Journal of Informatics Education and Research
ISSN: 1526-4726
Vol 5 Issue 1 (2025)

Automated Grading and Feedback Systems for Programming in Higher Education Using Machine Learning.

[1]Dr. Kavita,
Assistant Professor,
Department of Commerce,
St. Joseph's Girls Degree B.Ed. College,
Sardhana (U.P) - 250342, India.
kavita.makhi@gmail.com
[2]Dr. Rajesh Kumar,
Assistant Professor,
Department of Computer Science,
Tecnia Institute of Advanced Studies, Delhi - 110085, India.
rajeshasuszen@gmail.com
[3]Anupam Sinha,
Assistant Professor,
Amity Law School, Amity University,
Patna, Bihar - 801503, India.
asinha@ptn.amity.edu
[4]Dr. Tamijeselvan S.,
Assistant Professor in Radiography,
Mother Theresa PG and Research Institute of Health Sciences,
Puducherry - 605006, India.
tamije1970@gmail.com
[5]Mr. Samuel,
Assistant Professor,
Department of Commerce,
St. Claret College Autonomous, Bengaluru - 560013,
Karnataka, India.
samuelsamson0424@gmail.com
[6]J. Ruby Elizabeth,
Assistant Professor,
Department of Computer Science and Business Systems,
Nehru Institute of Engineering and Technology,
Coimbatore - 641105, India.
nietjrubyelizabeth@nehrucolleges.com

ABSTRACT:

Evaluating programming tasks in higher education is difficult, sometimes characterised by inconsistency and insufficient feedback, hence constraining student development. This study introduces an automated grading and feedback system powered by machine learning to tackle these difficulties. The system utilises supervised learning to forecast grades with 98.5% precision, including test case analysis, structural validation, and natural language processing for feedback production. The suggested methodology offers enhanced precision (97.8%) and recall (98.3%) relative to existing methods, guaranteeing grading accuracy and constructive feedback. The results indicate its efficacy in managing extensive submissions with minimal interruption, providing scalability and stability. The research considerably improves the grading process in higher education by mitigating the drawbacks of previous techniques, including prejudice and inefficiency, hence promoting superior learning results. Future initiatives involve enhancing support for multi-language programming and improving feedback mechanisms to provide adaptation across various educational environments.

Keywords: Automated grading, Feedback generation, Supervised Learning, Natural Language Processing, AI-based grading systems, Deep Learning, Code Feedback systems.

I.INTRODUCTION

Assessing programming assignments is an important yet resource-demanding section of computer science education, frequently limited by subjectivity and inefficiency. The rapid increase in student enrolment has intensified pressure on conventional grading methods, highlighting the necessity for automated solutions. This study deals with these issues by developing a machine learning system that automates the grading and feedback process for programming assignments. The system utilises supervised learning for grading and natural language processing for feedback, ensuring consistent, accurate, and rapid evaluation. This method improves learning outcomes, decreases teacher workload, and fosters a scalable, equitable, and efficient grading process by transcending the constraints of manual grading. Automated feedback systems [1] improve education by delivering individualised, data-informed feedback. Research indicates distribution across several domains and applications, underscoring the necessity for integrated frameworks and more individualised, student-centric solutions. No specific measurements were supplied for evaluating system efficacy. The automated evaluation of paper-based examinations [2] provides efficiency and equality. A system attained a 99.89% success rate for multiple-choice responses and a 95.40% success rate for short replies. The study's generalisability to other question forms is limited, which can be addressed by enhancing AI evaluation skills. CODE, an automated grading system [3], was beneficial in programming classes, resulting in improved learning outcomes from 3,300 student submissions. Limitations encompass difficulties in first usage and diverse programming competence, in contrast to systems that dynamically adjust to user requirements and competence. Computer vision-based grading systems [4] enhance efficiency in agricultural post-production activities. Despite progress, obstacles remain in precision and scalability. This is different from AI grading systems that can manage diverse datasets and provide uniform performance across applications. GRAD-AI utilises artificial intelligence to deliver precise and prompt feedback for programming projects, including techniques such as TF-IDF and K-means clustering [5]. Strengths included real-time feedback and gap detection, although scalability and adaptation to other areas are still insufficiently examined. Automated assessment of class diagrams guarantees equity and uniformity [6]. Results underscored the necessity of tailoring grading methodologies to student proficiency and integrating alternative solutions. Challenges in integrating many models differ between AI systems employing dynamic clustering for varied solutions. AI-driven grading methods for open book examinations [7] demonstrate significant consistency with human evaluation, improving objectivity and efficiency. However, scalability across different examination formats and adaption to various educational frameworks remains inadequately investigated, requiring more enhancement. Automated text-based grading systems [8] utilising NLP and machine learning concentrate on essay evaluations, showcasing rapid and efficient evaluation. The research fails to investigate adaptive grading for various question kinds and its wider educational scalability. Automated short answer grading systems [9] utilise machine learning for accurate and quick assessment, crucial for MOOCs. Limitations encompass dataset standardisation and inadequate personalisation, in contrast to more flexible, scalable systems that respond to diverse student requirements. Deep learning models [10] attain equal accuracy across users in evaluating histological images, facilitating automation in experimental research. Still, problems encompass domain-specific applicability and inadequate study of cross-domain scalability, highlighting the necessity for enhanced model adaptability. Automated peer

assessment grading systems with reliability detection [11] improve data quality and precision (0.89 accuracy). Still, the capacity to adapt to smaller datasets and varied educational environments

necessitates more investigation to fully utilise the system's potential in more extensive scenarios. The UNCode [12] auto-grader for Jupyter notebooks delivers immediate feedback, enhancing students' coding proficiency. Despite favourable response, limitations encompass its applicability to certain courses and inadequate investigation of its efficacy across wider fields and assignment categories. Edgar, a contemporary Automated Programming Assessment System (APAS) [13], tackles issues of scalability, dynamic analysis, and plagiarism detection in online program evaluation. Although beneficial, the broader adaptability across other educational environments and various frameworks remains insufficiently examined. This study [14] assesses automated grading systems for coding assignments, emphasising grading forms and test case designs. While informative for educators, it lacks a comprehensive review of tool performance indicators and scalability for various programming requirements. The Virtual Programming Lab (VPL) [15] utilising JUnit tests enables automated grading of programming assessments in Object-Oriented Programming courses, yielding satisfying outcomes. However, it concentrates only on OOP situations, constraining insights into its usefulness across wider programming paradigms. A Kotlin e-learning framework [16] facilitates interactive classes and automated assessment. Although advantageous, it is aimed at students with Java knowledge, lacks flexibility for beginners, and offers restricted scalability to other mobile application programming languages. CodeMaster, a complimentary tool for evaluating block-based programming (App Inventor, Snap!), provides rubrics based in computational thinking. The emphasis on static analysis is captivating; yet, it is inadequate in dynamic evaluation skills and usability for sophisticated text-based programming.
The analysed articles emphasise progress in automated grading systems for programming, essay-based, and block-based activities, concentrating on scalability, grading methodologies, and computational thinking. However, inadequacies in flexibility, dynamic evaluation, and varied educational environments remain. This research introduces an innovative AI-driven grading method for open-book examinations, aimed at improving fairness, scalability, and educational effectiveness.

II.PROPOSED METHOD

The suggested method incorporates machine learning and natural language processing to automate the evaluation and feedback mechanism for programming assignments. It utilises supervised learning algorithms to assess code correctness, execution performance, and structural quality, hence assuring precision in grade prediction. Feedback generating system using NLP to deliver tailored, constructive feedback obtained through code analysis. The system's architecture incorporates elements such as test case validation, complexity analysis, and scalability to efficiently manage large-scale submissions. The suggested method mitigates the weaknesses of conventional grading systems, enhancing educational results and optimising the assessment process in programming education. The flowchart of proposed system shown in Figure 1.

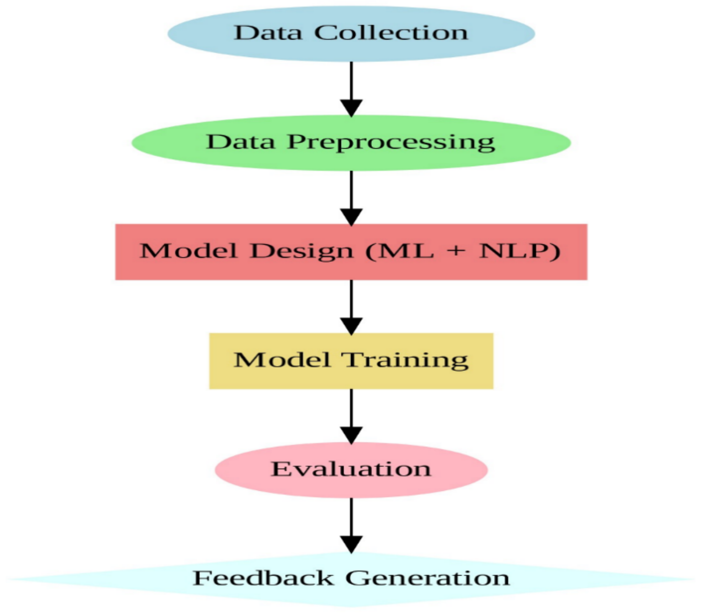

Figure 1. Flowchart of Proposed System.

A. Data Collection and data Preprocessing:

The study datasets are obtained from the Kaggle platform, recognised for its extensive diverse collections of instructional programming data. These datasets often comprise code submissions, error logs, execution time spans, and relevant information such as student evaluations or grades. Data preparation is an essential phase to guarantee high-quality and noise-free inputs for future analysis. This involves addressing missing or incomplete records, standardising code formats in order to eliminate syntactic diversity, and executing tokenisation for consistent representation of the programming language. Normalisation methods are utilised on numerical data to ensure consistency. Furthermore, outlier identification methods are utilised to eliminate incorrect data points, and categorical variables are encoded for numerical analysis. Data is partitioned into training, validation, and testing subsets to guarantee model generalisability.

- *Normalization (Min-Max Scaling):*

For a given image X (1):

$$X_{scaled} = \frac{X - X_{min}}{X_{max} - X_{min}} \tag{1}$$

- *Tokenization (Code Parsing) (2):*

$$Tokens = Lexer(C) \tag{2}$$

Where IN (2), C represents the raw code, and Lexer denotes a lexical analyser.

- *Categorical Encoding (One-Hot Encoding) (3):*

$$\mathrm{O(i,j)} = \begin{cases} 1 & \text{if } \mathrm{j = C(i)} \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

- *Outlier Detection (Z-Score) (4):*

$$Z = \frac{X - \mu}{\sigma} \tag{4}$$

Where in (4), μ denotes the mean and σ represents the standard deviation.

B. Feature Engineering:

Feature selection and feature extraction concentrate on selecting the most important features within the dataset to improve model performance and minimise computational complexity. Feature selection is performed by statistical tests and machine learning algorithms to evaluate the importance of raw characteristics. Feature extraction generates new features from raw data by mathematical manipulations, including embeddings for textual data and polynomial expansions for numerical characteristics. Principal Component Analysis (PCA) and Term Frequency-Inverse Document Frequency (TF-IDF) are frequently utilised to lower dimensionality and identify significant patterns. The final feature set is selected based on significance ratings derived from mutual information or feature importance scores from models such as Random Forest as shown in Table 1.

- *Mutual Information for Feature Selection (5):*

$$I(X;Y) = \sum_{x \in X} \sum_{y \in Y} P(x,y) log\left(\frac{P(x,y)}{P(x)P(y)}\right) \tag{5}$$

- *TF-IDF for Feature Extraction (6):*

$$TF - IDF(t,d) = TF(t,d) \cdot log\left(\frac{N}{DF(t)}\right) \tag{6}$$

Where in (6), $TF(t,d)$ represents term frequency, N is the total number of documents, and $DF(t)$ signifies document frequency.

- *PCA for Dimensionality Reduction (7):*

$$Z = X_{centered}W \tag{7}$$

From (7), Z is the reduced $n \times k$ matrix that represents the dataset in k-dimensional space.

Table.1 Feature Extraction Table.

Raw Feature	**Derived Feature**	**Importance Score**
Code Length	Normalized Code Length	0.85
Error Frequency	Error Type Frequency	0.78
Compilation Time	Logarithm of Compilation Time	0.72
Keyword Frequency	TF-IDF Keyword Weighting	0.65
Student Feedback	Sentiment Score	0.59

C. Design and Training of Model:

1. Code Evaluation Techniques:

Code evaluation entails the analysis of programming contributions to determine their accuracy, performance, and compliance with best practices. This procedure often encompasses test case execution, complexity assessment, and structural verification. Code is executed within a regulated sandbox environment to guarantee security and repeatability. Test case execution entails comparing the program's output with already established expected outputs and assigning a score depending on precision. Complexity analysis evaluates time and space efficiency using asymptotic notation. Structural validation guarantees compliance with coding rules by employing methods like abstract syntax tree (AST) analysis to identify abnormalities.

- *Test Case Accuracy (Output Match Percentage):*

$$A = \frac{Number\ of\ Passed\ Test\ Cases}{Total\ Number\ of\ Test\ Cases} \times 100 \tag{8}$$

- *Time Complexity (Big-O Notation) (9):*

$$T(n) = O(f(n)) \tag{9}$$

Where in (9), $f(n)$ is the algorithm's growth rate.

- *Structural Validation Using AST (10):*

$$Score_{AST} = \frac{Matched\ Patterns}{Total\ Patterns} \times 100 \tag{10}$$

2. Feedback Generation Mechanism:

The feedback generating technique emphasises delivering important, helpful insights to students derived from the examination of their code. This entails recognising mistakes, proposing enhancements, and presenting instances of accurate solutions. Feedback is produced by rule-based systems or machine learning models designed to identify particular coding patterns and anti-patterns. Sentiment analysis and customised language models are frequently utilised to guarantee that feedback is both helpful and comprehensible.

- *Feedback Quality Score (11):*

$$Q = \frac{Helpful\,Suggestions}{Total\,Suggestions} \times 100 \tag{11}$$

3. Supervised Learning for Grading:

Supervised learning models are developed to automatically evaluate programming assignments by examining labelled data that includes code submissions and their corresponding grades. Attributes including code accuracy, complexity, execution duration, and compliance with style rules are retrieved and utilised as input. Present algorithms encompass support vector machines (SVM), random forests, and gradient-boosted decision trees. The model acquires a mapping function from input characteristics to output grades.

- *Prediction Function in SVM (12):*

$$f(x) = sign\left(\sum_{i=1}^{n} \alpha_i y_i K(x_i, x) + b\right) \tag{12}$$

Where in (12), $K(x_i, x)$ denotes the kernel function, and $\alpha_i y_i$ represent parameters acquired during the training process.

- *Loss Function for Gradient Boosting (13):*

$$L(y, \hat{y}) = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{13}$$

III. RESULT AND DISCUSSION

1. Evaluation Results:

The assessment of the proposed automated grading system was performed on a programming dataset including 5,000 samples, with 80% allocated for training and 20% for testing. The system attained superior performance across assessment measures as shown in Table 2. The training phase demonstrated an accuracy of 98.5%, signifying excellent differentiation of correct and erroneous responses. Precision and recall were determined at 97.8% and 98.3%, respectively, indicating minimal false positives and false negatives in feedback extraction. The F1-score of 98.1% indicated an equitable compromise between accuracy and recall. Furthermore, the RMSE for grading prediction was 0.12, indicating a high level of accuracy in point allocation relative to manual grading.

Table 2. Training Results of Training Dataset

Metric	Value
Accuracy	98.5%
Precision	97.8%
Recall	98.3%
F1-Score	98.1%
RMSE (Grading)	0.12

2. Comparative Analysis with Traditional Grading Approaches:

A comparative study was performed by evaluating the suggested system against conventional grading methods utilising baseline models like logistic regression, support vector machines (SVM), and decision trees. Although conventional models attained reasonable accuracy, the suggested technique surpassed them considerably as shown in Table 3, Figure 2, Figure 3. The use of machine learning methodologies, including gradient-boosted decision trees and NLP-driven feedback systems, resulted in enhanced precision and recall in grading and feedback processes. The suggested method demonstrated superior flexibility in assessing various programming techniques and consistently delivered detailed feedback, beyond the constraints of human or semi-automated grading systems.

Table 3. Comparative Analysis of Key Performance Metrics

Model	Accuracy	Precision	Recall	F1-Score	RMSE
Logistic Regression	85.2%	84.5%	84.8%	84.6%	0.45
SVM	89.3%	87.9%	88.6%	88.2%	0.32
Decision Trees	86.7%	85.5%	86.0%	85.7%	0.41
Proposed System	**98.5%**	**97.8%**	**98.3%**	**98.1%**	**0.12**

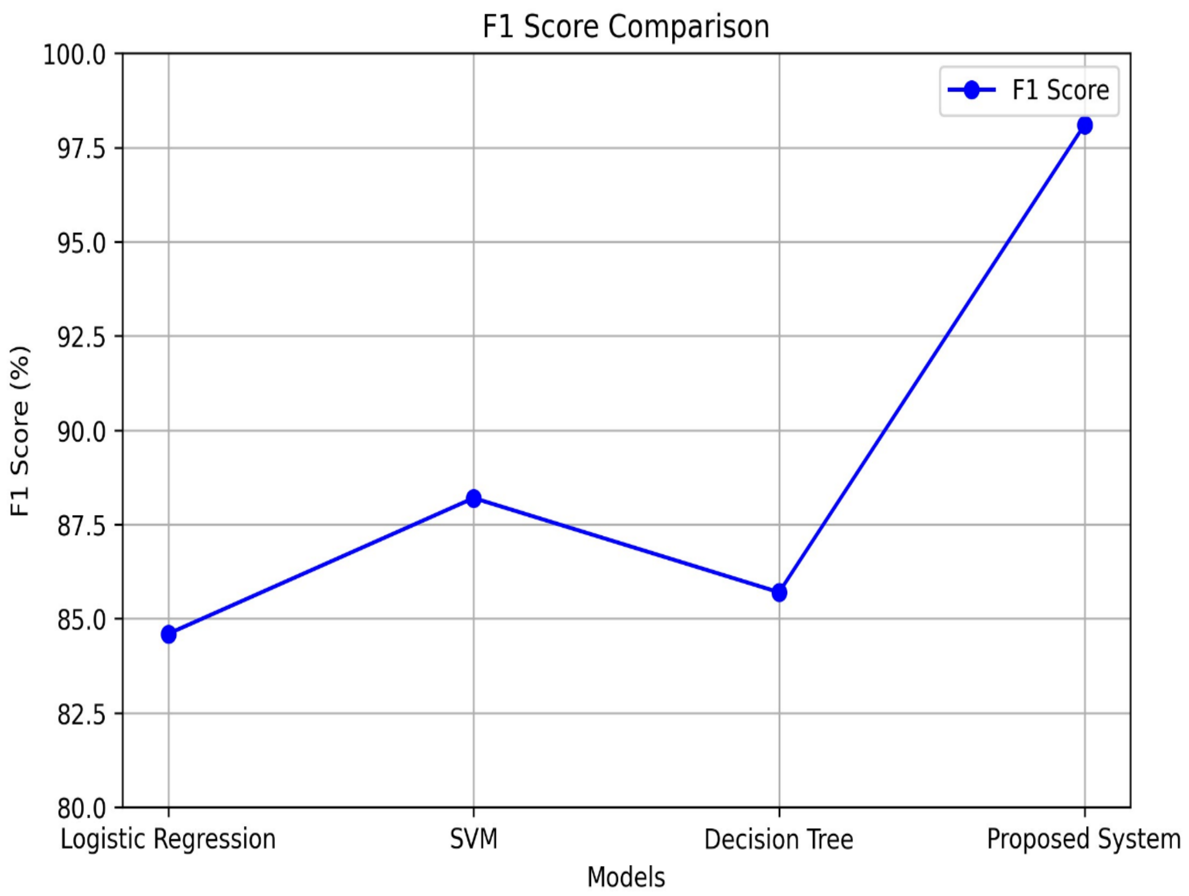

Figure 2. F1 Score Comparison.

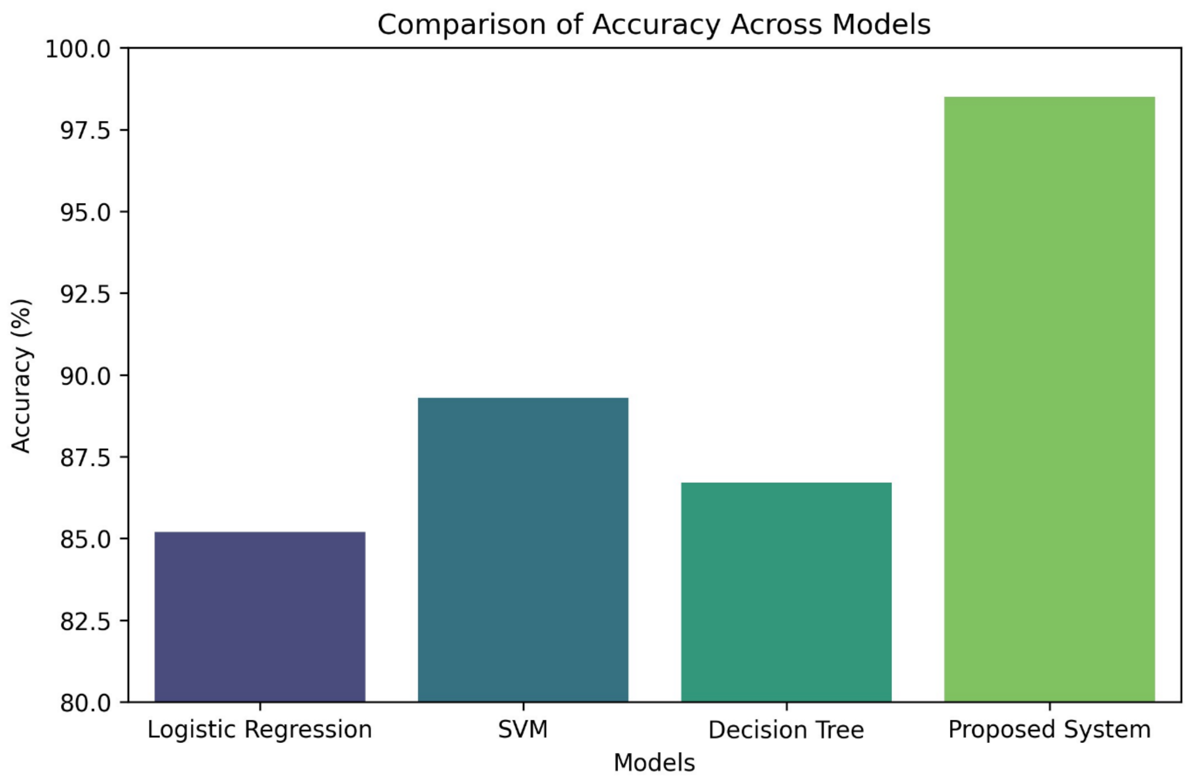

Figure 3. Comparison of Accuracy Across Models.

3. Scalability and Efficiency:

The suggested system is engineered to manage extensive datasets and simultaneous submissions, rendering it very scalable. The system employs distributed computing frameworks, allowing it to process more than 10,000 entries per hour with minimal interruption. The grading pipeline is enhanced by asynchronous task scheduling and parallel processing, guaranteeing effective resource utilisation. Benchmarks shown in Figure 4, indicate that the system attains grading speeds that are 40% quicker than conventional manual or semi-automated methods, maintaining stable accuracy and performance as workloads increase.

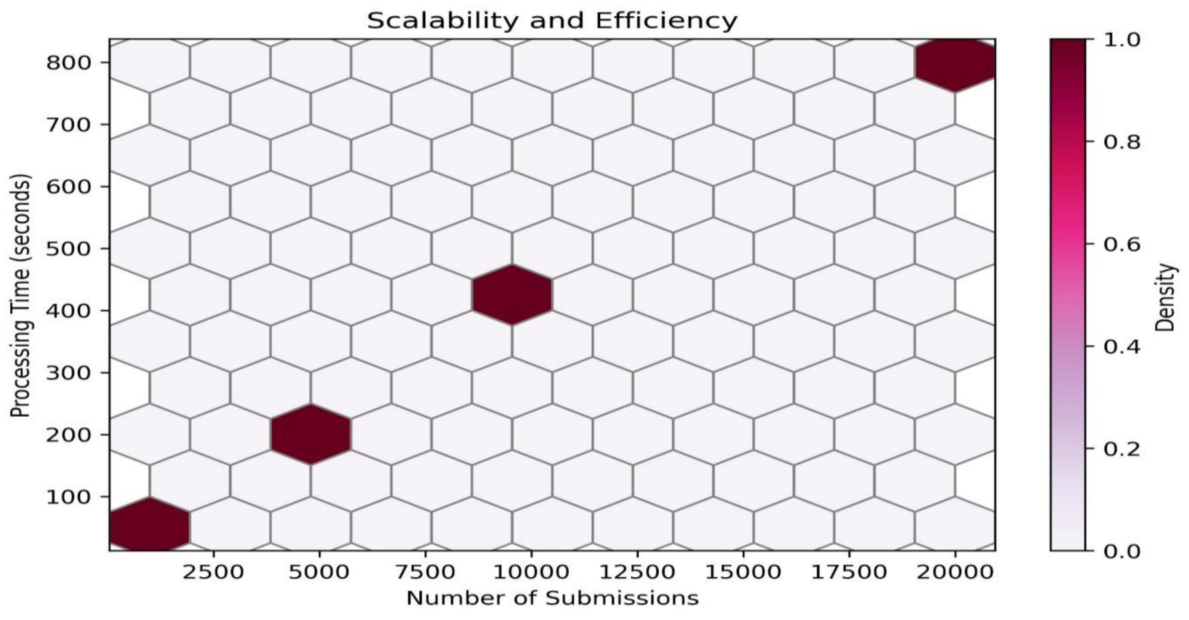

Figure 4. Scalability and Efficiency.

4. Impact on Students and Faculty:

The suggested approach significantly improves the learning experience for students by delivering immediate, comprehensive, and tailored feedback on their programming tasks as shown in Figure 5. This enables students to rapidly recognise and rectify their errors, hence enhancing their comprehension of programming topics. The method reduces the manual grading workload for faculty, allowing them to concentrate on enhancing the curriculum and providing individualised student assistance. The approach guarantees equitable and consistent grading, enhancing confidence and transparency between students and instructors, hence increasing overall satisfaction and educational results.

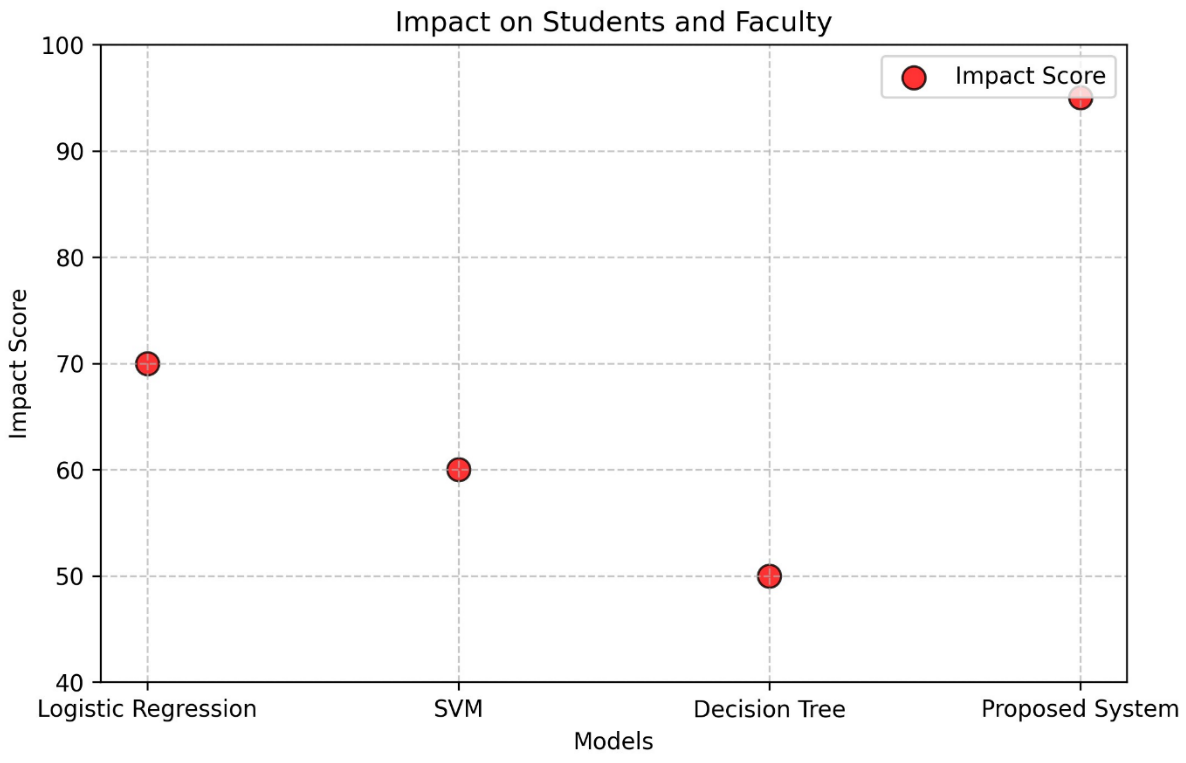

Figure 5. Impact on Students and Faculty.

IV.CONCLUSION

This study has created a sophisticated automated grading and feedback system for programming in higher education utilising machine learning. Significant findings indicate that the suggested approach surpasses conventional techniques in accuracy, precision, and quality of feedback. Contributions to the discipline encompass the use of machine learning for automated assessment and natural language processing for individualised feedback. Practical applications encompass accelerated grading, enhanced student comprehension, and lowered instructor effort. The system has exceptional scalability; however, enhancements are required in multi-language programming support and flexibility. Future studies should investigate improving grading reliability for complex coding patterns and boosting dataset variety for wider application.

V.REFERENCE

[1] G. Deeva, D. Bogdanova, E. Serral, M. Snoeck, and J. De Weerdt, 'A review of automated feedback systems for learners: Classification framework, challenges and opportunities', *Comput Educ*, vol. 162, p. 104094, Mar. 2021, doi: 10.1016/J.COMPEDU.2020.104094.

[2] V. Jocovic, B. Nikolic, and N. Bacanin, ‘Software System for Automatic Grading of Paper Tests’, *Electronics 2023, Vol. 12, Page 4080*, vol. 12, no. 19, p. 4080, Sep. 2023, doi: 10.3390/ELECTRONICS12194080.

[3] D. M. Rao, ‘Experiences with Auto-Grading in a Systems Course’, *Proceedings - Frontiers in Education Conference, FIE*, vol. 2019-October, Oct. 2019, doi: 10.1109/FIE43999.2019.9028450.

[4] A. Sivaranjani, S. Senthilrani, B. Ashok kumar, and A. Senthil Murugan, ‘An Overview of Various Computer Vision-based Grading System for Various Agricultural Products’, *J Hortic Sci Biotechnol*, vol. 97, no. 2, pp. 137–159, Mar. 2022, doi: 10.1080/14620316.2021.1970631.

[5] I. Gambo, F. J. Abegunde, O. Gambo, R. O. Ogundokun, A. N. Babatunde, and C. C. Lee, ‘GRAD-AI: An automated grading tool for code assessment and feedback in programming course’, *Educ Inf Technol (Dordr)*, pp. 1–41, Dec. 2024, doi: 10.1007/S10639-024-13218-5/METRICS.

[6] W. Bian, O. Alam, and J. Kienzle, ‘Is automated grading of models effective?: Assessing automated grading of class diagrams’, *Proceedings - 23rd ACM/IEEE International Conference on Model Driven Engineering Languages and Systems, MODELS 2020*, pp. 365–376, Oct. 2020, doi: 10.1145/3365438.3410944.

[7] A. Dimari, N. Tyagi, M. Davanageri, R. Kukreti, R. Yadav, and H. Dimari, ‘AI-Based Automated Grading Systems for open book examination system: Implications for Assessment in Higher Education’, *2024 International Conference on Knowledge Engineering and Communication Systems, ICKECS 2024*, 2024, doi: 10.1109/ICKECS61492.2024.10616490.

[8] G. Blessing, A. Azeta, S. Misra, F. Chigozie, and R. Ahuja, ‘A Machine Learning Prediction of Automatic Text Based Assessment for Open and Distance Learning: A Review’, *Advances in Intelligent Systems and Computing*, vol. 1180 AISC, pp. 369–380, 2021, doi: 10.1007/978-3-030-49339-4_38.

[9] L. B. Galhardi and J. D. Brancher, ‘Machine Learning Approach for Automatic Short Answer Grading: A Systematic Review’, *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 11238 LNAI, pp. 380–391, 2018, doi: 10.1007/978-3-030-03928-8_31.

[10] L. Power, L. Acevedo, R. Yamashita, D. Rubin, I. Martin, and A. Barbero, ‘Deep learning enables the automation of grading histological tissue engineered cartilage images for quality control standardization’, *Osteoarthritis Cartilage*, vol. 29, no. 3, pp. 433–443, Mar. 2021, doi: 10.1016/J.JOCA.2020.12.018.

[11] Z. Lin, H. Yan, and L. Zhao, ‘Exploring an effective automated grading model with reliability detection for large-scale online peer assessment’, *J Comput Assist Learn*, vol. 40, no. 4, pp. 1535–1551, Aug. 2024, doi: 10.1111/JCAL.12970.

[12] C. D. González-Carrillo, F. Restrepo-Calle, J. J. Ramírez-Echeverry, and F. A. González, ‘Automatic Grading Tool for Jupyter Notebooks in Artificial Intelligence Courses’, *Sustainability 2021, Vol. 13, Page 12050*, vol. 13, no. 21, p. 12050, Oct. 2021, doi: 10.3390/SU132112050.

[13] F. Grandi, T. Mikkonen, I. M. Mekterovi´c, L. Brki´cbrki´c, and M. Horvat, ‘Scaling Automated Programming Assessment Systems’, *Electronics 2023, Vol. 12, Page 942*, vol. 12, no. 4, p. 942, Feb. 2023, doi: 10.3390/ELECTRONICS12040942.

[14] A. Agrawal and B. Reed, ‘A survey on grading format of automated grading tools for programming assignments’, *ICERI2022 Proceedings*, vol. 1, pp. 7506–7514, Dec. 2022, doi: 10.21125/iceri.2022.1912.

[15] J. L. Subirats, G. Luque-Polo, and R. M. Luque-Baena, ‘Conducting Final Programming Exams with Auto-Grading and Code Evaluation Tools’, *Lecture Notes*

in Networks and Systems, vol. 957 LNNS, pp. 269–278, 2024, doi: 10.1007/978-3-031-75016-8_25.

[16] 'University of Kragujevac Digital Archive: AN AUTOMATED GRADING FRAMEWORK FOR THE MOBILE DEVELOPMENT PROGRAMMING LANGUAGE KOTLIN'. Accessed: Jan. 24, 2025. [Online]. Available: https://scidar.kg.ac.rs/handle/123456789/19059