Research on Differentiated Teaching Model of Python Programming Based on Learning Data Analysis

Wei Zhang[*]**, Yingxin Zhou**

Lanzhou Resources & Environment Voc-Tech University, Lanzhou, 730020, China
[*]*Corresponding author:zwarcher@163.com*

Abstract: *With the widespread application of the Python language in education, how to improve the quality of programming education through differentiated teaching models has become an urgent issue to address. This research, based on learning data analysis, constructs a differentiated teaching model for Python programming courses, aiming to utilize students' learning data to achieve personalized and precise teaching. The model uses data collection and analysis to develop differentiated teaching content and personalized resource recommendation strategies, catering to students' varying learning levels and needs. The research results indicate that this model effectively enhances students' learning interest and programming abilities, offering innovative insights for Python teaching.*

Keywords: *learning data analysis; Python programming; differentiated teaching; personalized learning; educational innovation*

Introduction

Python programming has gradually become an essential course in both higher education and secondary schools, gaining popularity among teachers and students due to its wide application and ease of operation. However, during the learning process, students often exhibit differences in learning progress and outcomes due to variations in their knowledge base and learning styles. The traditional unified teaching model struggles to meet the diverse learning needs of students. Therefore, exploring a differentiated teaching model based on learning data analysis, which incorporates students' learning habits and ability levels to provide tailored learning paths and resource recommendations for each student, holds significant practical relevance and innovative value.

1. The Application of Python Programming in Teaching Through Learning Data Analysis

1.1 Basic Theories and Methods of Learning Data Analysis

Learning data analysis is a technical approach based on educational data mining and learning analytics. Its goal is to collect and analyze data on students' learning behaviors, habits, and performance to accurately identify their learning status and needs, thereby supporting personalized teaching. The core theoretical foundations include data mining, learning behavior analysis, and personalized recommendation systems. Data mining involves extracting meaningful patterns and insights from large volumes of learning data using statistical methods, machine learning, and pattern recognition to provide data-driven support for teaching decisions. Learning behavior analysis focuses on revealing students' knowledge acquisition levels, learning styles, and engagement through their learning behavior data. Personalized recommendation systems, through detailed analysis of students' learning data, provide tailored learning content, paths, and strategies that make the learning experience more targeted. In programming education, the application of learning data analysis enables teachers to identify students' knowledge gaps, learning habits, and personalized needs at different stages of learning, allowing for the development of targeted teaching strategies and resources to enhance learning outcomes and programming skills.

1.2 Types of Python Programming Learning Data and Methods of Collection

In Python programming education, various types of learning data can be collected, covering aspects such as knowledge acquisition, learning behavior, and emotional feedback. First, knowledge acquisition data includes metrics such as exercise accuracy, error types, completion time, and number of code writing attempts, which reflect students' understanding of Python concepts and their proficiency in programming skills. Second, learning behavior data primarily includes metrics such as course participation, online study duration, video viewing frequency, and course completion rates, which reveal students' learning habits and attitudes. Additionally, emotional feedback data can be collected through emotion analysis and interaction records, providing insights into students' emotional states and learning motivation during programming studies.[1]

These data are mainly collected through tools such as learning management systems, online coding platforms, and educational apps to systematically monitor and dynamically track students' learning progress. Comprehensive data collection allows teachers to accurately grasp students' performance and needs during the programming learning process.

1.3 Teaching Improvement Supported by Learning Data Analysis

Python programming education can achieve targeted teaching improvements based on learning data analysis. First, teachers can use the results of learning data analysis to implement differentiated teaching. By analyzing students' knowledge acquisition and learning behavior, students can be grouped into different levels, enabling targeted teaching support.
Secondly, teachers can customize personalized learning resources for students based on their learning progress and difficulties, such as recommending specific programming exercises, study materials, or online courses to meet the needs of students at different levels. Furthermore, through emotional feedback data, teachers can adjust teaching methods in a timely manner to stimulate students' interest in learning and alleviate learning anxiety.

Finally, learning data analysis supports the establishment of a real-time feedback and evaluation mechanism, allowing teachers to adjust teaching strategies according to students' real-time performance, improving the adaptability and flexibility of teaching. Teaching improvements based on learning data analysis can not only enhance the efficiency of programming education but also significantly improve students' learning experiences and programming skills.[2]

2. Design of a Differentiated Teaching Model for Python Programming Based on Learning Data Analysis

2.1 Theoretical Foundations of Differentiated Teaching Model Design

The design of a differentiated teaching model is supported by multiple educational theories, primarily including constructivist learning theory, personalized learning theory, and adaptive learning theory, which provide a theoretical foundation for innovation in programming education. Constructivist learning theory posits that learning is a process in which students actively construct knowledge, emphasizing student-centered teaching and encouraging learners to follow their own paths of understanding to form personalized knowledge systems. As a result, differentiated teaching should focus on supporting students' choices of personalized learning paths.
Personalized learning theory highlights the unique learning needs of each student, requiring the teaching model to flexibly adapt to students' varied learning progress, interests, and needs, thereby optimizing their learning experience and enhancing learning outcomes.
Adaptive learning theory suggests that teaching should be dynamically adjusted based on students' ability levels, learning states, and personal characteristics to maximize learning efficiency and make teaching more adaptable.

Based on these theories, the differentiated teaching model leverages learning data analysis to accurately identify students' learning characteristics, and personalized teaching plans are developed in accordance with each student's actual situation to ensure their full development and progress in programming skills and thinking.

2.2 Student Stratification Based on Data Analysis

Student stratification based on learning data analysis is a core step in the differentiated teaching model. By systematically collecting and analyzing students' learning data, students are divided into multiple tiers according to their learning foundation, ability levels, and learning behaviors. This process typically involves a comprehensive analysis and assessment of data such as students' exercise accuracy, error frequency, learning engagement, and online learning time to precisely determine students' learning status. For instance, students can be categorized into three levels: beginner, intermediate, and advanced. Beginners usually show low knowledge mastery and weak programming foundations, focusing primarily on understanding and applying basic concepts. Intermediate students have a solid foundation in programming knowledge but still have room for improvement in handling complex concepts and optimizing code. Advanced students demonstrate strong programming abilities and logical thinking, capable of tackling difficult programming tasks and exploring more advanced programming techniques and code optimization methods.

Through stratification, teachers can gain deeper insights into the learning status of students at different levels and obtain accurate data support for designing differentiated teaching strategies. This ensures that each student receives targeted teaching guidance at an appropriate starting point and learning pace, truly achieving personalized education goals.[3]

2.3 Development of Differentiated Teaching Content and Progress

Based on student stratification, the differentiated teaching model should develop targeted teaching content and flexible learning progress plans according to the learning characteristics and needs of each level. For beginners, teaching content should focus on basic programming concepts, simple code writing, and understanding common syntax, helping them solidify foundational knowledge and build learning confidence. Intermediate students, while reinforcing foundational knowledge, should gradually delve into program structure, algorithm design, and logical thinking training to enhance their coding and problem-solving abilities. For advanced students, the content can shift toward more complex algorithms, data structures, and advanced programming techniques, with in-depth code optimization training. Through challenging project tasks, advanced students can cultivate innovation and application skills, achieving a higher level of programming expertise.

The arrangement of learning progress should align with each group's knowledge absorption speed and comprehension ability, ensuring that the teaching pace neither creates undue pressure nor hinders progress. The differentiated teaching model should also incorporate a dynamic adjustment mechanism, regularly evaluating and providing feedback on students' learning progress. By combining the results of learning data analysis, teachers can adjust the teaching content and difficulty levels based on each student's actual progress. This personalized dynamic adjustment mechanism allows teachers to provide real-time support throughout the learning process, making teaching content more suited to students' developmental needs. Thus, true personalized and adaptive teaching is achieved, improving both the effectiveness of programming education and the overall student learning experience.

2.4 Personalized Learning Resource Recommendations

Personalized learning resource recommendations are an essential support measure for implementing differentiated teaching. Through learning data analysis, the specific learning needs of students at different levels can be accurately identified, and the most suitable learning resources are provided to help them address knowledge gaps, consolidate foundational knowledge, and expand advanced content. For example, beginners can be provided with basic Python video tutorials, introductory exercises, and knowledge maps to gradually master fundamental concepts and programming syntax, building a solid foundation. For intermediate students, the system will recommend moderately difficult programming tasks, code debugging exercises, and introductory knowledge of data structures and algorithms to strengthen their application skills and logical thinking. Advanced students will receive more challenging projects, comprehensive practical tasks, and programming case studies in specialized fields, allowing them to further deepen and expand their programming techniques, enhancing their independent inquiry and programming innovation abilities.[4]

Through personalized resource recommendations, teachers can ensure that students at every stage of learning receive resources and support tailored to their current level. This approach not only precisely meets students' learning needs but also significantly enhances their initiative and interest in learning, providing them with a richer and more flexible learning experience during their programming studies. This strategy effectively enhances their programming skills while fostering their ability to learn independently and cultivate an exploratory spirit, laying a solid foundation for their long-term development.

3. Implementation Strategies for Differentiated Teaching Model of Python Programming Based on Learning Data Analysis

3.1 Implementation of Tiered Instruction

In the differentiated teaching model for Python programming, tiered instruction is a key step in the implementation strategy. Based on the student stratification results obtained from learning data analysis, teachers can categorize students into beginner, intermediate, and advanced levels, designing corresponding teaching activities and content according to the learning characteristics of each group. For beginners, teachers may adopt simplified teaching content, detailed code demonstrations, and more fundamental exercises to help students understand basic concepts such as variables and functions, establishing foundational programming thinking. Additionally, providing operational guides and basic code templates can lower the learning difficulty and boost students' confidence. For intermediate students, teaching can focus on problem-solving methods and code optimization, cultivating their application skills through case studies, hands-on practice, and programming tasks, and guiding them to gradually understand more complex programming structures and logic.

For advanced students, more challenging project tasks, independent research projects, and higher-difficulty programming competition problems can be assigned to stimulate their innovation potential and practical abilities. This teaching content design encourages advanced students to master higher-level programming skills, such as the application of complex algorithms, flexible use of data structures, and optimization of code efficiency, enhancing their overall programming competence. Tiered instruction can also be enriched through group study, group discussions, and project collaboration to increase interactivity, offering students opportunities for communication and learning at various levels, thereby strengthening teamwork skills and learning motivation. Through such differentiated teaching, teachers can provide tailored guidance to each group of students, ensuring they make progress and improvement at an appropriate pace, thus achieving efficiency and personalization in programming education.

3.2 Feedback and Adjustment Mechanism for Teaching

To ensure the dynamic adaptability and continuous optimization of differentiated teaching, it is crucial to establish a timely and flexible feedback and adjustment mechanism. By tracking and providing feedback on students' programming exercises, project completion, code quality, and error analysis, teachers can gain a comprehensive understanding of students' learning progress and specific weaknesses. For instance, teachers can use automatically generated data reports from learning platforms to identify when students consistently struggle with particular knowledge points or specific programming tasks. When students exhibit high error rates or sustained difficulties with a particular concept, teachers can focus on explaining that content in the next session and provide personalized support materials, such as tutorial videos, additional exercises, or study guides, to help students deepen their understanding of the concept.

Moreover, a feedback mechanism based on learning data helps teachers identify common difficulties in teaching and adjust the content and methods accordingly. For example, if most students struggle with complex topics such as recursion or data structures, teachers can adjust the teaching schedule, simplify explanations, or include more examples and exercises to ensure the precision and effectiveness of differentiated teaching. This dynamic adjustment not only improves teaching effectiveness but also allows students to learn at a more suitable pace, enhancing their learning experience.[5]

The feedback mechanism should also include students' self-feedback, obtained through regular surveys, online feedback, and reflections on their learning experiences. Teachers can use this feedback to make flexible adjustments, such as altering the depth or pace of content, or providing more interactive opportunities, thereby further improving the adaptability of teaching methods and enhancing students' satisfaction with their learning experience. Establishing a comprehensive feedback and adjustment mechanism allows differentiated teaching to meet students' needs in real time, providing teachers with reliable data support to continuously optimize teaching strategies and achieve more effective Python programming instruction.

3.3 Diversified Design of Teaching Evaluation

A diversified design of teaching evaluation is an important aspect of assessing the effectiveness of the differentiated teaching model, reflecting students' growth and overall performance in Python programming learning. Traditional single assessment methods often focus on final exam scores, which do not comprehensively assess students' actual learning outcomes and skills application during the process. Therefore, constructing a multidimensional evaluation system is particularly important. This evaluation system should include formative assessment, summative assessment, and self-assessment to ensure comprehensive and objective evaluations.

In formative assessment, teachers can assess students' daily programming exercises, project task completion, debugging skills, and learning progress, providing real-time feedback on their learning effectiveness, helping students identify weaknesses and adjust their learning strategies accordingly. Formative assessment can also include periodic quizzes, tests, and project progress reports, offering a variety of assessment formats to accurately record students' efforts and progress.

Summative assessment should focus on evaluating students' actual programming abilities, such as their ability to independently write project code, implement algorithms, and optimize code proficiency, to assess their mastery of programming knowledge and application skills. Additionally, project presentations, code quality analysis, and programming tests can be used to comprehensively evaluate students' practical skills, ensuring the assessment results are realistic and application-oriented.[6]

Furthermore, self-assessment and peer assessment are important complements to diversified evaluation, encouraging students to reflect on themselves and provide feedback to their peers, enhancing their autonomy and collaboration. In self-assessment, students can regularly reflect on their learning progress, problem-solving abilities, and interest in programming, discovering and planning their future learning directions. Peer assessment provides opportunities for collaboration and communication, allowing students to share and receive feedback on their learning outcomes, broadening their programming perspectives and improving teamwork abilities.

By building such a multi-level, multidimensional evaluation system, teachers can more comprehensively and objectively assess the actual effectiveness of the differentiated teaching model. This not only helps students achieve better growth in programming learning but also provides strong data support for optimizing teaching strategies and advancing differentiated teaching.

Conclusion

This study constructs a differentiated teaching model for Python programming based on learning data analysis, emphasizing the accurate identification of students' learning needs through the collection and analysis of learning data, thereby providing personalized teaching support. The research results indicate that through the collaborative application of student stratification, personalized resource recommendations, and dynamic feedback mechanisms, students' programming skills and learning motivation have been significantly improved. This model breaks through the limitations of traditional teaching approaches and provides an effective path for differentiated education in programming instruction. Future research may focus on further optimizing learning data analysis tools to improve the accuracy and efficiency of the analysis, ensuring real-time feedback in teaching. Additionally, the applicability of this model in the teaching of other programming languages could be explored to expand the application of differentiated teaching models, promoting the deep development of personalized education in programming courses.

Fund Project

Gansu Province Education Science "14th Five-Year Plan" 2023 Annual Project, Project Title: Exploration and Application of Differentiated Teaching Model in Smart Education Environment —Taking Python Programming at Undergraduate Level in Vocational Colleges as an Example.

References

[1] Yang Hui, Wu Guanglin, Tian Zhenglin, et al. Analysis of Issues and Countermeasures in Python Programming Teaching [J]. Industry and Science Forum, 2024, 23(17): 145-147.
[2] Li Bin. Comparative Analysis and Outlook of Traditional and Modern Python Programming Teaching Methods [J]. Computer Knowledge and Technology, 2024, 20(23): 145-147.
[3] Li Fan, Lü Jia, Cheng Lijun, et al. Research on the Teaching Model of "Python Programming" Based on OBE Concept [J]. Science and Technology Information, 2024, 22(15): 207-210.
[4] Li Yanning, Liu Junwei, Zhou Tao. Analysis of Computer Programming and Teaching Models Based on Python Language [J]. Electronics, 2024, 53(07): 186-187.

[5] Zhang Cong. Research on Innovative Teaching Models in Vocational Mathematics Courses Based on Python [J]. Science and Technology Information, 2024, 22(13): 196-199.
[6] Liu Xiaoyu, Chen Yin. Exploration and Practice of the Reform Path for Python Programming Courses in Vocational Computer Majors [J]. Modern Vocational Education, 2024, (14): 84-87.