ALFA BK UNIVERSITY
FACULTY OF INFORMATION TECHNOLOGY
FACULTY OF MATHEMATICS AND COMPUTER SCIENCE
ALFATECH Journal

Alfa BK Univerzitet

UDK: 004.8:159.953
DOI: 10.46793/AlfaTech1.2.28S
Original scientific paper

Generative Artificial Intelligence and Studying Programming Subjects at Universities: From Students Motivation to Comparative Analysis of AI and Human Code

Olga SHVETS[1], Andrey PESTUNOV[2], Egor CHEGLOV[3]

Abstract: The rapid expansion of generative artificial intelligence (AI) is reshaping process of studying programming subjects, as students increasingly employ AI tools to solve academic problems. In this paper we investigate student's motivation for using generative AI, their perceived advantages and drawbacks, and the distinctions between human-written and AI-generated code. Building upon earlier research on the impact of AI on academic performance, this paper analyses an experimental dataset comprising 950 code samples derived from student and AI solutions to identical programming tasks. The dataset was processed using Python-based analytical tools to extract structural and stylistic metrics, enabling a comparative assessment of the behaviour of different generative models. The findings reveal that certain models, particularly ChatGPT, exhibit greater stylistic proximity to student code, while others demonstrate more formalized programming patterns. The results contribute to a deeper understanding of AI's role in process of studying programming subjects and provide a methodological foundation for developing tools to evaluate and monitor AI-assisted learning.
Keywords: generative artificial intelligence, studying programming subjects, code analysis, AI-assisted learning, digital transformation of education.

1 INTRODUCTION

With the advancement of generative artificial intelligence (AI), students have gained access to computational tools that facilitate the automation of academic tasks across various disciplines. Previous studies [1] have examined the impact of generative AI on process of studying programming subjects, highlighting its growing influence on students' approaches to learning and problem-solving.

The 2023 study conducted by researchers from Darmstadt University of Applied Sciences (Germany), sampling over 8,800 students, examines the prevalence and domains of AI in education. The findings indicate that more than 63% of respondents incorporate AI-driven technologies into their learning processes. The adoption rate is notably higher among students in engineering disciplines (75%), followed by those in the natural sciences (71%), humanities (61%), and medical fields (52%) [2].

Programming is one of the academic disciplines most closely associated with AI. The integration of AI into process of studying programming subjects presents both advantages and disadvantages [3]. Generative AI facilitates the development of solutions by autonomously producing accurate and comprehensible program code, thereby significantly reducing the time and computational resources required to complete assignments. However, the incorporation of generative AI into the educational process has sparked extensive debate among educators and researchers.

On the one hand, AI can function as a personalized assistant for students, providing explanations of algorithms and guiding the development of complex software structures, as well as systematically organizing and presenting information in an accessible format [4]. On the other hand, concerns have been raised that relying on generative AI may diminish students' motivation for independent problem-solving and impede the advancement of critical and creative thinking skills, potentially leading to a decline in overall knowledge acquisition [5].

This raises several critical questions: does the use of generative artificial intelligence (AI) pose a threat to the educational process? Can it facilitate a deeper understanding of complex topics by freeing up time for analysis and creativity, or, conversely, does it deprive students of essential hands-on experience?

The objective of this study is to examine students' experiences in utilizing generative AI for solving programming-related learning tasks. The research aims to identify the distinctive characteristics and patterns of AI usage among students from various academic backgrounds, thereby providing a more comprehensive understanding of how AI integration influences the educational process in the context of programming instruction.

Furthermore, this study includes an analysis of an electronic programming course hosted on the university's Learning Management System (LMS). Universities commonly employ the CodeRunner plugin for the automated assessment of students' solutions [6].

Traditional teaching process often assumes that a teacher plays both roles: teaching and checking. In such a situation, the second (checking) role may dominate over the first one and the student may fall into some sort of confrontation with the teacher. Automatic systems such as CodeRunner take the checking role by themselves and the teacher becomes an ally with the student. Their common goal is to overcome all checking barriers of the automatic system. The teacher doesn't help his student directly, but tries to help him (her) to handle the problem.

However, GAI (generative AI) may spoil these educational process model by creating temptation of

exploiting it in order to solve the problem with the help of GAI instead of the teacher. There exists a difference between the teacher's and the GAI's help. The teacher is able to provide the student with the partial solution and the student would try to reach the final one. But GAI will elaborate the final solution at once and the student will be satisfied with that, thus, eliminating the teaching effect.

Thus, the study explores the challenge of detecting generative AI usage in academic assignments and proposes methods for identifying such cases within an automated verification environment.

2 RESEARCH METHODOLOGY

This study investigates the perceptions and experiences of students utilizing various generative AI models to complete programming tasks. To facilitate data collection, a structured questionnaire was designed, combining both quantitative measures and open-ended questions to capture qualitative insights.

The survey was administered to students enrolled in programming-related disciplines, including "Fundamentals of Algorithmizing and Programming" (secondary vocational education), as well as "Programming", "Programming Languages," and "Programming of Discrete Structures" (bachelor's degree programs). Participation was voluntary and anonymous. A total of 61 students participated, comprising 29 students from secondary vocational education programs and 32 students from bachelor's degree programs.

In addition, the study incorporated 100 tasks of basic complexity from an electronic course, organized into eight thematic blocks. The experimental procedure involved simulating the behaviour of a student who copies the task description from the LMS and inputs it into a generative AI model. The AI-generated solution is then submitted to an automated testing system for verification without modification. In case of failure, the student iterates the process by submitting clarifying queries to the AI model until the solution passes the tests successfully.

3 SURVEY RESULTS

According to the results of the survey, 34 students (55.7%) reported possessing only basic programming knowledge, while 11 students (18%) indicated that they were learning programming for the first time. Another 11 respondents (18%) self-assessed their skills as confident in one programming language, and only 5 students (8.2%) stated that they are proficient in multiple programming languages.

As illustrated in Figure 1, which presents the question "How would you rate your algorithms knowledge?", 47.5% of respondents acknowledged a lack of familiarity with algorithms, while 42.6% indicated that they were acquainted with basic ones, such as brute force or bubble sort. Only 6 students (9.8%) reported proficiency in more advanced algorithms, including binary search, quick sort, or Euclid's algorithm.

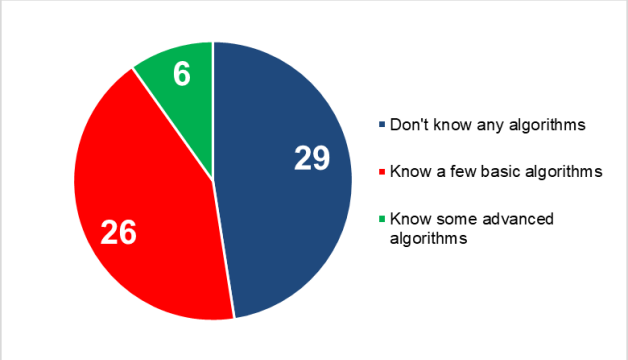

Figure 1: How would you rate your algorithms knowledge?

An analysis of the responses revealed two distinct groups based on the students' level of proficiency in programming and algorithms. Among those with limited or basic programming knowledge, the use of generative AI was most prevalent due to challenges faced after several unsuccessful attempts to complete tasks independently (67%) and the need to debug their own code (65%), as presented in Figure 2. In contrast, students with more advanced programming skills in one or more languages were more likely to turn to AI due to poor understanding of task conditions or the meaning of the problem (44%) and a lack of motivation to solve the task independently (38%). Additionally, both groups expressed a desire to obtain optimized solutions (13.3% of the total respondents).

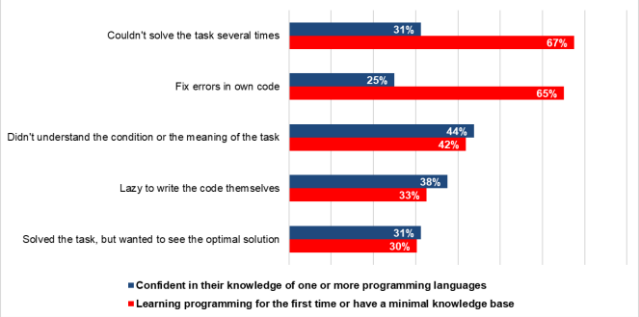

Figure 2: What is your main reason for using generative AI?

An analysis of the frequency of AI usage revealed that 73.8% of the respondents use it 1–3 times per week, 16.4% use it daily, and 9.8% use it 4–5 times per week.

The students identified several significant advantages of using generative AI in programming tasks. As shown in Figure 3, the most frequently mentioned benefits included time-saving, reported by 82.2% of respondents, and assistance in mastering new and complex algorithms, cited by 74.2%. Additional benefits highlighted by students included faster and improved learning processes, more accessible and comprehensible explanations and solutions compared to traditional teacher-provided explanations, as well as increased self-confidence and enhanced motivation to learn.

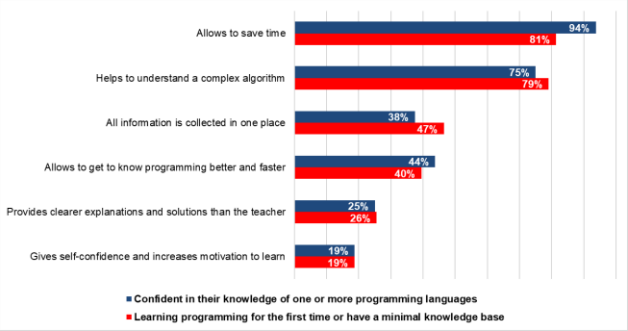

Figure 3: What is your main reason for using generative AI?

While a small percentage of students (2.9%) did not perceive any disadvantages in using generative AI for programming tasks, the majority identified several notable negative aspects. As presented in Figure 4, entry-level students with minimal programming knowledge most frequently pointed out the inaccuracy of the generated code (72%) and the risk of over-reliance on AI (44%), which they believed weakened their algorithmic thinking abilities (35%). Students with more advanced knowledge also highlighted frequent inaccuracies in the generated solutions (88%) and the issue of AI misinterpreting their requests, which often led to misleading results (50%). Other disadvantages noted by respondents included a perceived lack of creativity and variability in problem-solving.

The survey results suggest that the majority of students (61%) require multiple clarifying queries to obtain a correct solution from generative AI, although 34% of respondents reported successfully resolving the task on the first attempt. These findings may indicate that many students have not yet developed the skills to accurately formulate queries to AI and tend to simply copy the task description and submit it without modification.

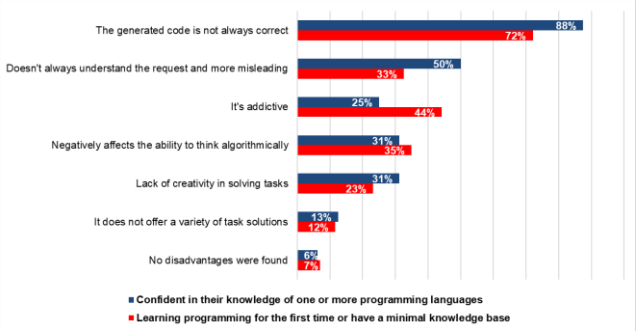

Figure 4: What disadvantages can you identify from using generative AI?

Furthermore, the survey revealed that students rarely present AI-generated solutions as their own work (only 11%). As indicated in Figure 5, they use AI primarily to improve their own solutions: 43% of respondents integrate part of the generated code into their projects, and 46% modify the proposed code before submitting it for evaluation.

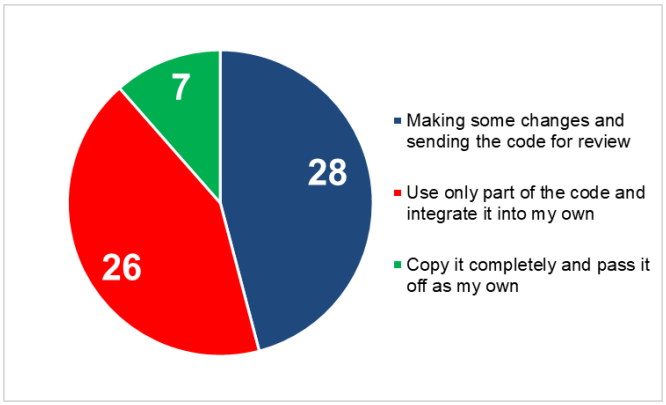

Figure 5: What disadvantages can you identify from using generative AI?

One of the primary objectives of this study was to identify the differences in problem-solving approaches between generative AI models and students. While an experienced instructor can typically distinguish between independently written and externally generated code, the challenge lies in training an automated system to detect such instances and respond appropriately in real time.

Generative AI adheres strictly to the programming standards, ensuring consistent code formatting and comprehensive comments—an approach that is uncommon among most students. Furthermore, AI-generated solutions often incorporate advanced programming language features that are not typically covered at the initial stages of learning, which may raise suspicions during the assessment of student work. Additionally, variations in variable naming conventions can serve as an indicator of AI involvement: while AI tends to assign names based on the semantic meaning of the variable's contents, students are more likely to use abstract or inconsistent naming patterns.

The identification of AI-generated code can be facilitated through various quantitative and qualitative metrics. Execution speed and memory consumption serve as fundamental indicators, as AI-generated code is often optimized for performance. This optimization stems from the efficient utilization of built-in programming language functions and advanced algorithms. In contrast, student-written code may exhibit higher memory consumption due to redundant variables or inefficient data structures.

Additional quantitative metrics indicative of GAI usage include features that are less commonly observed in student solutions, such as the frequency of inline comments, function definitions (including lambda functions), try-except blocks, and list comprehensions.

Furthermore, qualitative metrics can provide deeper insights into AI-generated code characteristics. These include adherence to programming style standards, the average length of variable and function names, Halstead Difficulty, which quantifies cognitive complexity in code comprehension, and Cyclomatic Complexity, a metric that assesses algorithmic complexity by measuring the number of linearly independent execution paths. The integration of these metrics enables a more comprehensive analysis of code provenance and aids in distinguishing between human-written and AI-generated solutions.

4 COMPARATIVE ANALYSE OF GENERATIVE AI AND STUDENT CODE

To extend the survey-based findings, an experimental dataset was developed to analyse the stylistic and structural similarities between student-written and AI-generated code.

The dataset is based on ten programming tasks from the topic “While Loop”, a foundational concept studied in introductory programming courses.

For each task, 50 student-written solutions and 45 generative AI solutions were collected, resulting in a total of 950 code samples — 500 authored by students and 450 by AI models.

The AI-generated solutions were produced using the systems most frequently mentioned by students in the survey: ChatGPT, Microsoft Copilot, Grok, Qwen, DeepSeek, Claude, Gemini, GigaChat, and YandexGPT.

Each model received the same task prompt and was instructed to generate a standard baseline solution to the problem, three alternative solutions implementing different approaches and one additional solution imitating a student-like style — with simplified logic, inconsistent formatting, abstract variable names, and missing comments.

The dataset was processed using a Python program that extracted stylistic and structural metrics with the help of libraries such as radon, ast, and flake8.

All numerical metrics were normalized and analysed using scikit-learn tools. A Random Forest classifier was trained to distinguish student and AI-written code and then used to compute the average probability of being classified as “student” for each AI model. This value served as an indicator of the model’s stylistic similarity to students’ programming behavior.

The results, visualized in Figure 6, demonstrate that ChatGPT exhibited the highest resemblance to student writing style, followed by Claude, DeepSeek, and Qwen. In contrast, GigaChat and Gemini displayed the lowest similarity, often producing highly optimized or syntactically uniform code that deviates from the typical structure of student solutions.

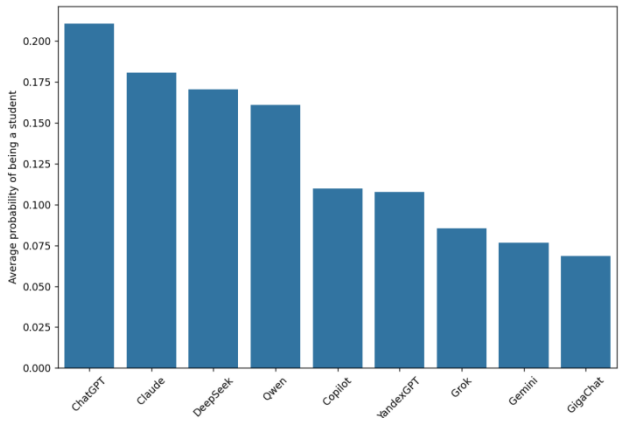

Figure 6: Probability of being classified as student

These results suggest that certain generative systems — particularly ChatGPT — tend to mimic human programming style more closely, producing solutions that reflect beginner-level logic and structure, while others follow formalized or industrial programming conventions that make them easily distinguishable from student work.

5 CONCLUSION

The findings of this study provide an opportunity to evaluate the feasibility of integrating generative AI into the educational process. On the one hand, AI can serve as a tool for personalized student support, adapting to their level of proficiency and individual learning needs. On the other hand, its widespread use may adversely affect the development of independent problem-solving skills and critical thinking. These results highlight the necessity for further research on AI integration in education, with a particular focus on developing methodologies that foster precise problem formulation skills and promote the effective use of AI as an educational aid without diminishing students' cognitive engagement and motivation for independent work.

The extended experimental stage introduced in this paper demonstrates that it is possible to compare stylistic characteristics of AI- and student-generated code using quantitative metrics. The developed Python-based analytical program provides a methodological foundation for future studies involving larger datasets and diverse task types. In subsequent research, such tools could support automated monitoring of AI-assisted academic work and the development of adaptive learning environments that balance human creativity with intelligent technological assistance.

6 REFERENCES

[1] Cheglov E., Pestunov A., Shvets O., Digital Transformation of education: the impact of generative AI on programmer training, Proceedings (Milan Gligorijevic), International Scientific and Professional Conference “ALFATECH” Smart Cities and modern technologies, Belgrade, Serbia, 2025, pp. 4-7

[2] Barannikov K.A., Dobryakova M.S., Novikova E.G., Ten N.G., Artificial Intelligence and Higher Education: Opportunities, Practices, and the Future, https://disk.yandex.ru/d/b45m7TMG49mSUg

[3] Yilmaz R., Karaoglan Yilmaz F.G., Augmented intelligence in programming learning: Examining student views on the use of ChatGPT for programming learning, Computers in Human Behaviour: Artificial Humans, 1, (2023), Issue 2, Article 100005, DOI: https://doi.org/10.1016/j.chbah.2023.100005

[4] Khaniev R.M., Use of innovative technologies in the learning process in higher education, Proceedings, IV All-Russian Scientific and Practical Conference “Problems and Prospects of Development of Social, Economic, and Humanitarian Sciences: Pedagogy, Psychology, Economics, Jurisprudence”, Pokrov, Russia, 2024, pp. 119-124

[5] Valiakhmetova N.R., Akhmadullina R.M., Yarmakeev I.E., Opportunities and risks of applying neural networks in education, Philology and Culture, 2, (2024), pp. 260-271

[6] Karmanova E.V., Automated control in teaching Python programming using CodeRunner LMS MOODLE plugin, Proceedings, XIV International Scientific and Practical Conference “SCIENCE. INFORMATIZATION. TECHNOLOGIES. EDUCATION”, Yekaterinburg, Russia, 2021, pp. 102-108

Contact information:

Olga SHVETS 1, PhD in Computer Science, Associate Professor, Novosibirsk State University of Economics and Management, Kamenskaya st., 52, Novosibirsk, Novosibirsk region, Russian Federation, 630099, o.y.shvec@edu.nsuem.ru, https://orcid.org/0000-0001-5710-9056
Andrey PESTUNOV 2, PhD in Physics and Mathematics, Associate Professor, Novosibirsk State University of Economics and Management, Kamenskaya st., 52, Novosibirsk, Novosibirsk region, Russian Federation, 630099, a.i.pestunov@nsuem.ru, https://orcid.org/0000-0002-4909-7953

Egor CHEGLOV, PhD student, Novosibirsk State University of Economics and Management, Kamenskaya st., 52, Novosibirsk, Novosibirsk region, Russian Federation, 630099, e.r.cheglov@edu.nsuem.ru