Machine Learning-Enabled Personalization of Programming Learning Feedback

Mohammad T. Alshammari
College of Computer Science and Engineering, University of Ha'il, Ha'il, Saudi Arabia

***Abstract*—Acquiring programming skills is daunting for most learners and is even more challenging in heavily attended courses. This complexity also makes it difficult to offer personalized feedback within the time constraints of instructors. This study offers an approach to predict programming weaknesses in each learner to provide appropriate learning resources based on machine learning. The machine learning models selected for training and testing and then compared are Random Forest, Logistic Regression, Support Vector Machine, and Decision Trees. During the comparison based on the features of prior knowledge, time spent, and GPA, Logistic Regression was found to be the most accurate. Using this model, the programming weaknesses of each learner are identified so that personalized feedback can be given. The paper further describes a controlled experiment to evaluate the effectiveness of the personalized programming feedback generated based on the model. The findings indicate that learners receiving personalized programming feedback achieve superior learning outcomes than those receiving traditional feedback. The implications of these findings are explored further, and a direction for future research is suggested.**

***Keywords*—Machine learning; programming; learning outcome; feedback; personalization**

I. Introduction

The advances in Artificial Intelligence (AI) and its subfield, Machine Learning (ML), are rapidly transforming the education landscape, offering innovative approaches to improve teaching methods, accelerate learning, and deliver personalized educational experiences. AI-powered tools, including adaptive learning systems, intelligent tutoring platforms, and predictive analytics, are revolutionizing the traditional pedagogical landscape by generating personalized feedback, suggesting individualized learning pathways, and forecasting academic achievement [1], [2]. These technologies serve the diverse interests and needs of learners and support data-driven interventions, thereby enhancing the levels of engagement and retention both in traditional and online learning environments [3].

An example of this can be seen in adaptive learning systems that use AI to modify the material and pace of learning based on how each learner performs, resulting in a highly personalized educational experience [4]. Another AI application is Intelligent Tutoring Systems (ITSs), which can improve personalization by simulating one-on-one teaching and using advanced algorithms to analyze how a learner behaves and provide personalized guidance based on those metrics. Learner interaction data can also be employed as predictive analytics, which fully utilizes advanced ML models to analyze historical and real-time data, predict learners' outcomes, identify at-risk learners, and allow for considerable strategic resource provision. ML can also guide evidence-based decision-making, enabling educators to adjust curricula and instructional strategies by appropriately identifying learner behavior and patterns during learner-content interaction [5].

Programming is one of the most common subjects taught in a computer science curriculum. Still, it introduces several challenges, including mapping abstract concepts with practical activities and high dropout rates in introductory courses [6]. AI and ML in education can help mitigate these challenges by creating dynamic and tailored learning experiences. AI-based educational systems, for example, may analyze learner-system interaction data to develop personalized learning pathways, adjusting how content is delivered to suit learners' needs [7], [8].

Many ML models have been developed. For example, neural networks can be utilized to create learning recommendations that recommend personalized learning activities to keep learners engaged and retain knowledge [9]. Other ML models, like fuzzy logic, have also been used to offer personalized feedback and modify instructional approaches according to learners' characteristics, such as knowledge level and task accomplishments [10]. Recent approaches view chatbot-assisted programming systems powered by ML techniques as powerful learning tools that aid debugging, explain code errors, and suggest possible solutions, thereby serving as a proxy for a human tutor [11].

A vast majority of ML algorithms have been employed to predict programming learning performance; some of these are Decision trees (DT), Random Forests (RF), Support Vector Machines (SVM), Logistic Regression (LR), k-nearest Neighbors (kNN), and Artificial Neural Networks (ANN) [3], [12], [13], [14]. These models leverage learner engagement patterns, prior learning activities, and coding behavior to provide learners with timely intervention [15].

ML-enabled behavioral analytics tools can track and analyze learner engagement with resources such as coding environments, learning management systems, online coding platforms, and discussion forums. The analyses can reveal meaningful relationships between activities like active coding and passive lecture reviewing that may impact overall learning. Thus, actionable insights drive the curriculum design, define effective intervention strategies that address individual learning needs, and optimize the effectiveness of programming educational outcomes.

A key limitation in existing ML model-based investigations for programming education is the lack of a comprehensive approach that spans from data collection to evaluation in authentic learning environments [8], [16], [17]. While prior research has explored ML applications in programming learning, many studies focus on isolated aspects, such as predictive modeling or feedback generation, without fully integrating these components into a structured learning framework. This research gap highlights the need for further investigation into how ML models can be systematically leveraged to enhance personalized learning experiences. To address this, the present study conducts a comparative analysis of ML models to identify programming learning weaknesses based on learners' prior knowledge, time spent, and Grade Point Average (GPA). Identifying the most effective ML model can facilitate the delivery of personalized programming feedback tailored to individual learners' needs. Additionally, an experimental evaluation is implemented to explore the impact of personalized feedback on learning performance. This dual approach enhances the theoretical understanding of ML-driven feedback mechanisms and provides empirical evidence to guide future advancements in AI-supported programming education.

The key research questions are as follows:

RQ1. How can machine learning be used to predict specific weaknesses in learners' programming skills, enabling feedback personalization?

RQ2. Can personalized feedback derived from machine learning predictions enhance learning outcomes?

This paper is structured as follows: Section II presents related work. Section III outlines the methodology used in this study. Section IV offers the results. Section V discusses the findings, and Section VI concludes the paper.

II. Related Work

The recent reviews on integrating AI and ML in online learning platforms highlight their essential role in personalizing learning through customized content delivery for individual learners [1], [12], [13], [14]. The dynamic adaptation and targeted interventions through ML techniques (e.g., clustering, reinforcement learning, deep learning) are promising to enhance learner engagement, retention, and academic performance [18]. Yet, data privacy concerns, high resource requirements, and the risk of diminishing human interaction limit the widespread adoption. Careful implementation and ongoing refinement would be critical to mitigate these challenges and realize the potential benefits of more mainstream AI-based online learning platforms.

A predictive analysis of learner performance in programming tutoring comparing different ML models with ANN on dataset input finds that ANN outperforms other methodologies such as SVM, DT, and kNN [19]. However, the study's limitations include its targeting of a specific cohort of Norwegian learners, domain-specific instructions, and potential biases integrating into the suggested framework that may undermine the present findings' utility to broader domains.

Another approach investigates how ML algorithms are implemented to predict learner programming performance as high or low based on different variables, including computational identity, computational thinking, programming empowerment, gender, and programming anxiety [16]. DT algorithm had the best accuracy of 96.6%, while the best cross-accuracy was obtained with RF. On the contrary, kNN performed the least favorably. Their study shows that male and university learners scored higher than female and high school learners. However, the groups were not significantly different as far as programming anxiety is concerned. The limitations of the research are the focus on the demographic of Turkish learners, the specificity of the constructs relating to specific tasks, and the need for generalisability across datasets of different populations.

The study in [3] offers a prediction model powered by AI with Genetic Programming (GP) modeling to analyze and predict learner academic performance in an online engineering education system. The performance of the GP model was better than that of traditional AI methods (e.g., ANN and SVM), demonstrating high accuracy and efficient predictions regarding learning effectiveness. Key determinants of performance are knowledge transfer, participation in class, and summative assessment, with prior knowledge having a limited effect. However, though it does advance these areas, the limited sample size, lack of real-time adaptability of the model, and locally specific findings limit its wider application and require additional research.

An extensive review of ML techniques applied to predict the learner's performance in the context of programming courses emphasizes the strength of different algorithms, the nature of the organization of datasets, and the significance of the evaluation metrics [20]. According to the review results, SVM had the best accuracy of 93.97%, whereas deep learning methods like Deep Neural Networks (DNN) have achieved significant success in detecting complex data patterns. Most studies used academic records as a dataset type; however, a meta-analysis of multiple data sources improved prediction accuracy. The review highlights the importance of using different metrics to evaluate model performance, including accuracy, F1-Score, precision, and recall, especially in unbalanced datasets. Noted limitations include using small or unbalanced datasets, a narrow range of algorithms, and overemphasizing accuracy. The results emphasize the need for larger datasets, more varied evaluation methods, and more examination of deep learning approaches to improve models of future predictive capabilities.

PerFuSIT is an example of a fuzzy logic-based module designed to personalize pedagogical approaches in ITSs for programming education [10]. This adaptive system updates different tutoring approaches dynamically, according to parameters such as performance measures, coding error types, help requests, and the time taken to solve a task. Therefore, through such an application, a fuzzy logic system can offer flexibility in the learning process, where the learners have different levels of interactivity, improving learners' performance and involvement. However, the dependency on expert regulations and the constrained testing environments may generate scaling challenges.

The study in [21] studied the relationship between the learning behaviors of learners and their grades in an online programming course employing the Random Matrix Theory to filter noisy data and discover possible patterns. It was observed that learners with superior academic performance regularly participated in practical tasks such as lab activities and exercises. The lower-achieving learners relied more on lecture notes and fell behind on engagement in practical tasks over time. This study demonstrated that data obtained from early-stage learning behaviors can be used as powerful binary predictors (pass or fail) of learning outcomes, and the use of cleaned datasets enabled significantly improved accuracy using ML algorithms, specifically SVM and XGBoost. However, this research also granted limitations; it had one institutional focus, short-term job performance outcomes, and methodological technical complexity, limiting the additional dissemination and uptake by broader generalization.

Another ML study uses data from a chatbot-assisted programming platform to explore the relationship between learning behaviors and programming performance [17]. The key performance features are solution verification, frequency of code review, code error correction using quizzes, programming practice logs, and engagement patterns. ANN produced better model predictions than other ML models like RF and SVM. While the study demonstrates the potential of ML for predicting outcomes, it is limited by its small sample size, single-institution focus, and short-term analysis of learning performance.

As an ML model, Recurrent Neural Networks (RNN) are also used as a basis for a personalized learning path recommendation system in the programming education domain [22]. The system can recommend appropriate problems to help the learner's speed and engagement by analyzing learners' ability charts and clustering users with similar skill levels. However, the system's reliance on advanced algorithms and lack of scalability testing may restrict its applicability and implementation in real-world scenarios, as they focus on data generated from a single online judge.

The study in [15] proposed personalized interventions based on the skills of at-risk learners as part of a Python programming course. Their study adopted advanced AI technologies (e.g., BERT and GPT-2) to generate personalized remedial content. Its personalized feedback significantly enhanced learners' coding competencies and learning strategies, demonstrating the power of AI-based interventions. However, the study also has limitations, including a relatively small cohort size, a focus on short-term outcomes, and reliance on resource-intensive AI models, which may limit applicability and scalability.

The study detailed in this manuscript differs from the above efforts to apply ML to programming education in three key aspects. First, it can be based on data retrievable from online learning platforms during a preliminary, intermediate, or final learning process. Second, different ML models can be evaluated using this data to find the best model for providing personalized programming feedback. Third, the empirical efficacy of such feedback in enhancing learning outcomes is assessed using controlled experimental evaluation.

III. METHOD

The experiment aims to use ML models to identify learners' weaknesses in Java programming learning and provide personalized learning feedback. Furthermore, it seeks to validate the effectiveness of this feedback in improving programming skills.

A. Hypotheses

There are two main hypotheses in this study as follows:

H1. Machine learning can be utilized to accurately predict specific weaknesses in learners' Java programming learning.

H2. Providing personalized feedback based on machine learning predictions will improve learners' performance in Java programming learning.

B. Data Collection

The experiment administers a comprehensive pre-test covering the essential concepts of basic Java programming. These concepts include syntax, variables, control flows (if, if-else, and switch), loops (while, do-while, and for loops), arrays, methods, classes, and objects. The pre-test contains 60 questions to cover 12 Java topics. Learners' responses to each concept are recorded to form the input dataset for training and testing the ML models. In addition, prior knowledge performance, time spent, and GPA are recorded.

C. Machine Learning Models

From data pre-processing, model selection, and training to evaluation, this phase lays the foundation for an ML model specifically designed to predict each learner's weaknesses from their pre-test scores, time spent on the exam, and GPA. The primary step involves validating the data by cleansing missing data, normalizing scores, and converting the categorical data (e.g., concept titles or feedback categories) to numerical values when required. The dataset will subsequently be randomly partitioned, allocating 70% for training and 30% for testing. The training dataset is utilized to develop the ML model, whereas the testing dataset evaluates its performance. So, several ML models such as RF, LR, SVM, and DT are compared to identify which model could best predict learners' weaknesses based on their pre-test, time spent, and GPA data. After finding a well-performing model, personalized guidance feedback will accordingly be delivered.

D. Personalized Feedback Generation

Based on the model's predictions, learners should be provided relevant feedback on the Java programming concepts they struggled with. For instance, personalized feedback will be prioritized and released when the model predicts that a learner struggles to grasp the programming concept while loops. The feedback input is pre-created per learning concept, with a theoretical overview, an example, an exercise, and a link to a video lesson suited to reflect the various learning types of learners.

E. Evaluation Metrics

In addition to the confusion matrix and Receiver Operating Characteristic (ROC) curves, alongside Area Under the Curve (AUC) scores, the predictive performance of the ML models is evaluated using metrics such as accuracy, precision, recall, and

F1-Score (displayed in formulas 1 through 4). These metrics are typically used in relevant work and are suitable for this study [23], [24]. A post-test is also used to investigate the effect and provide evidence for the advantages of using personalized feedback regarding improved learning outcomes.

$$Accuracy = \frac{TP+TN}{Total\ Predictions} \quad (1)$$

$$Precision = \frac{TP}{TP+FP} \quad (2)$$

$$Recall = \frac{TP}{TP+FN} \quad (3)$$

$$F1-Score = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall} \quad (4)$$

where,

- TP: True positive
- TN: True negative
- FP: False positive
- FN: False negative

F. Procedure

The experimental procedure is simplified and depicted in Fig. 1. After obtaining consent to participate in the experiment, a pre-test was conducted. The data collected was then used to train and test the ML models to select the best model for providing personalized feedback. Participants were randomly divided into two independent groups: experimental and control groups. The experimental group receives personalized feedback based on the selected ML model predictions, while the control group receives conventional feedback. Both groups were asked to follow the feedback for their learning.

This experimental setting was conducted in computer labs to ensure the experiment's control and to guarantee that participants completed the learning process based on the offered feedback. This process took five sessions, with each session lasting about 120 minutes. Once all learning sessions were finished, all participants completed the post-test.

IV. Results

This section presents the results pertaining to the prediction outcomes of the ML models. These predictions can be used of identifying the programming learning weaknesses of each learner. Furthermore, this section provides the results of the controlled experiment that aimed to investigate the effectiveness of learning Java programming through personalized feedback.

A. Dataset

This experiment involved 205 first-year undergraduate learners majoring in various computing disciplines. After collecting and pre-processing the pre-test results, 180 valid cases were identified for inclusion in the analysis of the ML models. The performance of these models was evaluated using standardized scores for each Java concept, categorizing learners as “weak” or “strong” according to a 50% threshold, time spent on the exam and GPA. Based on the optimal model, personalized feedback can be provided.

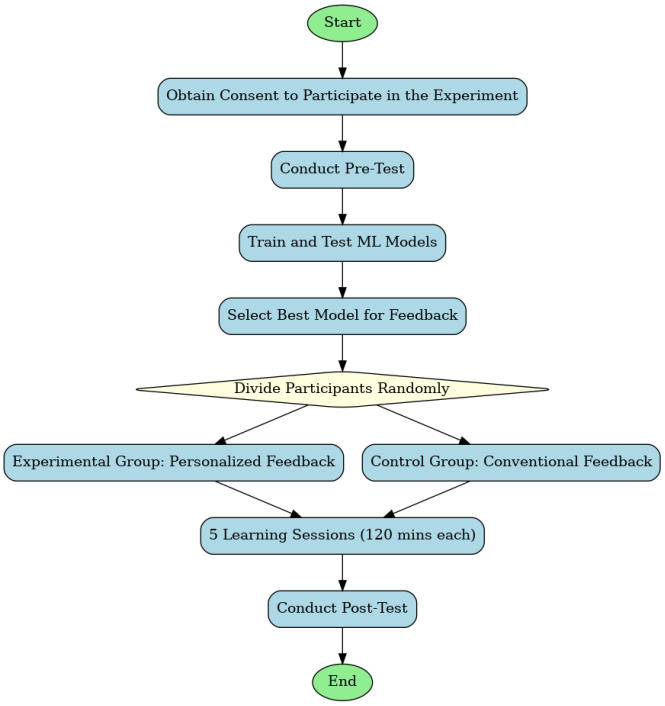

Fig. 1. Experimental procedure.

B. Analysis of the Machine Learning Models Performance

The dataset was utilized to analyze the four ML models: RF, LR, SVM, and DT. Table I summarizes the results of these models: accuracy, precision, recall, and F1-Score. Fig. 2 and Fig. 3 also present the confusion matrix and ROC curves with AUC scores, respectively. These findings will assist in identifying the appropriate ML model for the dataset.

Based on metrics analysis results, LR performs best at 98.33% accuracy meaning less errors, followed by SVM at 95% accuracy and finally RF at 81.67% accuracy. Looking into precision, SVM and LR is ahead in this perspective, with near-perfect precision that suggests low number of false positives. DT and RF are less effective since they have more false positives. Looking at the recall results, LR is outstanding (100%) at detecting all of the actual positive cases. SVM (97%) and RF (58%) display varying performance levels, while DT had the lowest recall results. For the F1-Scores, SVM (98.59%) and LR (97.96%) achieve a strong ratio between precision and recall. However, DT and RF can be less effective based on the results.

TABLE I. Performance Analysis of ML Models

Model	Accuracy	Precision	Recall	F1-Score
Random Forest	81.67%	93%	58%	72.00%
Logistic Regression	98.33%	96%	100%	97.96%
SVM	95.00%	100%	97%	98.59%
Decision Tree	75.00%	83%	68%	75.00%

In the confusion matrix results, LR has shown an exceptional level of accuracy and recall with a near 100% score in both metrics. LR can also predict positive and negative

outcomes with remarkable precision. SVM also had a high precision at 100% and a high recall at 97%, making it a strong candidate due to the balance between those two evaluation metrics. Although its recall is slightly lower than that of LR, this results in a much more competitive score. The RF model also performed well, with many true positives (35) and fewer false negatives (1). However, it was slightly decreased due to several false positives (10), leading to 93% precision. Given the model's simplicity, DT showed satisfactory recall (68%) and accuracy (75%). However, DT has also demonstrated a relatively high number of false negatives (11), signifying a worse performance than the other ML models.

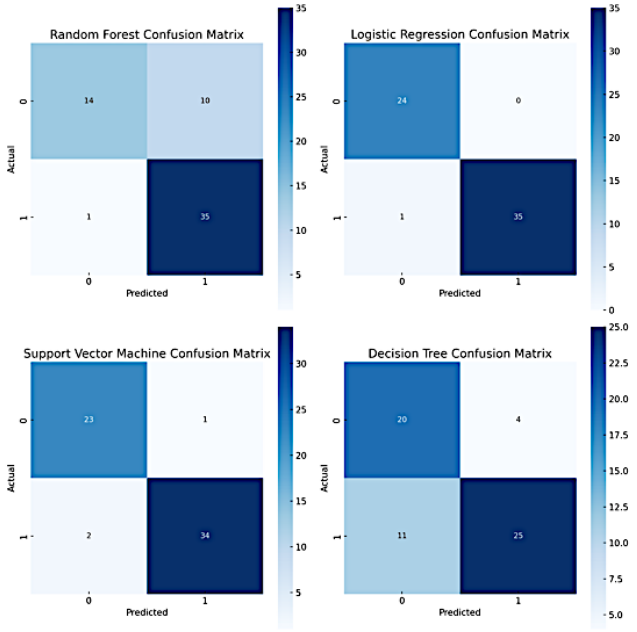

Fig. 2. Confusion matrix for the machine learning models.

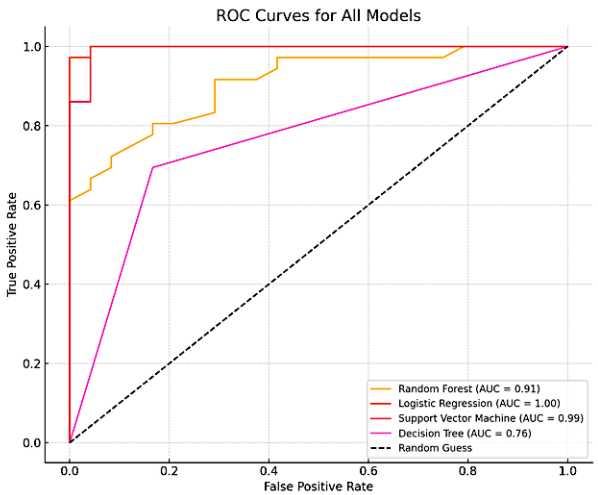

Fig. 3. ROC curves for the machine learning models.

When considering the ROC curves and AUS scores, LR stands out with an AUC of 1.00, implying perfect discrimination. SVM is also very close, with an AUC of 0.99, suggesting excellent performance. RF has a solid AUC of 0.91, indicating strong classification abilities but slightly worse than SVM and LR. DT falls behind with an AUC of 0.76, reflecting its lower performance in separating the weaknesses classes.

LR had the best overall outcomes considering all these metrics, with perfect recall, high precision, and balanced metrics. SVM performed competitively, with only slightly lower recall. RF and DT can be supplementary models for exploratory analysis or explainability. Therefore, LR is the candidate model for classifying and predicting each learner's weaknesses in Java programming learning.

C. Identifying Weaknesses of Programming Learning

LR has been identified as the most effective model among the ML approaches considered. To illustrate its capability to discern the weaknesses of each learner, two specific scenarios have been selected and presented, as depicted in Fig. 4.

Firstly, the identification of the highest-performing learner within the dataset is depicted in Fig. 4 (A). This learner exhibited deficiencies in knowledge and skills across three topics out of the twelve evaluated. These topics pertain to Java loops (including do-while and for loops) and arrays. Secondly, the identification of the lowest-performing learner in the dataset is illustrated in Fig. 4 (B). This learner demonstrated weaknesses in knowledge and skills across ten topics. These identified areas of weakness can be prioritized for each learner to facilitate the provision of personalized feedback.

Based on these findings, H1 is confirmed. It can be stated that ML can be utilized to accurately predict specific weaknesses in learners' Java programming learning.

Fig. 4. Identification of weak topics for the best-performing learner (case A) and the poorest-performing learner (case B)

D. Learning Effectiveness of Personalized feedback

After completing the phase of selecting the best-performing ML model (i.e., LR), each learner can receive personalized learning feedback. Table II gives a sample of the personalized recommendations generated for three random learners from the dataset.

TABLE II. TOPIC PERSONALIZED RECOMMENDATIONS FOR LEARNERS

Learner ID	Weak topics with scores
20161	[('Arrays', 20), ('Do-While Loops', 40), ('Classes', 40), ('Syntax', 60), ('Variables', 60), ('If Statements', 60), ('Switch Statements', 60), ('While Loops', 60), ('For Loops', 60), ('Objects', 60), ('If-Else Statements', 80), ('Methods', 100)]
20125	[('Methods', 20), ('Variables', 40), ('Do-While Loops', 40), ('For Loops', 40), ('Objects', 40), ('If Statements', 60), ('Switch Statements', 60), ('While Loops', 60), ('Syntax', 80), ('If-Else Statements', 80), ('Arrays', 100), ('Classes', 100)]
20310	[('If-Else Statements', 0), ('Do-While Loops', 0), ('For Loops', 0), ('Methods', 0), ('Classes', 20), ('Syntax', 40), ('Variables', 40), ('Arrays', 40), ('Switch Statements', 60), ('While Loops', 60), ('Objects', 60), ('If Statements', 80)]

LR can be used to adeptly identify the specific weaknesses of each learner to deliver personalized feedback. Nonetheless, a pertinent question arises: are these recommendations practical for learning? To address this issue, a controlled experimental evaluation was undertaken, wherein learners completed the learning process based on either personalized or conventional feedback as offered according to their group. The outcomes of this experiment are detailed in Table III. The sample consisted of 40 participants (out of 180 participants) who completed the experiment. Each group had 20 participants randomly assigned to either the control or experimental groups. The pre-test results indicate that they had almost similar scores. In contrast, the experimental group had better scores than the control group regarding the post-test and overall learning outcome (i.e., post-test – pre-test).

TABLE III. SUMMARY RESULTS OF THE PRE-TEST, POST-TEST AND LEARNING OUTCOME

Variable	Group	N	Mean	SD
Pre-test	Control	20	39.92	3.52
	Exp.	20	41.17	3.83
Post-test	Control	20	59.45	10.68
	Exp.	20	73.40	13.00
Learning outcome	Control	20	19.53	10.47
	Exp.	20	32.23	13.29

An independent sample t-test was run for all these variables. It was found that there was no statistically significant difference between the pre-test of the experimental group compared to the control group, $t(38) = -1.074$, $p = .289$. Regarding the post-test results, there was a statistically significant difference between the experimental group and the control group, $t(38) = -3.708$, $p<.001$. For the overall learning outcome, there was also a statistically significant difference between the experimental group and the control group, $t(38) = -3.358$, $p = .002$. According to the findings, H2 can be confirmed. It can be stated that providing personalized feedback based on ML predictions will improve learners' performance in Java programming learning.

V. DISCUSSION

The research presented in this study aimed to explore the use of ML to predict the programming learning weaknesses of learners. It emphasized the importance of learners' key data as input to ML models to identify learning patterns and gaps of programming concepts to provide adaptive and personalized interventions with relevant learning resources and feedback. The data features considered were prior knowledge obtained from a pre-test, time spent on the test, and GPA.

The study findings found that LR was the most effective model for identifying the programming learning weaknesses of learners compared to other ML models. It achieved the highest accuracy among other models, confirming its suitability for educational data analytics, particularly on the identified features for the programming domain. SVM also had a competitive performance but with lower recall compared to LR. This finding confirms the importance of precision and sensitivity in programming learning platforms when using ML. However, the limited utility of other models (i.e., RF and DT) is due to the higher false positives, though they can be helpful to exploratory analyses. These results align with previous research highlighting the robustness of LR and SVM in educational contexts [6], [7].

The presented study also took a step further by conducting a controlled experimental evaluation to investigate the effectiveness of personalized feedback based on the results of the selected ML model (i.e., LR). The findings revealed that providing personalized feedback based on LR predictions improves learners' outcomes in Java programming. By addressing these learning weaknesses, instructors and online learning platforms can implement targeted interventions to bridge learning gaps, thereby enhancing overall comprehension and engagement [10], [15]. These findings highlight the importance of integrating ML tools into programming curricula to meet individual learning needs. This is consistent with existing literature highlighting individualized learning pathways as a form of personalization for deeper understanding and long-term knowledge retention [2], [22].

The presented study offers significant implications for the design and deployment of ML in programming education interventions. This study demonstrated how ML can be used to provide precise and actionable insights that enhance learning performance. The findings also underscore the potential of integrating ML tools into computer science curricula to meet the diverse needs of learners effectively. However, some limitations need to be considered. Generalization can be limited since the results were based on a dataset from a single institute with restricted data features. Thus, more experiments and diverse data features are needed in future research. Nevertheless, this study provided initial findings that can serve as a foundation for further explorations highlighting the importance of ML-enabled personalization in education.

VI. CONCLUSION

This study addressed the challenge of identifying and resolving weaknesses in programming education using ML.

Three key contributions were made. First, multiple ML models were evaluated to predict learners' programming difficulties based on prior knowledge, time spent, and GPA. Second, a comparative analysis determined that LR was the most effective model for generating personalized feedback. Third, a controlled experimental evaluation provided empirical evidence that personalized feedback significantly enhances programming learning outcomes compared to conventional methods.

These findings highlight the potential of ML-driven personalized learning in programming education. Future research will enhance this approach by incorporating additional learner-system interaction features, such as time spent on specific concepts, quiz scores, quiz attempts, and lesson visits, to build more dynamic learner profiles. By leveraging these profiles, intelligent online learning platforms can be developed to generate adaptive learning pathways, provide targeted feedback, and integrate gamification elements to boost engagement and motivation. Furthermore, a more extensive experimental evaluation will be conducted to assess the long-term impact of personalized feedback on learning effectiveness.

REFERENCES

[1] Gligorea, M. Cioca, R. Oancea, A.-T. Gorski, H. Gorski, and P. Tudorache, "Adaptive learning using artificial intelligence in e-learning: a literature review," Educ Sci (Basel), vol. 13, no. 12, p. 1216, 2023.

[2] M. Tedre et al., "Teaching machine learning in K–12 classroom: Pedagogical and technological trajectories for artificial intelligence education," IEEE Access, vol. 9, pp. 110558–110572, 2021.

[3] P. Jiao, F. Ouyang, Q. Zhang, and A. H. Alavi, "Artificial intelligence-enabled prediction model of student academic performance in online engineering education," Artif Intell Rev, vol. 55, no. 8, pp. 6321–6344, 2022, doi: 10.1007/s10462-022-10155-y.

[4] M. T. Alshammari and A. Qtaish, "Effective Adaptive E-Learning Systems According to Learning Style and Knowledge Level.," Journal of Information Technology Education: Research, vol. 18, pp. 529–547, 2019, doi: https://doi.org/10.28945/4459.

[5] R. Mustapha, G. Soukaina, Q. Mohammed, and A. Es-Saadia, "Towards an Adaptive e-Learning System Based on Deep Learner Profile, Machine Learning Approach, and Reinforcement Learning," International Journal of Advanced Computer Science and Applications, vol. 14, no. 5, pp. 265–274, May 2023.

[6] S. Marwan, G. Gao, S. Fisk, T. W. Price, and T. Barnes, "Adaptive Immediate Feedback Can Improve Novice Programming Engagement and Intention to Persist in Computer Science," in Proceedings of the 2020 ACM Conference on International Computing Education Research, in ICER '20. New York, NY, USA: Association for Computing Machinery, 2020, pp. 194–203. doi: 10.1145/3372782.3406264.

[7] M. Murtaza, Y. Ahmed, J. A. Shamsi, F. Sherwani, and M. Usman, "AI-based personalized e-learning systems: Issues, challenges, and solutions," IEEE Access, vol. 10, pp. 81323–81342, 2022.

[8] W. Xu and F. Ouyang, "A systematic review of AI role in the educational system based on a proposed conceptual framework," Educ Inf Technol (Dordr), vol. 27, no. 3, pp. 4195–4223, 2022.

[9] T. Saito and Y. Watanobe, "Learning path recommendation system for programming education based on neural networks," International Journal of Distance Education Technologies (IJDET), vol. 18, no. 1, pp. 36–64, 2020.

[10] K. Chrysafiadi and M. Virvou, "PerFuSIT: Personalized Fuzzy Logic Strategies for Intelligent Tutoring of Programming," Electronics (Basel), vol. 13, no. 23, p. 4827, 2024.

[11] M. Abolnejadian, S. Alipour, and K. Taeb, "Leveraging ChatGPT for Adaptive Learning through Personalized Prompt-based Instruction: A CS1 Education Case Study," in Extended Abstracts of the 2024 CHI Conference on Human Factors in Computing Systems, in CHI EA '24. New York, NY, USA: Association for Computing Machinery, 2024. doi: 10.1145/3613905.3637148.

[12] P. L. S. Barbosa, R. A. F. do Carmo, J. P. P. Gomes, and W. Viana, "Adaptive learning in computer science education: A scoping review," Educ Inf Technol (Dordr), 2023, doi: 10.1007/s10639-023-12066-z.

[13] A. T. Bimba, N. Idris, A. Al-Hunaiyyan, R. B. Mahmud, and N. L. B. M. Shuib, "Adaptive feedback in computer-based learning environments: a review," Adaptive Behavior, vol. 25, no. 5, pp. 217–234, 2017, doi: 10.1177/1059712317727590.

[14] F. Okubo, T. Shiino, T. Minematsu, Y. Taniguchi, and A. Shimada, "Adaptive Learning Support System Based on Automatic Recommendation of Personalized Review Materials," IEEE TRANSACTIONS ON LEARNING TECHNOLOGIES, vol. 16, no. 1, pp. 92–105, Feb. 2023, doi: 10.1109/TLT.2022.3225206.

[15] A. Y. Q. Huang et al., "Personalized Intervention based on the Early Prediction of At-risk Students to Improve Their Learning Performance," Educational Technology & Society, vol. 26, no. 4, pp. 69–89, 2023, [Online]. Available: https://www.jstor.org/stable/48747521

[16] A. Durak and V. Bulut, "Classification and prediction-based machine learning algorithms to predict students' low and high programming performance," Computer Applications in Engineering Education, vol. 32, no. 1, p. e22679, Jan. 2024, doi: https://doi.org/10.1002/cae.22679.

[17] Y.-S. Su, Y.-D. Lin, and T.-Q. Liu, "Applying machine learning technologies to explore students' learning features and performance prediction," Front Neurosci, vol. 16, 2022, [Online]. Available: https://www.frontiersin.org/journals/neuroscience/articles/10.3389/fnins.2022.1018005

[18] Y. Jing, L. Zhao, K. Zhu, H. Wang, C. Wang, and Q. Xia, "Research Landscape of Adaptive Learning in Education: A Bibliometric Study on Research Publications from 2000 to 2022," Sustainability, vol. 15, no. 4, 2023, doi: 10.3390/su15043115.

[19] M. Ilić, G. Keković, V. Mikić, K. Mangaroska, L. Kopanja, and B. Vesin, "Predicting Student Performance in a Programming Tutoring System Using AI and Filtering Techniques," IEEE Transactions on Learning Technologies, vol. 17, pp. 1891–1905, 2024, doi: 10.1109/TLT.2024.3431473.

[20] J. P. J. Pires, F. Brito Correia, A. Gomes, A. R. Borges, and J. Bernardino, "Predicting Student Performance in Introductory Programming Courses," Computers, vol. 13, no. 9, p. 219, 2024.

[21] T. T. Mai, M. Bezbradica, and M. Crane, "Learning behaviours data in programming education: Community analysis and outcome prediction with cleaned data," Future Generation Computer Systems, vol. 127, pp. 42–55, 2022, doi: https://doi.org/10.1016/j.future.2021.08.026.

[22] T. Saito and Y. Watanobe, "Learning path recommendation system for programming education based on neural networks," International Journal of Distance Education Technologies (IJDET), vol. 18, no. 1, pp. 36–64, 2020.

[23] Y. A. Alsariera, Y. Baashar, G. Alkawsi, A. Mustafa, A. A. Alkahtani, and N. Ali, "Assessment and Evaluation of Different Machine Learning Algorithms for Predicting Student Performance," Comput Intell Neurosci, vol. 2022, no. 1, p. 4151487, Jan. 2022, doi: https://doi.org/10.1155/2022/4151487.

[24] H. Luan and C.-C. Tsai, "A Review of Using Machine Learning Approaches for Precision Education," Educational Technology & Society, vol. 24, no. 1, pp. 250–266, 2021, [Online]. Available: https://www.jstor.org/stable/26977871