical programming course is demonstrated in various aspects. Firstly, most students have a consensus, which is reflected in the high participation rate in pre-class activities. Secondly, measuring learners' learning time allows for assessing their investment in the new teaching method, with longer study durations indicating higher effectiveness. Thirdly, illustrative data collected from the Moodle system enables the observation of learners' development over time, aiding in assessing their self-study progress. Finally, the impact of completing pre-class activities before in-class labs is evident, with students who fully meet the pre-class assignments consistently achieving higher scores in the subsequent in-class tests.

Moreover, the data analysis revealed that students who performed well in PreLab activities had higher scores in the subsequent InLab sessions, indicating a strong positive correlation between PreLab and InLab results. Additionally, increased attempts and engagement with PreLab questions were associated with improved

performance in the InLab. Naive Bayes classification demonstrated promising predictive effects, with high accuracy and support for specific score ranges in the InLab.

In the future, we will investigate a more detailed teaching model that includes homework, tentatively called PostLab. Based on PreLab's results, clustering students into groups for practical InLab sessions could be a potential research direction.

Acknowledgements

This research is funded by Vietnam National University Ho Chi Minh City (VNU-HCM) under grant number DS2022-20-07. The authors also acknowledge Ho Chi Minh City University of Technology (HCMUT), VNU-HCM for supporting this study.

References

Abeysekera, L. and Dawson, P., 2014. Motivation and cognitive load in the flipped classroom: definition, rationale and a call for research. *Higher Education Research & Development*, 34(1), pp. 1–14. https://doi.org/10.1080/07294360.2014.934336

Ahmaderaghi, B., Barlaskar, E., Pishchukhina, O., Cutting, D. and Stewart, D., 2024. Enhancing Students' Performance in Computer Science Through Tailored Instruction Based on their Programming Background. In: Proceedings of 2024 IEEE Global Engineering Education Conference (EDUCON), pp. 1-5. https://doi.org/10.1109/EDUCON60312.2024.10578709

Bellaj, M., Dahmane, A.B., Boudra, S. and Sefian, M.L., 2024. Educational Data Mining: Employing Machine Learning Techniques and Hyperparameter Optimization to Improve Students' Academic Performance. *International Journal of Online and Biomedical Engineering (iJOE)*, 20(03), pp. 55–74. https://doi.org/10.3991/ijoe.v20i03.46287

Benesty, J., Chen, J., Huang, Y. and Cohen, I., 2009. Pearson Correlation Coefficient. In: Noise Reduction in Speech Processing. *Springer Topics in Signal Processing*, vol 2. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-00296-0_5

Blanquero, R., Carrizosa, E., Ramirez-Cobo, P. and Sillero- Denamiel, M.R., 2021. Variable selection for Naive Bayes classification. *Computers & Operations Research*, vol. 135, Article 105456. https://doi.org/10.1016/j.cor.2021.105456

Caviativa, Y.P., Castro, F.C., Guzman, V.J. and Sanz, F.A., 2022. Educational resources with digital content for pedagogical and research teaching in technologies of knowledge and learning. In: Proceedings of the 40th Central America and Panama Convention (CONCAPAN), pp. 1-6. https://doi.org/10.1109/CONCAPAN48024.2022.9997770

Cerezo, R., Lara, J., Azevedo, R. and Romero, C., 2024. Reviewing the differences between learning analytics and educational data mining: Towards educational data science. *Computers in Human Behavior*, 154, 108155. https://doi.org/10.1016/j.chb.2024.108155

Davis, N.L., 2016. Anatomy of a flipped classroom. *Journal of Teaching in Travel & Tourism*, *16*(3), 228–232. https://doi.org/10.1080/15313220.2015.1136802

Dietrich, H. and Evans, T., 2022. Traditional lectures versus active learning – A false dichotomy? *STEM Education*, 2(4), pp. 275-292. https://doi.org/10.3934/steme.2022017

Estaji, M. and Jonaidi-Jafari, L., 2022. Self-directed learning vs teacher-led instruction: Effects on oral proficiency and structural accuracy. *Electronic Journal of Foreign Language Teaching (e-FLT), 19*(2), 145–160. https://doi.org/10.56040/esjo1923

Farhana, S., 2021. Classification of Academic Performance for University Research Evaluation by Implementing Modified Naive Bayes Algorithm. *Procedia Computer Science*, vol. 194, pp. 224–228. https://doi.org/10.1016/j.procs.2021.10.077

Fessakis, G., Gouli, E. and Mavroudi, E., 2013. Problem solving by 5–6 years old kindergarten children in a computer programming environment: A case study. *Computers & Education*, vol. 63, pp. 87–97. https://doi.org/10.1016/j.compedu.2012.11.016

Förster, M., Maur, A., Weiser, C. and Winkel, K., 2022. Pre-class video watching fosters achievement and knowledge retention in a flipped classroom. *Computers & Education*, v179, 104399. https://doi.org/10.1016/j.compedu.2021.104399

Gamage, S.H.P.W., Ayres, J.R. and Behrend, M.B., 2022. A systematic review on trends in using Moodle for teaching and learning. *International Journal of STEM Education* 9, Aritcle 9. https://doi.org/10.1186/s40594-021-00323-x

Groher, I., Sabitzer, B., Demarle-Meusel, H., Kuka, L. and Hofer, A., 2021. Work-in-Progress: Closing the Gaps: Diversity in Programming Education. In 2021 IEEE Global Engineering Education Conference (EDUCON), pp. 1449-1453. https://doi.org/10.1109/EDUCON46332.2021.9454035

Han, E., and Klein, K.C., 2019. Pre-Class Learning Methods for Flipped Classrooms. *American Journal of Pharmaceutical Education*, 83(1), 6922. https://doi.org/10.5688/ajpe6922

Hellín, C.J., Valledor, A., Gómez, J. and Tayebi, A., 2022. Enhancing Student Motivation and Engagement through a Gamified Learning Environment. *Sustainability*, 15(19), 14119. https://doi.org/10.3390/su151914119

Jin-gang, J., Jia-wei, Z., Hai-yan, D., Kai-rui, W., Ye, D. and De-dong, T., 2024. Application of Flipped Classroom Teaching Method in the Single-Chip Microcomputer Course. In: Proceedings of the 13th International Conference on Educational and Information Technology (ICEIT), pp. 184-188. https://doi.org/10.1109/ICEIT61397.2024.10540809

Kadar, R., Mahlan, S.B., Shamsuddin, M., Othman, J. and Wahab, N.A., 2022. Analysis of Factors Contributing to the Difficulties in Learning Computer Programming among Non-Computer Science Students. In: Proceedings of the 12th Symposium on Computer Applications & Industrial Electronics (ISCAIE), pp. 89-94. https://doi.org/10.1109/ISCAIE54458.2022.9794546

Kalelioğlu, F., 2015. A new way of teaching programming skills to K-12 students: Code.org. *Computers in Human Behavior*, vol. 52, pp. 200–210. https://doi.org/10.1016/j.chb.2015.05.047

Kraml, M., 2024. Advantages and Disadvantages of Flipped Classroom in Adult Education using Distance Learning for Learning Programming. *EPH-International Journal of Educational Research*, 8(1), 26-31. https://doi.org/10.53555/ephijer.v8i1.109

Li, F., Qi, J., Wang, G. and Wang, X., 2014. Traditional Classroom vs E- learning in Higher Education: Difference between Students' Behavioral Engagement. *International Journal of Emerging Technologies in Learning (iJET)*, 9(2), p. 48–51. https://doi.org/10.3991/ijet.v9i2.3268

Liu, F., Wang, X. and Izadpanah, S., 2023. The Comparison of the Efficiency of the Lecture Method and Flipped Classroom Instruction Method on EFL Students' Academic Passion and Responsibility. *SAGE Open*, *13*(2). https://doi.org/10.1177/21582440231174355

Liu, J., Cao, S., Liu, X., Ye, C. and Siano, P., 2024. Pre-class mode "flipped" again: Making videos instead of just watching them. *Heliyon*, 10(6), e28105. https://doi.org/10.1016/j.heliyon.2024.e28105

Liu, Q., Gladman, T., Muir, J., Wang, C. and Grainger, R., 2023. Analytics-Informed Design: Exploring Visualization of Learning Management Systems Recorded Data for Learning Design. *SAGE Open*, 13(3). https://doi.org/10.1177/21582440231193590

Lobb, R. and Harlow, J., 2016. Coderunner: a tool for assessing computer programming skills. *ACM Inroads*, 7(1), pp. 47-51. https://doi.org/10.1145/2810041

López-Pimentel, J.C., Medina-Santiago, A., Alcaraz-Rivera, M. and Del-Valle-Soto, C., 2021. Sustainable Project-Based Learning Methodology Adaptable to Technological Advances for Web Programming. *Sustainability*, 13(15), 8482. https://doi.org/10.3390/su13158482

Malkoc, S., Steinmaurer, A., Gütl, C., Luttenberger, S. and Paechter, M., 2024. Coding Decoded: Exploring Course Achievement and Gender Disparities in an Online Flipped Classroom Programming Course. *Education Sciences*, 14(6), Article 634. https://doi.org/10.3390/educsci14060634

Martins, G., Lopes, P.S., Conte, D.J. and Bruschi, S.M., 2020. Learning Parallel Programming Through Programming Challenges. In: Proceedings of IEEE Frontiers in Education Conference (FIE), pp. 1-9. https://doi.org/10.1109/FIE44824.2020.9274009

Papadakis, S. and Kalogiannakis, M., 2019. Evaluating the effectiveness of a game-based learning approach in modifying students' behavioural outcomes and competence in an introductory programming course: A case study in Greece. *International Journal of Teaching and Case Studies*, 10(3), pp. 235-250. https://doi.org/10.1504/IJTCS.2019.102760

Papadakis, S., 2020. Evaluating a game-development approach to teach introductory programming concepts in secondary education. *International Journal of Technology Enhanced Learning (IJTEL)*, 12(2), pp. 127–145. https://doi.org/10.1504/IJTEL.2020.106282

Pears, A., 2010. Enhancing student engagement in an introductory programming course. In: Proceedings of the IEEE Frontiers in Education Conference (FIE), pp. F1E-1-F1E-2. https://doi.org/10.1109/FIE.2010.5673334

Pears, A., 2010. Enhancing student engagement in an introductory programming course. In: Proceedings of the IEEE Frontiers in Education Conference (FIE), pp. F1E-1-F1E-2. https://doi.org/10.1109/FIE.2010.5673334

Pliuskuvienė, B., Radvilaitė, U., Juodagalvytė, R., Ramanauskaitė, S. and Stefanovič, P., 2024. Educational data mining and learning analytics: Text generators usage effect on students' grades. *New Trends in Computer Sciences*, 2(1), 19–30. https://doi.org/10.3846/ntcs.2024.21318

Rabie, A.H., Mansour, N.A., Saleh, A.I. and Takieldeen, A.E., 2022. Expecting individuals' body reaction to Covid-19 based on statistical Naive Bayes technique. *Pattern Recognition*, vol. 128, Article108693. https://doi.org/10.1016/j.patcog.2022.108693

Rivera, M., and Flores, G., 2024. Flipped classroom approach for enhancing linguistic competence. *International Journal of Evaluation and Research in Education (IJERE)*, 13(5), pp. 3369-3378. https://doi.org/10.11591/ijere.v13i5.27365

Rouhani, M., Lillebo, M., Farshchian, V. and Divitini, M., 2022. Learning to Program: an In-service Teachers' Perspective. In: Proceedings of 2022 IEEE Global Engineering Education Conference (EDUCON), pp. 123-132. https://doi.org/10.1109/EDUCON52537.2022.9766781

Sánchez-Cuervo, M.-E., 2024. Flipped Classroom in the English Literature Lesson. In: Proceedings of the World Conference on Research in Education, 1(1), pp. 35-45. https://doi.org/10.33422/worldcre.v1i1.211

Santos, S., Tedesco, P., Borba, M. and Brito, M., 2020. Innovative Approaches in Teaching Programming: A Systematic Literature Review. In: Proceedings of the 12th International Conference on Computer Supported Education (CSEDU), vol. 1, pp. 205-214. https://doi.org/10.5220/0009190502050214

Shah, A., Alhumrani, F., Griswold, W.G., Porter, L. and Soosai Raj, A.G., 2024. A Comparison of Student Behavioral Engagement in Traditional Live Coding and Active Live Coding Lectures. In: Proceedings of the 2024 on Innovation and Technology in Computer Science Education V.1, pp. 513–519. https://doi.org/10.1145/3649217.3653537

Shahamiri, S.R., 2019. The Challenges of Teaching and Learning Software Programming to Novice Students. In M. Khosrow-Pour (Ed.), *Advanced Methodologies and Technologies in Network Architecture, Mobile Computing, and Data Analytics*, pp. 1350-1357. https://doi.org/10.4018/978-1-5225-7598-6.ch099

Silverajah, V.S.G., Wong, S., Govindaraj, A., Khambari, N., Wirza, R. and Deni, A., 2022. A Systematic Review of Self-Regulated Learning in Flipped Classrooms: Key Findings, Measurement Methods, and Potential Directions. *IEEE Access*, vol. 10, pp. 20270-20294. https://doi.org/ACCESS.2022.3143857

Singh, A., Singh, D. and Chhikara, S., 2024. Challenges and Suggestions to Enhance Student Engagement in Higher Education Institutions. https://doi.org/10.4018/979-8-3693-0409-9.ch004

Sobral, S.R., 2021. Flipped Classrooms for Introductory Computer Programming Courses. *International Journal of Information and Education Technology*, 11(4), pp. 178–183. https://doi.org/10.18178/ijiet.2021.11.4.1508

Vishwakarma, M. and Kesswani, N., 2023. A new two-phase intrusion detection system with Naive Bayes machine learning for data classification and elliptic envelop method for anomaly detection. *Decision Analytics Journal*, vol. 7, Article 100233. https://doi.org/10.1016/j.dajour.2023.100233

White, P.J., Naidu, S., Yuriev, E., Short, J.L., McLaughlin, J.E. and Larson, I.C., 2017. Student Engagement with a Flipped Classroom Teaching Design Affects Pharmacology Examination Performance in a Manner Dependent on Question Type. *American Journal of Pharmaceutical Education*, 81(9), Article 5931. https://doi.org/10.5688/ajpe5931

Zhou, L., 2024. The Application of Flipped Classroom Teaching Model in High School Class. *Frontiers in Science and Engineering*, 4(7), pp. 170-175. https://doi.org/10.54691/6c43ny64

