JETIR.ORG ISSN: 2349-5162 | ESTD Year : 2014 | Monthly Issue
JOURNAL OF EMERGING TECHNOLOGIES AND INNOVATIVE RESEARCH (JETIR)
An International Scholarly Open Access, Peer-reviewed, Refereed Journal

# PREDICTING STUDENT PROGRAMMING PROFICIENCY USING A HYBRID PCA–MLP MACHINE LEARNING MODEL

**[1] 1st Aryan Darji, [2nd] Prof.Deep Joshi Author, 3rd Prof.Kamaljit Kaur**

[1]Designation of 1st Student, [2]Designation of 2nd Assistant Professor, [3]Designation of 3rd Assistant Professor,
Computer Department,
Grow More Faculty Of Engineering, Himmatnagar, India

***Abstract :*** Programming proficiency is a critical skill in computer science and information systems education, yet many students struggle to master foundational programming concepts. Early identification of students who may face difficulties can help educators provide timely interventions and personalized learning support. This study proposes a hybrid machine learning framework combining Principal Component Analysis (PCA) and a Multilayer Perceptron (MLP) classifier to predict student programming proficiency using cognitive, behavioural, and learning engagement features. The proposed system first pre-processes the student dataset, performs feature selection using correlation analysis, and applies PCA for dimensionality reduction. The reduced feature set is then used to train multiple machine learning classifiers, including KNN, Random Forest, SVM, and XGBoost, followed by the MLP classifier. Experimental results demonstrate that the MLP classifier achieves the highest performance with an accuracy improvement of approximately 5–6% compared with other models. The integration of PCA reduces feature vector size and removes redundant information, enabling the neural network to learn more efficiently. The results highlight the effectiveness of hybrid machine learning techniques for predictive learning analytics in programming education and demonstrate the potential of AI-driven systems to support early intervention and personalized teaching strategies.

***IndexTerms*** **- Component,formatting,style,styling,insert.**

## 1. Introduction

Programming education is a fundamental component of computer science and information systems curricula. However, many students face challenges in understanding programming concepts such as algorithm design, debugging, and logical reasoning. Identifying students who may struggle with programming courses at an early stage is essential for providing targeted support and

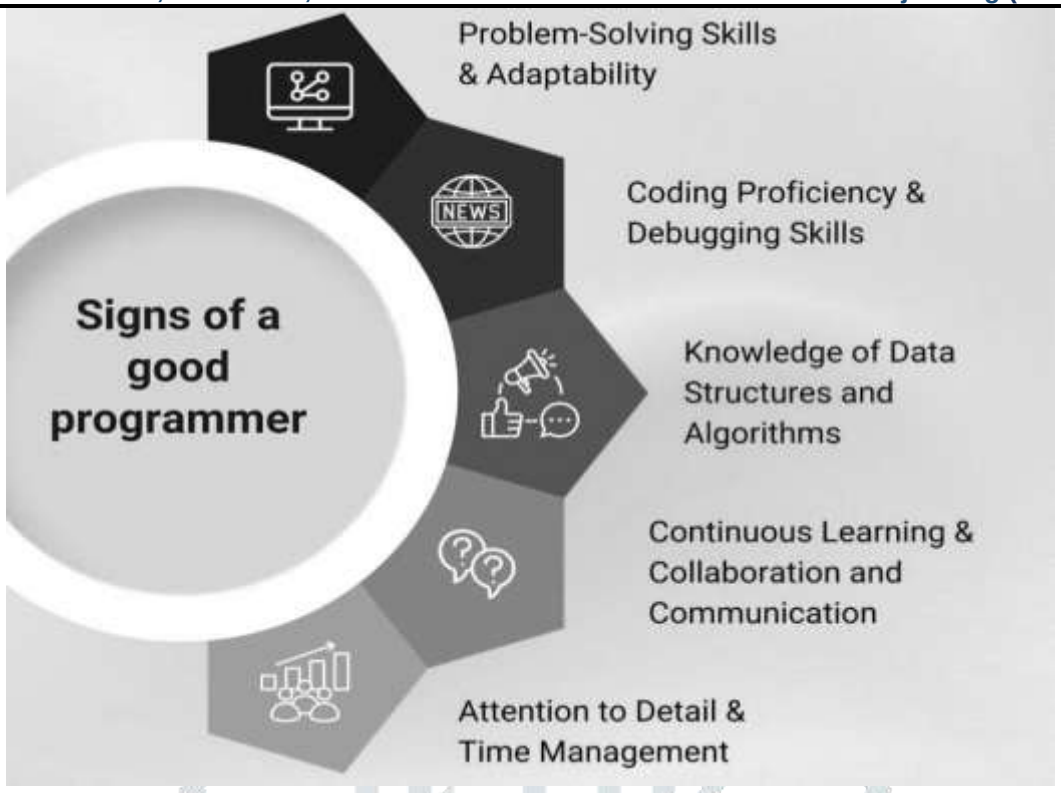


*Figure 1 Essential Signs of a good programmer*

Recent advances in artificial intelligence and machine learning have enabled the development of predictive models capable of analysing educational datasets and forecasting student performance. These models leverage various indicators such as academic background, learning behaviour, coding activity, and problem-solving ability to estimate programming proficiency.

Despite the progress in predictive analytics for education, many existing models suffer from high dimensionality and feature redundancy, which can negatively affect model performance. To address these issues, this study proposes a hybrid PCA–MLP machine learning framework for predicting student programming proficiency. PCA is used to reduce dimensionality and eliminate correlated features, while the MLP classifier learns complex nonlinear relationships between the features and programming performance. The proposed system aims to improve prediction accuracy and provide insights that can help educators support students more effectively.

**2. Literature Review**

Predictive analytics has become an important research area in programming education, enabling institutions to identify students at risk and improve teaching strategies. Several studies have explored machine learning techniques to predict student performance in programming courses.

In [1], researchers examined data-driven approaches for predicting student performance in introductory programming courses by analysing factors such as prior academic performance, coding behaviour, and engagement patterns. Machine learning models including decision trees, logistic regression, and neural networks were used to forecast student outcomes and enable early academic interventions.

Similarly, [2] conducted a comparative evaluation of machine learning algorithms for predicting student outcomes in coding courses. The study analysed features such as academic history, coding activity logs, and assignment performance, evaluating models including decision trees, random forests, support vector machines, and neural networks using metrics such as accuracy, precision, and recall.

Artificial intelligence has also been applied more broadly in higher education to predict academic performance. In [3], a predictive model was developed using machine learning algorithms such as random forests and gradient boosting to analyse student behaviour and coursework data. The study demonstrated how AI-driven analytics can support personalized learning and improve educational decision-making.

Learning analytics systems have further expanded the use of AI for monitoring student engagement and retention. The work presented in [4] showed how predictive analytics can identify early signs of disengagement by analysing digital learning behaviour and platform interaction data, enabling instructors to intervene proactively.

Recent research has also explored the integration of advanced AI technologies in programming education. For example, the BeGrading system proposed in [5] uses large language models to provide personalized feedback on student code submissions, helping students identify errors and improve their programming skills.

Another innovative approach is presented in [6], which introduces code-edit embeddings to model student debugging behaviour. By analysing sequences of code edits and error corrections, the study demonstrates how machine learning can reveal patterns in student problem-solving strategies.

A systematic review of predictive models for introductory programming courses was conducted in [7], highlighting the use of algorithms such as decision trees, logistic regression, support vector machines, and neural networks for predicting student performance. The review also identified challenges related to data quality and model interpretability.

Similarly, [8] analysed multiple machine learning classification approaches to predict programming aptitude using academic and cognitive features, concluding that neural network models often achieve superior predictive performance.

The integration of generative AI into programming education has also attracted significant attention. In [9], researchers examined student perceptions of AI-assisted coding tools and highlighted challenges related to verifying AI-generated solutions and correcting errors.

Finally, deep learning approaches for predicting student performance have been explored in [10], where neural network models implemented using PyTorch demonstrated strong predictive capabilities for analysing academic performance data.

These studies highlight the potential of machine learning and artificial intelligence in programming education. However, many existing approaches do not adequately address the issue of high-dimensional datasets and feature redundancy. The present study addresses this limitation by introducing a hybrid PCA–MLP model that combines dimensionality reduction with neural network-based prediction.

**3. Proposed Methodology**

The proposed system aims to predict student programming proficiency using a hybrid machine learning framework that integrates feature selection, PCA-based dimensionality reduction, and neural network classification.

The process begins with the collection of a student dataset containing multiple features related to programming education. These features may include logical reasoning ability, coding practice frequency, debugging performance, academic background, and engagement with learning tools.

The dataset undergoes data pre-processing, which involves handling missing values, removing duplicates, normalizing data, and preparing the dataset for machine learning analysis.

Next, feature selection is performed to identify the most relevant attributes affecting programming proficiency. Candidate features are evaluated using correlation analysis to determine their relationship with the target variable. This step eliminates irrelevant or redundant variables, improving the efficiency of the prediction model.

After feature selection, Principal Component Analysis (PCA) is applied to reduce the dimensionality of the dataset. PCA transforms the original feature space into a smaller set of principal components that capture the maximum variance in the data. This transformation reduces the size of feature vectors and removes correlations among features.

The reduced dataset is then used to train multiple machine learning classifiers, including:

- K-Nearest Neighbour (KNN)
- Random Forest
- Support Vector Machine (SVM)
- XGBoost

These models provide baseline performance comparisons.

Finally, the MLP classifier is trained using the PCA-transformed dataset. The neural network learns complex nonlinear relationships between the input features and the programming proficiency labels.

## 4. Proposed Flow & Algorithm

### 4.1 Proposed Flow:

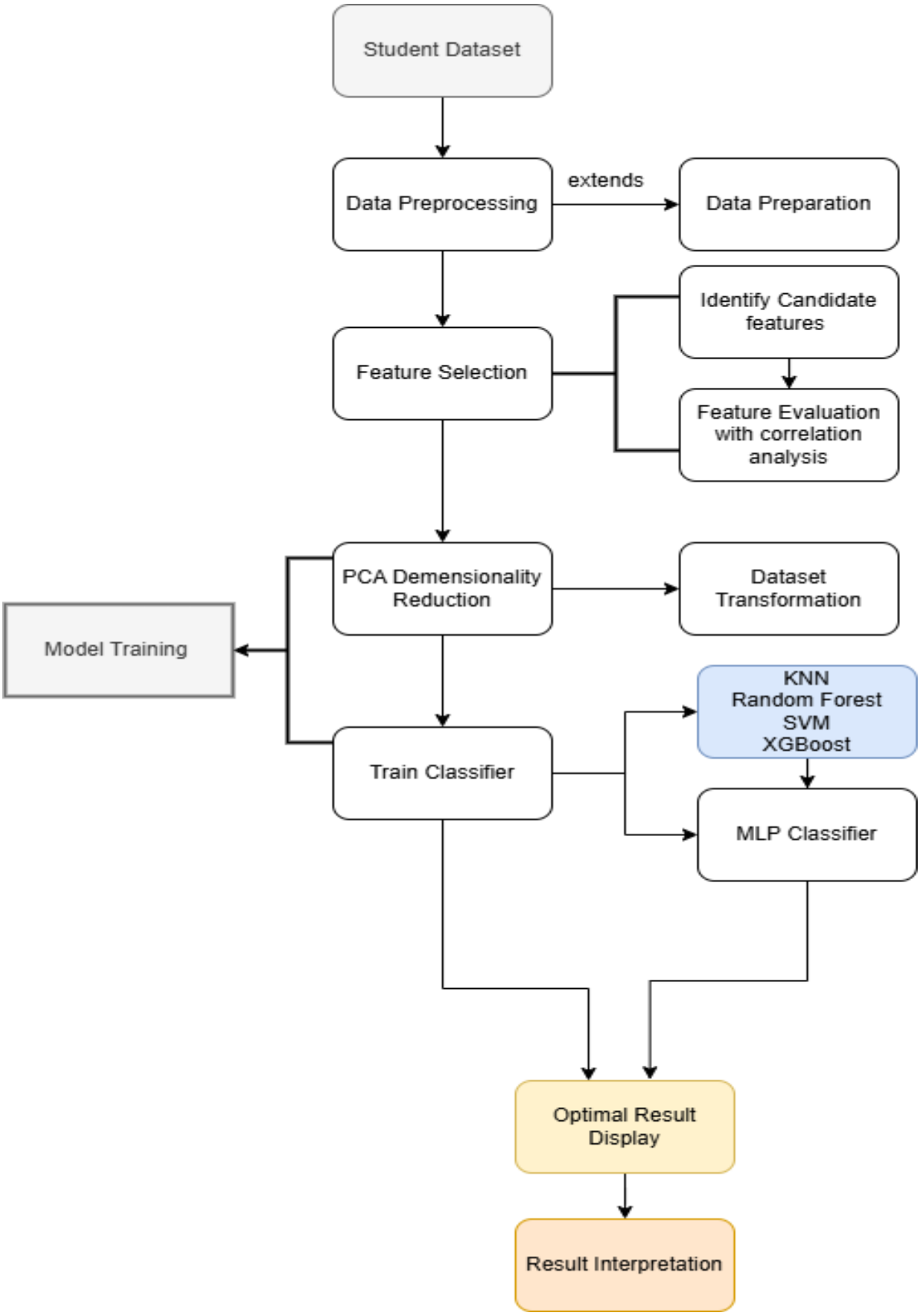


*Figure 2 Proposed Flow*

### 4.2 **Algorithm:** PCA–MLP Hybrid Model for Programming Proficiency Prediction

i Start
ii Collect student dataset containing programming-related features
iii Perform data pre-processing and normalization
iv Identify candidate features affecting programming proficiency

| | |
|---|---|
| v | Evaluate features using correlation analysis |
| vi | Select relevant feature subset |
| vii | Apply PCA for dimensionality reduction |
| viii | Transform dataset into principal component feature space |
| ix | Train machine learning models (KNN, Random Forest, SVM, XGBoost) |
| x | Train MLP classifier using PCA-transformed features |
| xi | Evaluate models using performance metrics |
| xii | Select best-performing model |
| xiii | Display optimal results and interpret predictions |
| xiv | End |

The proposed methodology aims to develop an AI-based predictive system that can evaluate and forecast students' programming proficiency using machine learning techniques. The overall process follows a structured workflow consisting of several important stages, including data collection, data pre-processing, feature selection, dimensionality reduction using PCA, and model development. Each stage is designed to gradually transform raw educational data into meaningful insights that can help predict a student's ability in programming. By combining techniques from data analytics, machine learning, and educational data mining**,** the system ensures accurate predictions while remaining practical for use in real educational environments.

### *I. Data Collection*

The first step in the process is collecting relevant data related to students' programming activities and academic performance. The dataset may include several types of information such as assignment scores, quiz marks, and lab evaluation results. In addition to academic records, behavioural indicators are also considered, for example the amount of time a student spends solving coding problems, the number of code submissions made, and participation in online coding platforms or learning management systems. Code-related information such as the number of syntax errors, debugging attempts, and the complexity of the written logic can also be included. These data points are gathered from sources like institutional Learning Management Systems (LMS), online programming platforms, and academic records**.** During this stage, ethical considerations such as student consent, data privacy, and confidentiality are carefully maintained.

### *II. Data Pre-processing and Feature Engineering*

After the dataset has been collected, the next step is data pre-processing. In real-world educational datasets, the raw data may contain missing values, duplicate records, or inconsistent formats. Therefore, the dataset is cleaned to remove errors and ensure reliability. Data normalization and encoding techniques are also applied so that all features follow a consistent scale and format suitable for machine learning algorithms.

Following pre-processing, feature engineering is performed to generate more meaningful indicators from the available data. For example, new variables such as average coding accuracy, learning improvement rate, number of debugging attempts, or average code execution time can be calculated from the original dataset. These engineered features help capture students' learning behaviour and programming ability more effectively, making them strong predictors for the final model.

### *III. PCA for Dimensionality Reduction*

Since the dataset may contain many features, some of them may be redundant or highly correlated. To address this issue, Principal Component Analysis (PCA) is applied as a dimensionality reduction technique. PCA helps simplify the dataset while preserving the most important information.

The process begins by standardizing the dataset so that all features have similar scales. This step ensures that variables with larger numerical values do not dominate the analysis. Next, a covariance matrix is computed to analyse how different features are related to each other. The covariance matrix helps identify relationships between variables and detect redundant features.

Based on the covariance matrix, eigenvalues and eigenvectors are calculated. Eigenvectors represent the directions in which the data varies the most, while eigenvalues indicate how much information (variance) is captured in those directions. These directions form the principal components of the dataset.

The principal components are then ranked according to their eigenvalues. Components that capture the most variance are considered more important. A smaller subset of these components is selected so that they collectively explain most of the dataset's information, usually around 90–95% of the total variance. This step reduces the number of features while retaining the essential patterns in the data.

Finally, the original dataset is transformed using these selected principal components. The result is a reduced-dimensional dataset, where the feature vectors are smaller but still contain the important information needed for prediction. This reduced dataset improves computational efficiency and allows machine learning models to learn more effectively by removing unnecessary or redundant features.

*IV. Model Development*

Once the dataset has been prepared and reduced using PCA, the next phase involves developing machine learning models to predict students' programming proficiency**.** Several supervised learning algorithms are applied to analyse the patterns in the dataset and estimate student performance.

## 5. Results and Discussion

The performance of the machine learning models was evaluated using metrics including accuracy, precision, recall, and F1-score. The experimental results demonstrate that the MLP classifier outperforms other machine learning algorithms.

| Model | Accuracy (%) | Precision (%) | Recall (%) | F1-score (%) |
|---|---|---|---|---|
| **Random Forest** | 87.25 | 87.2 | 85.9 | 86.5 |
| **XGBoost** | 85.48 | 85.4 | 89.7 | 89.0 |
| **KNN** | 86.21 | 85.32 | 85.14 | 85.0 |
| **SVC** | 84.18 | 83.89 | 83.77 | 83.0 |
| **MLP Classifier** | **89.61** | **90.5** | **90.8** | **90.1** |

The results show that the MLP classifier improves prediction accuracy by approximately 5–6% compared with other models. This improvement can be attributed to the neural network's ability to capture nonlinear relationships between programming skills and student learning behaviours.

Additionally, the integration of PCA significantly reduces feature vector size, removing redundant features and simplifying the input space. This allows the MLP classifier to train more efficiently and reduces the risk of overfitting. The hybrid PCA–MLP framework therefore provides a robust approach for predicting student programming proficiency.

## 6. Conclusion and Future Work

This study presents a hybrid machine learning framework that combines Principal Component Analysis (PCA) with a Multilayer Perceptron (MLP) classifier to predict student programming proficiency. The proposed system effectively pre-processes educational datasets, selects relevant features, and reduces dimensionality before training predictive models.

Experimental results demonstrate that the MLP classifier achieves the highest performance among the evaluated models, with an accuracy improvement of approximately 5–6% compared with other machine learning algorithms. The application of PCA further enhances model efficiency by reducing feature vector dimensionality and eliminating redundant variables.

The findings confirm that integrating AI-driven predictive analytics with dimensionality reduction techniques can significantly improve the accuracy and reliability of programming proficiency prediction models.

Future work may extend this research by integrating additional behavioural indicators such as coding platform interaction data and real-time programming activity logs. Furthermore, the proposed system can be expanded into an adaptive learning platform that provides personalized feedback and recommendations based on individual student competency profiles.

## 7. References

1. Al-Shabandar, R., Hussain, A. J., Keight, R., Laws, A., & Radi, N. (2018). Machine learning approaches to predict learning outcomes in massive open online courses. *Computers in Human Behavior, 92*, 678–688.
2. Hafdi, Zakaria Soufiane, and Said El Kafhali. "A Comparative Evaluation of Machine Learning Methods for Predicting Student Outcomes in Coding Courses." AppliedMath 5, no. 2 (2025): 75.
3. Pacheco-Mendoza, Silvia, Cesar Guevara, Amalín Mayorga-Albán, and Juan Fernández-Escobar. "Artificial intelligence in higher education: A predictive model for academic performance." Education Sciences 13, no. 10 (2023): 990.
4. Khosravi, H., Sadiq, S., & Gasevic, D. (2022). AI-enabled learning analytics for personalized education. *Computers & Education: Artificial Intelligence, 3*, 100044.
5. Yousef, Mina, Kareem Mohamed, Walaa Medhat, Ensaf Hussein Mohamed, Ghada Khoriba, and Tamer Arafa. "BeGrading: large language models for enhanced feedback in programming education." Neural Computing and Applications 37, no. 2 (2025): 1027-1040.
6. Heickal, Hasnain, and Andrew Lan. "Learning Code-Edit Embeddings to Model Student Debugging Behavior." In International Conference on Artificial Intelligence in Education, pp. 91-98. Cham: Springer Nature Switzerland, 2025.

7. Mohamed, A., Ali, M., & Rahman, A. (2024). Predicting student performance in introductory programming courses: A systematic review. Computers, 13(9), 219. MDPI. https://doi.org/10.3390/computers13090219

8. Sari, E., & Nugroho, W. (2023). Analysis of machine learning classification approaches for predicting students' programming aptitude. Sustainability, 15(17), 12917. MDPI. https://doi.org/10.3390/su151712917

9. Yilmaz, A., & Lee, M. (2025). Integrating generative AI into programming education: Student perceptions and the challenge of correcting AI errors. International Journal of Artificial Intelligence in Education. Springer. https://doi.org/10.1007/s40593-025-00496-4

10. Patel, R., & Zhang, H. (2025). Predicting student academic performance using deep learning: A PyTorch-based approach. Preprints.org, Version 1. https://doi.org/10.20944/preprints202507.2493.v1

11. Liu, K., & Chen, Y. (2025). A machine learning approach for predicting student progress in online programming education. International Journal of Artificial Intelligence in Education. Springer. https://doi.org/10.1007/s40593-025-00510-9

12. Rahman, S., & Hossain, F. (2025). The influence of artificial intelligence tools on learning outcomes in computer programming: A systematic review and meta-analysis. Computers, 14(5), 185. MDPI. https://doi.org/10.3390/computers14050185

13. Kwon, D., & Martinez, J. (2025). Detecting struggling student programmers using proficiency taxonomies. arXiv preprint. https://arxiv.org/abs/2508.17353

14. Kumar, V., & Alhassan, R. (2025). Teaching with AI: A systematic review of chatbots, generative tools, and tutoring systems in programming education. arXiv preprint. https://arxiv.org/abs/2510.03884

15. Shen, G. (2023). The prediction of programming performance using student-written code. Education and Information Technologies.
https://doi.org/10.1007/s10639-022-11146-w SpringerLink+1

16. Llanos, J., et al. (2023). Early prediction of student performance in a CS1 programming course. Journal of Educational Data Mining.
https://www.ncbi.nlm.nih.gov/articles/PMC10702991/ PMC

17. Pires, J. (2024). Predicting higher education students' aptitude to learn to program in introductory programming courses. ACM Proceedings. https://doi.org/10.1145/3674912.3674949 ACM Digital Library

18. Rodriguez-Ortiz, M. Á., Santana-Mancilla, P. C., & Anido-Rifón, L. E. (2025). Machine learning and generative AI in learning analytics for higher education: A systematic review of models, trends, and challenges. Applied Sciences, 15(15), Article 8679. https://doi.org/10.3390/app15158679 MDPI

19. Zhang, V., Jeffries, B., & Koprinska, I. (2025). A machine learning approach for predicting student progress in online programming education. International Journal of Artificial Intelligence in Education. https://doi.org/10.1007/s40593-025-00510-9 SpringerLink

20. Keuning, H., Alpízar-Chacón, I., Lykourentzou, I., Beehler, L., Köppe, C., de Jong, I., & Sosnovsky, S. (2024). Students' perceptions and use of generative AI tools for programming across different computing courses. arXiv preprint. https://arxiv.org/abs/2410.06865