ISSN 2522-6398 Vol. 8, Issue 10: 127-132, DOI: 10.25236/FER.2025.081019

Learning Effectiveness Evaluation System Construction – Taking Python Programming as an Example

Xu Li[1,*], Mengyuan Jing[1]

[1]*School of Computer Science and Technology, Xinjiang Normal University, Urumqi, 830054, China*
[*]*Corresponding author:1165297698@qq.com*

Abstract: *Learning effectiveness evaluation is an important part of diagnosing problems in the teaching process and optimizing the precise delivery of teaching resources. Taking undergraduate students from the School of Computer Science at a certain university as the research object, this study aims to address prominent issues such as the teacher's "monologue" mode and excessive reliance on quantitative results in the teaching process of Python programming. Using mathematical models to identify process data characteristics, a learning effectiveness evaluation system is constructed, and a multi-dimensional learning effectiveness evaluation model with strong operability is established. At the same time, using the learning effectiveness evaluation model to conduct relevant empirical research, suggestions and measures for programming courses such as Python programming design are proposed, and the research results are attempted to be promoted to other classes in related majors in universities to enhance learners' practical programming abilities.*

Keywords: *Learning Outcome; Evaluation; Importance; Python Programming; Indicator System*

1. Introduction

Under information-rich conditions, fine-grained management and intelligent services constitute an exemplary pathway for teaching and learning, and simultaneously constitute a pivotal methodology for reforming learning-outcome assessment in the context of Emerging Engineering Education. From the learner's perspective, the systematic mining of process data enables a holistic diagnosis of individual learning profiles and a real-time enhancement of metacognitive awareness. From the educator's perspective, the construction of evidence-based course-assessment frameworks uncovers students' critical and core needs, thereby furnishing a rigorous empirical basis for the iterative re-design of curricular architectures.

Research conducted within the context of medical schools [1] explored issues such as curriculum design, teaching methodology reform, and assessment approaches. Addressing the prominent disconnect between theory and practice [2], an embedded hardware integration model was designed for programming courses. This model provides reference and insights for optimizing university programming curricula by evaluating goal attainment and practical skill mastery. Guided by engineering education accreditation principles[3], this study reconfigured course objectives and improved teaching models to explore mechanisms for cultivating students' engineering application capabilities, thereby enhancing their problem-solving abilities in engineering contexts. Tailored to specific teaching scenarios[4], this study investigates application strategies for large language models in programming instruction, specifying corresponding teaching interventions such as intelligent tutoring systems. As an innovative teaching model, the split-classroom approach addresses passive teaching methods in Python programming courses by breaking the teacher-centered "monologue" model. It proposes diversified assessment schemes and learning strategies leveraging digital technologies[5]. Addressing shortcomings in programming course delivery, this approach constructs a curriculum knowledge framework using knowledge graph entities and relationships [6], enabling knowledge reorganization and reconstruction. Knowledge graph technology is applied to programming course teaching practice. Tailored to programming course requirements [7], it selects programming cases and develops instructional experimental programs to build foundational experiments of various types. An application platform for practical systems is also constructed, providing a vital experimental environment for learners to conduct programming exercises.

ISSN 2522-6398 Vol. 8, Issue 10: 127-132, DOI: 10.25236/FER.2025.081019

The scientific rigor and effectiveness of the evaluation system are pivotal to assessing course objectives. Within the context of new engineering education development, the degree of course objective attainment serves as a key indicator for educational quality evaluation. By integrating dimensions such as training objectives and curriculum frameworks, this approach proposes a method for calculating multidimensional objective attainment levels. It facilitates timely teaching quality assessments and establishes a comprehensive teaching quality management plan. Addressing the primary factors causing learning difficulties in programming courses, this study proposes optimization strategies for programming course learning by refining learning pathways and constructing progressive knowledge state paths[8]. Object-oriented Java programming is a vital course for cultivating practical programming skills. Focusing on aspects such as course nature, positioning, and teaching objectives, this study proposes project-based learning as the main approach[9]. It reconstructs teaching module design, refines instructional content, and presents actionable teaching strategies. Research indicates that traditional teaching models can hinder students' mastery of programming skills. By leveraging online course resources, a blended teaching model is designed to propose strategies for programming course development. By analyzing key concepts in C++ such as functions, pointers, and loop statements, this study explores integration points for modules like template metaprogramming and function call optimization[10], proposing concrete methods to enhance C++ learners' application capabilities. Modular programming serves as a vital approach for decomposing complex systems into programmable units. Drawing from practical cases, this work introduces modular programming[11] and proposes optimization strategies to enhance code readability.

This study focuses on the implicit information within process data, such as learning styles and preferences, employing theoretical methods including dimension selection and data analysis. The process data referred to in this research encompasses data generated throughout all stages of the learning process that reflects learners' authentic behavioral performance in natural settings. Based on learner profiles, both fundamental static labels and dynamic learning labels are constructed, leading to the proposal of an evaluation metric system for programming courses.

2. Theoretical Foundation

This study focuses on undergraduate students in the School of Computer Science at a university. It sequentially constructs a learning effectiveness evaluation model for the group, analyzing the model's efficacy and learning outcomes from multiple dimensions including academic performance, characteristic expression, and the depth of attribute influence. The research delves into the impact of various learner attributes on learning effectiveness and examines how learning outcomes vary across individual factors such as majors and classes. It summarizes relevant issues encountered during the learning process and conducts differential analyses of learning effectiveness across different groups. Simultaneously, utilizing decision tables to screen effective and redundant features helps reduce cognitive load from massive datasets. This enables timely identification of critical needs and core requirements, pinpointing the foundation and direction for teaching efforts. It further supports the formulation and adjustment of instructional strategies.

A tuple $S=(U,AT,\{V_a \mid a\in AT\},\{I_a \mid a\in AT\})$ is called an information table[12,13], where U is a universal set and AT is a finite, non-empty set of attributes. If $AT=C\cup D$ and $C\cap D=\varnothing$, where C is the conditional attribute set and D is the decision attribute set, then the tuple S is called a decision table[14-16], denoted as $(U,C\cup D)$.

As an expression form, decision tables can describe objects based on human cognition and establish relationships between them, reflecting the essence of objective phenomena to a certain extent. They are increasingly becoming an important tool for solving complex problems. Building upon this foundation, different mathematical models can be employed to address practical issues in engineering applications and other fields according to varying objective requirements. Within decision tables, conditional attribute sets are primarily used to describe the characteristics possessed by objects. Key concepts such as attribute importance can characterize the significance of different features within the decision table and further describe the interrelationships between these features. Decision attribute sets primarily classify global sets through multi-label data.

Definition 1 Given a subset $A\subseteq AT$, the equivalence relation is defined as $R_A=\{(x,y)\mid (x,y)\in U\times U, I_a(x)=I_a(y),\forall a\in A\}$. The equivalence class of a property subset is denoted as

ISSN 2522-6398 Vol. 8, Issue 10: 127-132, DOI: 10.25236/FER.2025.081019

$[x]_A = \{y|(x,y) \in R_A\}$.

According to set theory, equivalence classes are determined by equivalence relations, which satisfy reflexivity, symmetry, and transitivity. Since learner process data encompasses multiple data types—such as symbolic data, continuous-value data, and others—it is generally necessary to process learner-related data through discretization methods to establish a foundation for data analysis.

Definition 2 Set $Pos_C D = \{x \mid [x]_C \subseteq [x]_D\}$, where $|Pos_C D|$ represents the cardinality of $Pos_C D$.

By Definition 2, when the attribute set grows larger, the corresponding $Pos_C D$ increases accordingly; conversely, when the attribute set grows larger, the corresponding $Pos_C D$ decreases accordingly. Definition 2 describes the set of objects that can be completely determined under the given conditional attribute set, characterizing classification capability from a set-theoretic perspective and employing the concept of equivalence classes to depict the certainty of object classification.

Definition 3: let $(U, C \cup D)$ be the decision table and the indicator importance be defined as follows, where $|\cdot|$ is the cardinality, $Sig_a(C,D) = \frac{|Pos_C D - Pos_{C-\{a\}} D|}{|U|}$.

Using Definition 3, weight analysis can be performed on each indicator within the evaluation metric system to identify effective indicators. Clearly, the greater the importance of an indicator, the more significant its role within the evaluation metric system. Leveraging the real-time characteristics of process data, attribute importance is calculated and an invariant matrix for attribute importance is established. To address model reliability issues caused by partial data inconsistencies, we attempt to break through by analyzing correlations between attributes. The focus is on mining clustering features such as learner profiles and learner group profiles to reconstruct the learner profiling model.

Existing models heavily rely on subjective experience when calculating evaluation metric weights. The process of constructing evaluation metric systems involves multiple subjective factors, such as using expert scoring methods to describe metric system weights. This approach lacks rationality in the evaluation metric construction process, leading to evaluation result biases to a certain extent. Based on data concerning different learner types, specific characteristics of the programming learning process, and evaluation outcomes, learner data is described through various elements within decision table structures—such as objects, attributes, and attribute values—to construct decision tables tailored to distinct learners.

3. Evaluation System Development

The evaluation system construction process primarily involves key stages such as defining evaluation objectives, data collection and cleaning, data analysis and processing, establishing mathematical models, and data output. Based on evaluation requirements and precise delivery needs, the construction objectives are clarified and requirements are determined. Activity data and sentiment data related to teaching content are collected from classroom teaching processes and question-answering records. The data collection and cleaning workflow is then implemented through data cleansing, integration, and transformation. Concurrently, decision table models are utilized to construct feature importance analysis, enhancing the stability of mathematical models.

The data collected by the institute includes learners' basic information such as age, gender, and learning background. It also encompasses behavioral data during the programming learning process (see Table 1), including code submission status, number of debugging attempts, online learning duration, and discussion participation frequency. Additionally, it captures learners' learning abilities and styles, such as logical thinking skills and problem-solving capabilities. Concurrently, process data—including instructional content and activity records, learner characteristics, and affective data—is gathered from classroom teaching videos, online platforms, and question-answering logs. Through data cleansing, integration, transformation, and reduction processes, the data format is standardized. Missing, erroneous, redundant, and uncertain data are filtered out, converting the dataset into a consistent dimensional format for enhanced comprehension and analysis. Learner tags are categorized into static tags and dynamic learning tags, establishing a tagging system for learner profiles. A simplified model calculates feature importance to identify key characteristics for learner profiling.

First, learner characteristics are preliminarily determined through comprehensive consideration of

ISSN 2522-6398 Vol. 8, Issue 10: 127-132, DOI: 10.25236/FER.2025.081019

general learner attributes, learning engagement, and motivation. Second, multimodal data is integrated to explore mapping relationships between learner characteristics and diverse assessment metrics—such as process-oriented data from peer-to-peer and teacher-student interactions, as well as learning styles—establishing correlations between learner traits and factors influencing learning outcomes. Third, features reflecting learner characteristics are extracted from aspects such as innovative thinking and problem-solving abilities, encompassing human-computer interaction, interpersonal interaction, and creative practice. Finally, construct an object set based on learner data and an attribute set based on analyzed characteristics to propose a learning effectiveness evaluation model grounded in rough set theory. Utilize this model to extract cross-factors between profile characteristics and learning outcomes, focusing particularly on student innovation thinking and problem-solving abilities. This establishes a highly operational, multi-dimensional learning effectiveness evaluation model.

Table 1 Overview of Data Dimensions and Evaluation Criteria for Python Programming Courses

Level Indicator	Sub-level Indicator	Evaluation Criteria	Evaluation Grade Selection
Teaching Dimension	Curriculum Content Analysis	1. Analyze teaching materials to clarify connections between this section and preceding/subsequent knowledge; 2. Align with course standards and accurately delineate knowledge points; 3. Identify key points and difficulties appropriate to learner levels with feasible measures.	Excellent/Good/Average/Poor
	Learner Profile Analysis	1. Accounts for individual differences (cognitive styles, intelligence, error factors); 2. Identifies existing knowledge/skill foundations and gaps; 3. Assesses current information literacy and attitudes.	Excellent/Good/Average/Poor
	Learning Objectives Analysis	1. Objectives are clearly stated, measurable, and use standardized terminology; 2. Includes scope and difficulty levels, emphasizing key points; 3. Reflects multidimensional values, emphasizes competency development.	Excellent/Good/Average/Poor
Learning Dimensions	Static Tags	Age, gender, learning background, logical thinking ability, problem-solving ability.	Excellent/Good/Average/Poor
	Dynamic Tags	Code submission frequency, debugging attempts, online duration, discussion participation, emotional tendencies.	Excellent/Good/Average/Poor
Process Dimensions	Teaching Content & Activities	Classroom teaching videos, online platform activities, Q&A records, assignment trajectories.	Excellent/Good/Average/Poor
	Data Quality Assurance	Cleaning of missing values, erroneous values, redundant values, and uncertain values; dimensional consistency.	Excellent/Good/Average/Poor
Outcome-Level Dimension	Learner Profiling	Reduced model feature importance, static + dynamic label profiling, learner labeling system.	Excellent/Good/Average/Poor

4. Programming Course Examples Programming Course Examples

Reviewing relevant domestic and international literature, this study selected a Python programming course at a university as a case study and collected experimental data for research. The curriculum covered 12 chapters including sequential structures, selection structures, and functions, with teaching processes involving classroom instruction, lab operations, and team time. A survey was conducted on 78 learners across two classes.

Taking dictionary learning issues as an example (Table 2), metadata from Learning Pass, competitive assessments, and virtual simulation platforms revealed: 97.43% mastery rate for everyday scenarios (name-phone number mapping), 94.81% for simple learning scenarios (student ID-grade mapping), yet only 89.74% for complex structured scenarios (nested dictionaries). Research revealed that learners' knowledge gaps primarily stemmed from: disorganized nested hierarchy design, inconsistent naming of nested keys, and high error rates in adding, deleting, or modifying nested data. After multiple learning cycles, the accuracy of key-value association logic improved from an initial 50% to 92.3% after reinforcement, further solidifying to 94.81%. Currently, 2.56% of students still lack proficiency in exception handling.

ISSN 2522-6398 Vol. 8, Issue 10: 127-132, DOI: 10.25236/FER.2025.081019

Table 2 Python Dictionary Learning Outcomes and Error Tracking (N=78)

Stage/Scenario	Key Metric	Mastery Rate (%)	Primary Error Manifestation	Remarks
Daily Scenario	Name-Phone Mapping	97.43	Key spelling/format errors	Initial Round
Basic Learning Scenario	Student ID-Grade Mapping	94.81	Misaligned single-level key-value pairs	Initial Round
Complex Structured Scenario	Nested Dictionary Mapping	89.74	(1)Chaotic nested hierarchy design (2)Inconsistent key naming (3)Errors in add/delete/modify operations	Initial Round
Multi-Round Learning	Key-Value Association Logic	50→92.3→94.81	Decreasing trends for (1)(2)(3) above	Reinforcement + Consolidation Rounds
Exception Handling Mastery	Correct Implementation	97.44	Missing except/finally blocks	End of Consolidation Round
Unmastered Students	Incomplete exception blocks	2.56	Failed to implement try-catch structure	End of Consolidation Round

Reviewing relevant domestic and international literature, this study selected a Python programming course at a university as a case study and collected experimental data for research. The curriculum covered 12 chapters including sequential structures, selection structures, and functions, with teaching processes involving classroom instruction, lab operations, and team time. A survey was conducted on 78 learners across two classes.

Taking dictionary learning issues as an example (Table 2), metadata from Learning Pass, competitive assessments, and virtual simulation platforms revealed: 97.43% mastery rate for everyday scenarios (name-phone number mapping), 94.81% for simple learning scenarios (student ID-grade mapping), yet only 89.74% for complex structured scenarios (nested dictionaries). Research revealed that learners' knowledge gaps primarily stemmed from: disorganized nested hierarchy design, inconsistent naming of nested keys, and high error rates in adding, deleting, or modifying nested data. After multiple learning cycles, the accuracy of key-value association logic improved from an initial 50% to 92.3% after reinforcement, further solidifying to 94.81%. Currently, 2.56% of students still lack proficiency in exception handling.

5. Conclusion

Learning effectiveness evaluation is an important part of diagnosing problems in the teaching process and optimizing the precise delivery of teaching resources. Actively integrate research findings into the practical teaching of programming courses, with the initial application expected in the formative assessment of programming courses offered by the college, such as Python and C programming. Based on usage outcomes and feedback, the existing model will undergo refinement and upgrades to progressively optimize the comprehensive assessment system, ensuring its practicality and effectiveness. Following successful application and validation of the model's efficacy, this approach will be extended to other classes within relevant university programs. By enhancing the learning effectiveness evaluation model, we aim to further stimulate students' proactive engagement in learning. This will encourage every student to actively participate in all learning stages, fully integrate into the entire learning process, continuously improve learning outcomes, and strengthen learners' practical programming skills.

Acknowledgements

This work was supported by the Teaching Development Project of Xinjiang Normal University (Grant NO.SDJG2024-25).

References

[1] Fu XiaoXue, Li Xiang. Teaching reform in programming language courses in medical colleges——

ISSN 2522-6398 Vol. 8, Issue 10: 127-132, DOI: 10.25236/FER.2025.081019

taking C language programming as an example [J/OL].Medical Education Management,1-9[2025-09-02].
[2] Zhu Lixin, Li Xin, Su Jinchao. A Study on the Design and Application of Computer Network Programming Course[J].Digital Education,2025,11(03):77-85.
[3] Wang Changyuan, Zhang Yi, Ren Chunhua. Curriculum Teaching Reform and Practice Based on Engineering Education Philosophy: The Case of Linux Programming and Applications Course [J].Electronic Components and Information Technology,2025,9(06):249-251.
[4] Wang Dong. Strategies and Application of Large Language Model in the Teaching of Computer Programming Practice Courses[J].Information & Computer,2025,37(11):239-241.
[5] Yao Jiahui. Exploration of High School Python Teaching Strategies Based on Split Classroom in Digital Environment[J].Computer Knowledge and Technology,2025,21(03):168-170+173.
[6] Wang Kemeng. Research on the Application of Knowledge Graph Technology in the Teaching of "Object-Oriented Programming and Design" Course[J].Science & Technology Information, 2025, 23(09): 206-209.
[7] Chen Jia, Wei Lanqi, Liu Lianzhong, et al. Construction of Practice Teaching Platform for Secure Python Programming[J/OL].Software Guide,1-7[2025-09-02].
[8] Wang Xiaochun, Gu Xiaoqing, Liu Wen. Obstacles and Countermeasures in Learning General Programming Courses in Universities[J].China University Teaching,2024,(07):45-51+96.
[9] Xu Lifeng, Ding Weilong. Course Design of Object Oriented Java Programming[J].The Theory and Practice of Innovation and Entrepreneurship,2024,7(11):28-34+71.
[10] Wang Juan. Research on Computer Software Programming Based on C++Language[J]. Information Recording Materials,2025,26(05):157-159+240.DOI:10.16009/j.cnki.cn13-1295/tq. 2025. 05. 072.
[11] Zhou Kai, Huang Weifan. Scientific and Technological Innovation[J].Scientific and Technological Innovation,2025,(08):68-72.
[12] Li Xu, Tang J.G., Tang J.Y.. Local Indiscernibility Relation Reduction for Information Tables[J]. IEEE Access, 2022, 10: 78588-78596.
[13] Li Xu, Xiao H.P., Tang J.G., Zhu J.. New Variable Precision Reduction Algorithm for Decision Tables[J]. IEEE Access, 2023.10:42201-20712.
[14] Shangzhi WU , Ren Y , Shuyue G E ,et al.An attribute reduction algorithm of weighting neighborhood rough sets with Critic method[J].Journal of Beijing University of Aeronautics and Astronautics, 2025, 51(1):75-84.
[15] Xie, L.; Lin, G.; Li, J.; Lin, Y. A novel fuzzy-rough attribute reduction approach via local information entropy[J]. Fuzzy Sets and Systems, 2023, 473, 108733.
[16] Wang H, Zhi H, Li Y, et al. A Generalized Multigranulation Rough Set Model by Synthesizing Optimistic and Pessimistic Attitude Preferences[J]. Mathematics, 2025, 13(9): 1367.