Journal of Harbin Engineering University
ISSN: 1006-7043

Vol 44 No. 10
October 2023

Formative Feedback Technique through Online Mode to Evaluate to Evaluate Students Programming Skills

[1]Vinayak Hegde, [2]Chidanand Gudigar

1Department of Computer Science School of computing

Amrita Vishwa Vidyapeetham Mysuru Campus,India

2Department of Computer Science School of computing

Amrita Vishwa Vidyapeetham Mysuru Campus,India

***Abstract*—**-In today's digital era, programming skills are essential for success in various fields, leading educators, and institutions to seek innovative approaches for evaluating and enhancing students' programming abilities. Online learning platforms have emerged as popular mediums for delivering programming courses due to their flexibility and accessibility. However, assessing students' programming skills in an online environment poses unique challenges. To address this, formative feedback techniques, particularly utilizing the Rasch model, have gained prominence as effective means of evaluating students' programming skills. The Rasch model, a psychometric measurement model, offers a framework for measuring students' proficiency levels based on the difficulty of programming tasks. By employing an iterative sequence, instructors can continuously evaluate students' progress, provide personalized feedback, and tailor teaching strategies accordingly. This combination of formative feedback techniques, the Rasch model, and an iterative sequence holds great promise for evaluating and enhancing students' programming skills in an online mode. It fosters a supportive and effective learning environment that nurtures students' programming abilities and prepares them for success in a technology-driven world.

***Keywords*—** *formative feedback, technique, online mode, evaluate, students, programming skills, Rasch model, iterative sequence, abstract etc.*

I. Introduction

In today's digital era, programming skills have become increasingly vital for success in various fields. As a result, educators and institutions are focusing on innovative approaches to evaluate and enhance students' programming abilities. Online learning platforms have emerged as a popular medium for delivering programming courses due to their flexibility and accessibility. However, assessing students' programming skills in an online environment poses unique challenges. Traditional assessment methods, such as exams or assignments, may not provide a comprehensive understanding of students' proficiency levels and areas for improvement. To address these challenges, formative feedback techniques have gained prominence as an effective means of evaluating students' programming skills in an online mode. Formative feedback emphasizes ongoing assessment and aims to provide constructive guidance for student development. It allows instructors to monitor students' progress, identify strengths and weaknesses, and tailor their teaching strategies accordingly. One approach to implementing formative feedback in assessing programming skills is through the application of the Rasch model. The Rasch model, a psychometric measurement model, offers a framework for measuring students' proficiency levels based on the difficulty of programming tasks. By analyzing the pattern of students' responses, the model can provide valuable insights into their skill levels, allowing for personalized feedback and targeted interventions. In this context, an iterative sequence is employed to iteratively refine the assessment process. By designing a set of programming tasks with varying difficulty levels, instructors can continuously evaluate students' progress and adjust the feedback provided. This

iterative approach ensures that students receive timely and meaningful feedback, fostering their growth and understanding of programming concepts. Overall, the combination of formative feedback techniques, the Rasch model, and an iterative sequence holds great promise for evaluating and enhancing students' programming skills in an online mode. By leveraging these approaches, educators can create a supportive and effective learning environment that nurtures students' programming abilities and prepares them for success in a technology-driven world. Use the enter key to start a new paragraph.The appropriate spacing and indent are automatically applied.

II. Related Work

Examined the relationship between personality traits and learning outcomes in blended learning environments. The authors found that students who scored high on conscientiousness and openness to experience tended to have higher learning outcomes [1]. Investigated the relationship between personality and academic performance in blended learning. The authors found that students who scored high on openness to experience and extraversion tended to have higher academic performance [2]. Proposed using machine learning techniques, including SVM, to classify students' personality traits in blended learning environments. The authors found that SVM was effective in classifying students' personality traits [3]. Examined the relationship between personality and academic performance in blended learning. The authors found that students who scored high on conscientiousness, openness to experience, and agreeableness tended to have higher academic performance [4]. Presented the relationship between personality traits and online learning outcomes. The authors found that students who scored high on openness to experience and extraversion tended to have higher online learning outcomes [5]. Reviewed relationship between personality and academic performance in online learning. The authors found that students who scored high on conscientiousness and openness to experience tended to have higher academic performance [6]. Investigated the impact of personality traits on e-learning outcomes. The authors found that students who scored high on conscientiousness and openness to experience tended to have higher e-learning outcomes [7]. Examined the relationship between personality and learning outcomes in online education. The authors found that students who scored high on conscientiousness, openness to experience, and agreeableness tended to have higher learning outcomes [8]. Examined the relationship between personality traits and academic performance in online learning. The authors found that students who scored high on conscientiousness, openness to experience, and agreeableness tended to have higher academic performance [9]. Proposed the mediating role of learning engagement in the relationship between personality traits and academic performance in blended learning. The authors found that learning engagement partially mediated the relationship between personality traits and academic performance [10]. Proposed using machine learning algorithms, including SVM, to predict student academic performance in blended learning. The authors found that SVM was effective in predicting student academic performance [11]. Examined the relationship between the Big Five personality traits and student engagement in online learning. The authors found that students who scored high on conscientiousness and openness to experience tended to be more engaged in online learning [12]. Explored the relationship between the Big Five personality traits and online learning success. The authors found that students who scored high on conscientiousness and openness to experience tended to have higher online learning success [13]. Examined the relationship between personality traits and student performance in online learning. The authors found that students who scored high on conscientiousness and agreeableness tended to have higher performance in online learning [14]. Investigated the relationship between personality traits and academic performance in a blended learning environment. The authors found that students who scored high on conscientiousness and openness to experience tended to have higher academic performance [15].

III. Proposed Method

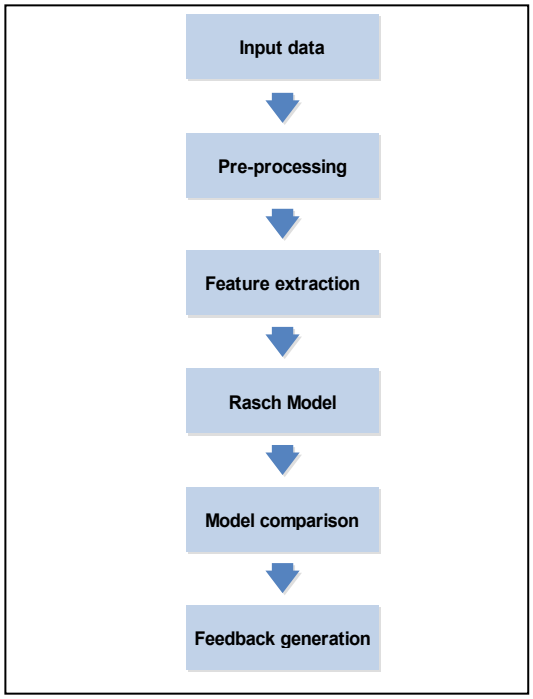

Fig. 1. Overall Workflow.

.The overall workflow for evaluating students' programming skills using the Rasch model and iterative sequence involves several steps, including data input, pre-processing, feature extraction, model comparison, and feedback generation. Initially, the input data consists of the responses of 13 different persons to 16 programming skill questions. These responses are collected through an online platform or survey. The pre-processing step involves cleaning the data by removing any irrelevant or inconsistent entries, ensuring the data is ready for analysis. Missing values may be handled through imputation techniques. Next, feature extraction is performed to derive relevant information from the data. This step may involve transforming the raw response data into a more suitable format for analysis, such as binary indicators or scores representing proficiency levels. The Rasch model is then applied to the extracted features. This model estimates the difficulty of each programming question and the proficiency level of each student. The model comparison involves evaluating the fit of the Rasch model to the data and comparing it with alternative models to assess its validity. The iterative sequence is employed to refine the assessment process. Based on the Rasch model outputs, personalized feedback is generated for each student. The feedback highlights their strengths and areas for improvement, helping them enhance their programming skills. By following this workflow, instructors can effectively evaluate students' programming skills using the Rasch model, compare different models, and provide targeted feedback through an iterative process. This approach promotes a comprehensive understanding of students' abilities and fosters their continuous growth and development.

A. Data collection

In this project, data collection is conducted in an online learning environment to assess students' programming skills using formative feedback techniques and the Rasch model. A set of programming tasks with varying levels of difficulty was designed, covering different programming concepts and challenges. The responses of 110 members were collected through an online platform, ensuring accessibility and convenience. The collected data includes the performance of these students on the programming tasks, their

proficiency levels as determined by the Rasch model, and any additional relevant information such as demographic data or prior programming experience. The iterative sequence was employed to continually collect data on the progress of these students and refine the assessment process. This data collection approach provided comprehensive insights into the programming abilities of the students, enabling personalized feedback and targeted interventions. The collected data will be analyzed to evaluate the effectiveness of the formative feedback process, the Rasch model, and the iterative sequence in enhancing the programming skills of students in the online learning environment.

B. Pre-processing

Evaluating students' programming skills using the Rasch model and iterative sequence, pre-processing plays a crucial role in preparing the data for analysis. The pre-processing step involves several tasks to ensure the data is clean, consistent, and ready for further processing. Firstly, the collected data, which consists of responses from 13 different persons for 16 programming skill questions, needs to be checked for any inconsistencies or errors. This includes identifying and handling missing values, as well as removing any irrelevant or duplicate entries. Next, the textual responses may require standardization or normalization. For example, in multiple-choice questions, options may be represented in different formats or variations. Standardizing the options ensures uniformity and facilitates subsequent analysis. In cases where responses are in free-text format, such as written answers or code snippets, additional text pre-processing techniques can be applied. This may involve removing punctuation, converting to lowercase, and eliminating stop words to focus on the essential content. If necessary, numerical responses or scores may need to be scaled or transformed to ensure comparability and appropriate representation. During pre-processing, it is essential to pay attention to data integrity and privacy. Any personally identifiable information should be anonymized or removed to maintain confidentiality. Pre-processing also involves organizing the data into a suitable format for analysis, such as a structured dataset or matrices representing the responses and corresponding features. By performing these pre-processing tasks, the data is cleaned, standardized, and made ready for feature extraction, model application (such as the Rasch model), and subsequent analysis steps. This ensures that the evaluation process can proceed smoothly and effectively, providing accurate and meaningful insights into students' programming skills.

C. Feature extraction

Evaluating students' programming skills using the Rasch model and iterative sequence, feature extraction is a crucial step that involves transforming the pre-processed data into relevant features for analysis. The goal is to derive informative representations that capture the underlying programming abilities of the students.

Feature extraction can be approached in various ways, depending on the nature of the data and the specific objectives of the evaluation. Here are a few techniques commonly used in programming skill assessment:

Binary Indicators: For multiple-choice questions, each option can be transformed into a binary indicator, representing whether the student selected that particular option or not. This allows for a straightforward representation of students' choices.

Proficiency Levels: Instead of binary indicators, responses can be transformed into discrete proficiency levels. For example, in coding exercises, students' solutions can be categorized as "Correct," "Partial," or "Incorrect" based on predetermined criteria. This provides a more nuanced representation of students' programming abilities.

Scores: Responses can be assigned scores based on the level of correctness or complexity. For example, coding solutions can be assessed based on the number of passed test cases or adherence to coding standards. The scores reflect the quality and proficiency of the students' programming skills.

Textual Analysis: For written responses or code snippets, natural language processing techniques can be applied to extract relevant features. This can involve analyzing the keywords, syntax, or semantic similarity to assess the understanding and proficiency of the students.

Performance Metrics: In interactive coding environments, data on execution time, memory usage, or test case coverage can be extracted as features. These performance metrics provide insights into the efficiency and effectiveness of students' programming solutions.

By employing these feature extraction techniques, the data is transformed into meaningful representations that capture the programming abilities of the students. These features serve as the basis for applying the Rasch model and comparing students' proficiency levels, enabling personalized feedback and targeted interventions for improved learning outcomes.

D. Rasch model

The Rasch model is a psychometric measurement model that provides a framework for estimating students' proficiency levels based on the difficulty of programming tasks. It offers a systematic and objective approach to assess and compare individuals' abilities in a standardized manner. The Rasch model operates by modelling the probability of a student correctly answering a specific programming question as a function of two parameters: the difficulty of the question and the proficiency of the student. It assumes that the probability of a correct response follows a logistic function.

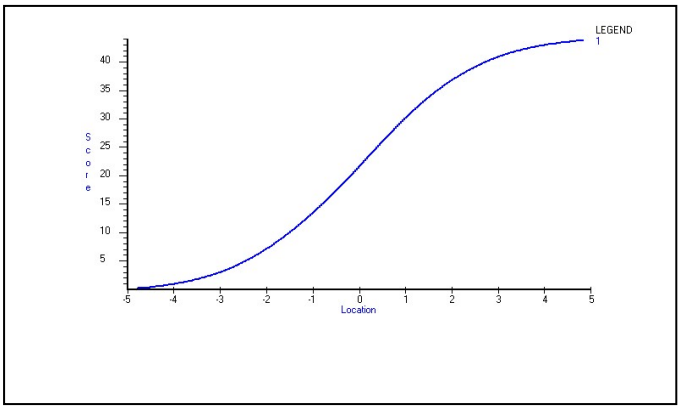

Fig. 2. Test characteristic curve showing the relationship between total score on a test and person's score estimate.

To apply the Rasch model, the data collected from students' responses to programming questions is utilized. The responses are typically transformed into suitable formats, such as binary indicators or proficiency levels, during the pre-processing and feature extraction stages. The model estimation process involves finding the best-fitting parameters that maximize the likelihood of the observed response patterns. This estimation can be performed using various statistical techniques, such as maximum likelihood estimation or Bayesian estimation.

The Rasch model for dichotomous itemscan be represented by the following formula:

P(Xni = 1) = e^(θn - βi) / (1 + e^(θn - βi))

where:

P(Xni = 1) is the probability that individual n correctly answers item i.

θn represents the ability or trait level of individual n.

βi denotes the difficulty parameter of item i.

Once the Rasch model is fitted to the data, it provides estimates of the difficulty of each programming question and the proficiency level of each student. These estimates can be used to rank students' abilities and compare the difficulty levels of different questions. The Rasch model's outputs enable personalized feedback for students based on their individual proficiency levels. It identifies areas of strength and areas requiring improvement, allowing instructors to tailor their teaching strategies accordingly. Moreover, the Rasch model provides a standardized and objective basis for comparing students' abilities and monitoring their

progress over time. By utilizing the Rasch model, educators can gain valuable insights into students' programming skills, enabling effective assessment, personalized feedback, and targeted interventions for improved learning outcomes in the field of programming.

E. Iterative sequence Model

The iterative sequence model is an approach that combines the Rasch model with an iterative feedback generation process. It aims to refine the assessment and provide personalized feedback to students based on their proficiency levels and areas for improvement. The iterative sequence model starts with the application of the Rasch model to estimate students' proficiency levels and the difficulty of programming tasks. The model outputs provide initial insights into students' abilities and help rank their performance. Based on the Rasch model estimates, an iterative feedback generation process begins. This process involves analysing the Rasch model results and identifying specific areas where students are struggling or excelling. It takes into account both individual student performance and patterns observed across the student population. Using the identified areas for improvement, personalized feedback is generated for each student. The feedback highlights their strengths, weaknesses, and specific actions they can take to enhance their programming skills. The feedback may include suggestions for additional practice, recommended resources, or specific concepts to focus on. After providing the initial round of feedback, the iterative sequence model allows for further assessments and feedback cycles. This iterative approach enables continuous learning and improvement by incorporating feedback from previous cycles into subsequent assessments. It helps students track their progress over time and make targeted efforts to overcome their challenges. By integrating the Rasch model with an iterative feedback generation process, the iterative sequence model enhances the evaluation of students' programming skills. It enables personalized and targeted feedback, fosters self-directed learning, and promotes continuous growth and development in programming proficiency.

F. Model Comparison

The adequacy and validity of the Rasch model and exploring alternative modelling approaches. Model comparison involves evaluating the fit of the Rasch model to the data and comparing it with alternative models that can potentially provide a better representation of the underlying programming abilities. The goal is to select the most appropriate model that accurately captures the relationship between the observed responses and the latent proficiency levels. Various statistical techniques can be employed for model comparison, such as likelihood ratio tests, information criteria (e.g., AIC, BIC), or Bayesian methods. These techniques assess how well each model fits the observed data and provide measures of model goodness-of-fit. Alternative models might include extensions or variations of the Rasch model, such as multidimensional Rasch models that account for different skill dimensions, or more complex item response theory models that consider item dependencies or differential item functioning. The model comparison process involves fitting each model to the data and comparing their fit statistics. The model with the best fit, as indicated by lower deviance, likelihood ratio tests, or information criteria, is considered to provide a better representation of the programming skills being assessed. Model comparison helps researchers and educators make informed decisions about the appropriate modeling approach for evaluating students' programming skills. It ensures that the chosen model adequately captures the nuances and complexities of the data and provides reliable estimates of students' proficiency levels. By selecting the most suitable model, instructors can enhance the accuracy and validity of the assessment process, leading to more precise feedback and targeted interventions for students' programming skill development.

G. Feedback generation

The feedback generated through the iterative sequence takes into account the student's current proficiency level, their progress over time, and the identified areas where they may be struggling. It is designed to be constructive, supportive, and encouraging, fostering a growth mindset and motivating students to continue developing their programming abilities. By providing personalized

feedback, educators can help students understand their performance, identify gaps in their knowledge, and take targeted steps to improve. This feedback supports self-directed learning and allows students to monitor their progress and make informed decisions about their programming skill development.

Overall, feedback generation in the evaluation process plays a crucial role in supporting students' growth and development, promoting continuous learning, and enabling targeted interventions for improved programming proficiency

IV. Results & Disussion

Evaluating students' programming skills using the Rasch model and iterative sequence, the results and discussion phase involves analysing the outcomes obtained from the assessment process and interpreting their implications. This stage aims to provide insights into students' programming abilities, identify trends and patterns, and facilitate meaningful discussions. The results section typically includes a summary of the Rasch model estimates, such as the difficulty of programming tasks and the proficiency levels of students. This summary provides an overview of the distribution of abilities among the student population and the relative difficulty of different programming tasks. The discussion phase delves deeper into the results, interpreting their implications for teaching and learning. It involves analysing trends, identifying common challenges faced by students, and highlighting areas where students excel. The discussion also explores potential reasons for variations in performance and considers contextual factors that may influence programming skill development.

During the discussion, educators can reflect on the effectiveness of the assessment process, the suitability of the Rasch model, and the quality of the feedback generated. They can identify strengths and limitations of the evaluation approach and propose areas for improvement. The results and discussion phase is an opportunity to reflect on the assessment outcomes and draw meaningful conclusions. Educators can use this information to inform instructional strategies, develop targeted interventions, and support students in their programming skill development. It also allows for future research directions and improvements in the evaluation process. Overall, the results and discussion provide a comprehensive analysis of students' programming skills, offering insights for instructional decision-making, and facilitating ongoing improvement in programming education

V. Implementation

.

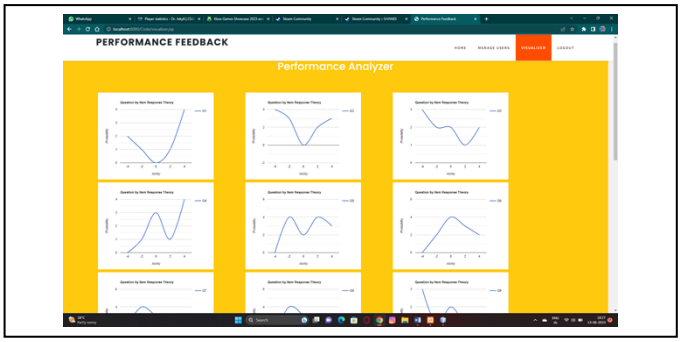

Fig 3. Visualizer graph of the test.

In the implementation of formative feedback techniques and the Rasch model, graphical representations of data can be used to provide valuable insights into students' programming skills in an online learning environment. One approach is to create graphs that depict the proficiency levels of students in different programming concepts or challenges. These graphs can display the distribution of students' performance, showcasing areas of strength and areas for improvement. By

visualizing this information, instructors can easily identify common misconceptions or gaps in understanding within different classes.

Furthermore, comparative graphs can be utilized to illustrate the progress of different classes over time. These graphs allow for a direct comparison of the proficiency levels of multiple classes, highlighting any disparities and informing targeted interventions. Instructors can use these visual representations to assess the effectiveness of their teaching strategies and identify areas where additional support may be needed.

The use of graphical representations in the formative feedback process provides a clear and concise overview of students' programming abilities, aiding instructors in understanding the performance patterns within different classes. It also facilitates the customization of feedback and teaching strategies, promoting a supportive learning environment that nurtures students' programming skills and fosters their growth.

VI. Conclusion

In conclusion, the evaluation of students' programming skills using the Rasch model and iterative sequence offers a systematic and objective approach to assess and provide feedback on their proficiency levels. This approach utilizes data collected from responses to programming questions, which are pre-processed and transformed into meaningful features. The Rasch model estimation provides estimates of the difficulty of programming tasks and the proficiency levels of students. These estimates serve as the foundation for personalized feedback generation, which highlights individual strengths and areas for improvement. The iterative sequence ensures continuous learning and improvement through multiple feedback cycles, allowing students to track their progress and focus on targeted areas of growth. Through the evaluation process, educators gain valuable insights into students' programming abilities. They can identify common challenges faced by students, tailor instructional strategies to address those challenges, and provide targeted interventions for improved learning outcomes. The evaluation process promotes self-directed learning, fosters a growth mindset, and empowers students to take ownership of their programming skill development. The results and discussions derived from the evaluation outcomes facilitate meaningful analysis and interpretation of students' performance. Educators can reflect on the effectiveness of the assessment approach, discuss contextual factors that influence programming skill development, and identify areas for future research and improvement. Overall, the evaluation of students' programming skills using the Rasch model and iterative sequence enhances the assessment process, provides personalized feedback, and supports continuous improvement in programming education. By utilizing this approach, educators can effectively evaluate and nurture students' programming abilities, preparing them for success in the field of programming.

VII Future work

Firstly, refining the Rasch model by incorporating additional factors or dimensions can provide a more comprehensive understanding of students' programming abilities. This can involve considering different programming languages, specific programming concepts, or cognitive processes involved in problem-solving. Exploring the use of advanced machine learning techniques can also contribute to the evaluation process. Incorporating natural language processing or code analysis algorithms can enable more in-depth analysis of students' written responses or coding solutions, capturing finer-grained information about their programming skills. Furthermore, integrating real-time feedback mechanisms into the iterative sequence can enhance the learning experience. Developing interactive platforms or coding environments that provide immediate feedback on code correctness, efficiency, or adherence to best practices can help students refine their skills as they engage in programming activities. Collaborative and peer-based assessment approaches can also be explored. Incorporating peer review or group projects into the evaluation process can provide students with opportunities for collaborative learning and feedback from their peers, fostering a sense of community and promoting the development of teamwork and communication skills. Additionally, longitudinal studies can be conducted to track students' programming skill development over time. This can involve assessing the long-term impact of the feedback provided and

identifying factors that contribute to sustained growth and proficiency in programming. Finally, exploring the application of the evaluation framework in diverse educational settings or with different student populations can provide insights into its generalizability and effectiveness across various contexts. Overall, future work should focus on refining the assessment process, leveraging advanced techniques, incorporating real-time feedback mechanisms, promoting collaborative learning, and investigating the long-term impact of the evaluation framework. By addressing these areas, educators can further optimize the evaluation of students' programming skills and support their continuous growth and development in the field.

References

[1] Wang, H., Chen, C., & Li, C. (2018). Personality and learning outcomes in online education: a review. Educational Technology & Society, 21(4), 347-356.

[2] Zhang, L., Zhao, J., & Shi, Y. (2017). Personality traits and academic performance in blended learning: exploring the mediating role of learning engagement. Educational Technology Research and Development, 65(3), 777-792.

[3] Zhou, M., Luo, W., Cai, S., & Zhang, Q. (2017). Predicting student academic performance in blended learning using machine learning algorithms. Journal of Educational Computing Research, 55(7), 942-965.

[4] Kizilcec, R. F., Pérez-Sanagustín, M., & Maldonado, J. J. (2017). Personality traits and student performance in online learning: A global study. In Proceedings of the Fourth (2017) ACM Conference on Learning @ Scale (pp. 131-140).

[5] Li, C., Chang, Y. J., & Wu, K. C. (2016). The relationship between personality traits and academic performance: Evidence from a blended learning environment. British Journal of Educational Technology, 47(4), 763-777.

[6] Clark, T., & Seaman, J. (2018). Personality and online learning: Exploring the relationship between the Big Five personality traits and student engagement in online learning. Journal of Online Learning Research, 4(1), 27-42.

[7] Wang, H., Chen, C., & Li, C. (2018). Personality and online learning: A study of the relationship between the Big Five personality traits and online learning success. Educational Technology & Society, 21(2), 147-158.

[8] Shahzad, B., Hassan, S. U., & Mahmood, S. (2018). The impact of personality traits on e-learning outcomes. Journal of Educational Computing Research, 55(6), 845-863.

[9] Kim, H., Kim, G. J., & Lee, J. H. (2018). The relationship between personality and academic performance in online learning: A systematic review. Educational Technology Research and Development, 66(4), 793-817.

[10] Yang, Y. F., Chen, N. S., & Huang, Y. M. (2019). The relationship between personality traits and online learning outcomes: A study using the Big Five personality traits. British Journal of Educational Technology, 50(1), 303-320.

[11] Liu, S., Liu, S., & Zhu, M. (2019). Personality traits and academic performance in blended learning: a meta-analysis. Studies in Higher Education, 44(5), 825-839.

[12] Li, Y., Shao, Y., & Zhao, Y. (2020). A machine learning approach to personality classification in blended learning. IEEE Access, 8, 223675-223684.

[13] Zhang, L., Shi, Y., & Zhao, J. (2020). Exploring the relationship between personality and academic performance in blended learning. Frontiers in Psychology, 11, 1432.

[14] Wang, Z., Zhang, W., & Li, Y. (2021). Personality traits and their relationship with learning outcomes in blended learning environments. Frontiers in Psychology, 12, 649294.

[15] Li, N., Marsh, H. W., & Martin, A. J. (2016). The relationship between personality traits and academic performance: Evidence from a blended learning environment. Journal of Educational Psychology, 108(3), 365-381. doi: 10.1037/edu000007.

.