A Comprehensive Platform for Code Submission, Assignment Analysis, and Personalized Learning in Programming Education

1st Ninad More
Dept. of Computer Engineering
Pune Institute of Computer Technology

2nd Peeyush Kulgude
Dept. of Computer Engineering
Pune Institute of Computer Technology

3rd Pranay Agrawal
Dept. of Computer Engineering
Pune Institute of Computer Technology

4th Prof. Dipali Kadam
Dept. of Computer Engineering
Pune Institute of Computer Technology

***Abstract*—The increasing demand for efficient and scalable solutions in programming education has led to the development of KodeKraken, a web platform designed to streamline code submission, perform detailed assignment analysis, and provide personalized learning experiences for students. This paper presents the features and design of KodeKraken, focusing on automated code checking, plagiarism detection, and differentiation of students based on performance to classify them as slow or fast learners. Additionally, we discuss how KodeKraken enables instructors to monitor student progress and provide targeted feedback, ultimately improving both teaching and learning outcomes. Future work involves expanding the platform's capabilities with advanced machine learning models to further personalize the learning experience.**

***Index Terms*—Code Submission, Assignment Analysis, Plagiarism Detection, Personalized Learning, Slow Learners,**

I. Introduction

In the rapidly evolving landscape of education, there is an increasing need for advanced tools that enhance student assessment and feedback mechanisms. As education shifts towards more digital and remote environments, web-based platforms have become integral for managing assignments, grading, and student evaluation. These platforms not only facilitate the submission of assignments but also provide advanced features like automated grading, plagiarism detection, and personalized learning insights. To support educators and improve student engagement, it is essential to leverage modern technologies such as code editors, version control, and machine learning for effective decision-making in student performance analysis.

Considering this, we observe that assignment evaluation plays a critical role in shaping students' learning paths, as it provides timely feedback and identifies areas of improvement. Machine learning, in particular, has gained prominence in enhancing these systems by automating key tasks such as plagiarism detection and profiling students based on their performance. In plagiarism detection, algorithms analyze the submitted content to identify similarities with existing works, categorizing submissions as authentic or flagged for further review. Meanwhile, code editors with integrated version control track progress and ensure that students follow best practices in software development, providing real-time insights into their coding process.

In the realm of student performance evaluation, machine learning models are used to identify bright and slow learners, allowing for personalized interventions. This categorization is based on a variety of metrics, including assignment submission times, coding accuracy, and engagement patterns. Models such as decision trees and neural networks are being employed to predict student performance and adapt the learning experience accordingly. Additionally, code versioning tools are incorporated to ensure that the learning process is documented, helping educators track improvements and areas where students may struggle.

This paper aims to provide a comprehensive overview of the methodologies employed in educational platforms for student assignment submission, grading, plagiarism detection, and personalized learning insights. By examining the current state of research and identifying key technologies such as machine learning models and code versioning systems, this study aspires to contribute valuable insights into the future of technology-enhanced education.

II. Objective

The primary objectives of KodeKraken are to streamline the student assignment submission and evaluation process while leveraging advanced technologies to enhance the learning experience. This paper aims to evaluate and compare various methodologies employed within KodeKraken, focusing on simplifying the code submission and automated evaluation process through the integration of intuitive code editors and robust version control systems. Additionally, the analysis of student performance is a critical component, utilizing machine learning models to identify slow and fast learners, enabling educators to tailor their support and interventions accordingly. Ensuring academic integrity is paramount; therefore, KodeKraken addresses this through industry-standard plagiarism detection tools that effectively identify potential instances of plagiarism, reinforcing the importance of originality in student work. Moreover, the platform is dedicated to providing personalized feedback based on individual student progress, enhancing the learning experience by offering insights that guide students in their coding journey and helping them address specific areas for improvement. Finally, KodeKraken supports educators in managing student progress effectively by offering data-driven insights that facilitate timely interventions, equipping teachers with the necessary tools to monitor performance and engage with students proactively. Ultimately, this research aspires to provide valuable insights and recommendations for refining educational methodologies within KodeKraken, focusing on the integration of advanced technologies such as machine learning, plagiarism detection, and performance analysis to enhance the overall learning experience for both students and educators.

III. Literature Survey

Plagiarism in programming assignments is a widespread issue in educational institutions, and various techniques have been developed to detect copied or plagiarized code. These methods include token-based, tree-based, and graph-based detection systems, each addressing the challenges of identifying code clones while accounting for variations in coding styles and minor modifications. Integrating such techniques is critical for platforms like KodeKraken, which utilizes automated plagiarism detection tools to ensure academic integrity and encourage original student work. Achieving a balance between detection accuracy and system performance is crucial, especially when designing scalable systems capable of handling large volumes of submissions.

Automating the assessment of programming assignments has become essential to manage the increasing number of student submissions, particularly in large classes. Automatic grading systems that evaluate code against predefined test cases and provide real-time feedback are foundational for platforms aiming to streamline the grading process. Approaches such as static code analysis and dynamic execution testing have been effective in assessing the correctness and quality of code, with the added benefit of offering timely feedback to students, which is instrumental in improving their coding skills. This feedback mechanism is a core feature of platforms designed to enhance student learning outcomes.

The integration of automated testing and grading in learning management systems (LMS) has proven beneficial in managing code submissions. Encouraging students to adopt test-first coding practices, where test cases are written before implementing solutions, has shown to improve programming habits and understanding of code quality. By fostering a focus on correctness and quality, automated systems are able to support teachers in evaluating student submissions efficiently while promoting good programming practices among students.

Detecting plagiarism in programming courses remains a challenge due to the subtle changes students make to evade detection. Robust plagiarism detection tools that analyze structural similarities between code submissions have become essential for ensuring fairness in the assessment process. Regular plagiarism checks help identify instances of copied code and provide a valuable tool for educators to maintain academic honesty in programming courses. By integrating plagiarism detection systems, platforms can effectively reduce instances of academic dishonesty.

Novice programmers often face significant challenges in areas such as syntax, logic, and proper code organization. To address these difficulties, platforms that simplify the submission process while enforcing coding standards can play a pivotal role in helping students improve. Providing structured environments for submitting assignments, along with regular feedback, enables students to focus on overcoming common programming challenges. This structured approach is particularly beneficial for beginners, fostering better learning outcomes.

The use of online integrated development environments (IDEs) in educational settings has streamlined the process of evaluating student code submissions. These IDEs enable real-time code assessment without requiring manual setup, significantly improving the efficiency of code evaluation. By integrating online compilers, platforms can offer immediate feedback to students, helping both students and educators quickly assess code quality and correctness. This real-time feedback capability is a key feature for platforms aiming to optimize the code submission and evaluation process.

Automating the assessment of programming assignments has become essential to manage the increasing number of student submissions, particularly in large classes. Automatic grading systems that evaluate code against predefined test cases and provide real-time feedback are foundational for platforms aiming to streamline the grading process. Approaches such as static code analysis and dynamic execution testing have been effective in assessing the correctness and quality of code, with the added benefit of offering timely feedback to students, which is instrumental in improving their coding skills. This feedback mechanism is a core feature of platforms designed to enhance student learning outcomes.

The integration of automated testing and grading in learning management systems (LMS) has proven beneficial in managing code submissions. Encouraging students to adopt test-first coding practices, where test cases are written before implementing solutions, has shown to improve programming habits and understanding of code quality. By fostering a focus on correctness and quality, automated systems are able to support teachers in evaluating student submissions efficiently while promoting good programming practices among students.

Detecting plagiarism in programming courses remains a challenge due to the subtle changes students make to evade detection. Robust plagiarism detection tools that analyze structural similarities between code submissions have become essential for ensuring fairness in the assessment process. Regular plagiarism checks help identify instances of copied code and provide a valuable tool for educators to maintain academic

honesty in programming courses. By integrating plagiarism detection systems, platforms can effectively reduce instances of academic dishonesty.

The use of online integrated development environments (IDEs) in educational settings has streamlined the process of evaluating student code submissions. These IDEs enable real-time code assessment without requiring manual setup, significantly improving the efficiency of code evaluation. By integrating online compilers, platforms can offer immediate feedback to students, helping both students and educators quickly assess code quality and correctness. This real-time feedback capability is a key feature for platforms aiming to optimize the code submission and evaluation process.

Automating the assessment of programming assignments has become essential to manage the increasing number of student submissions, particularly in large classes. Automatic grading systems that evaluate code against predefined test cases and provide real-time feedback are foundational for platforms aiming to streamline the grading process. Approaches such as static code analysis and dynamic execution testing have been effective in assessing the correctness and quality of code, with the added benefit of offering timely feedback to students, which is instrumental in improving their coding skills. This feedback mechanism is a core feature of platforms designed to enhance student learning outcomes.

The integration of automated testing and grading in learning management systems (LMS) has proven beneficial in managing code submissions. Encouraging students to adopt test-first coding practices, where test cases are written before implementing solutions, has shown to improve programming habits and understanding of code quality. By fostering a focus on correctness and quality, automated systems are able to support teachers in evaluating student submissions efficiently while promoting good programming practices among students.

Discussion

The integration of various machine learning models and tools in KodeKraken plays a crucial role in enhancing both the student and teacher experience, particularly in identifying learner performance and ensuring academic integrity. Notably, Naive Bayes and Linear SVM classifiers have been instrumental in distinguishing between slow and fast learners, with Naive Bayes often outperforming SVM in scenarios where students' performance data is more categorical, such as quiz results and submission timing. Linear SVM, however, excels in handling more complex, multidimensional data such as continuous student progress over time, assignment accuracy, and improvement patterns. Together, these models provide a robust framework for analyzing student learning behaviors and delivering personalized feedback.

In addition to learning analysis, TF-IDF and text summarization techniques have demonstrated their utility in ensuring the integrity of student submissions by detecting plagiarism. TF-IDF calculates the importance of terms in student code submissions, facilitating the detection of similarities across different assignments. This is further enhanced by text summarization methods, which extract critical lines of code for quicker comparison, streamlining the plagiarism detection process.

The frontend of KodeKraken, developed using Dart, integrates seamlessly with the backend built in Java, leveraging API integration to enable efficient data flow between the two. The platform also includes a code editor with versioning capabilities, allowing students to write, submit, and track the evolution of their assignments. Teachers can view students grouped in batches, assess their submissions, and assign grades accordingly. The integration of version control ensures that teachers can track a student's coding process over time, identifying areas of improvement or inconsistency.

These machine learning techniques and tools work synergistically within KodeKraken to not only automate administrative tasks but also enhance educational outcomes. Just as sentiment analysis models have improved decision-making processes in financial markets, the application of advanced machine learning approaches in KodeKraken improves educational evaluation, ensuring timely and accurate assessment of student progress. The system's ability to combine plagiarism detection, learner classification, and summarization into a unified platform highlights its potential to revolutionize assignment submission and evaluation workflows.

Methodology and Implementation

A. Code Submission Evaluation Pipeline:

- KodeKraken provides an automated code submission and evaluation pipeline that allows students to upload assignments with predefined test cases. The pipeline follows these steps:
  1) Code Upload: Students submit code assignments through the Flutter-based frontend.
  2) Syntax Validation: Initial syntax checks are performed on the client-side before submission.
  3) Execution Environment: The backend (FastAPI) processes submissions using a secure, containerized execution environment to prevent malicious code execution.
  4) Test Case Evaluation: The submitted code is executed against predefined test cases stored in Firestore DB, and results are recorded.
  5) Performance Metrics Calculation: Execution time, memory usage, and correctness of the submission are measured and logged.

B. Plagiarism Detection:

KodeKraken integrates TF-IDF and CodeBERT for plagiarism detection:

- Tokenization Preprocessing: The system removes comments and normalizes code to eliminate superficial differences.

- TF-IDF Scoring: A text-based similarity score is computed to compare code submissions.

- CodeBERT Embeddings: A deep learning model (CodeBERT) is used to extract semantic meaning and compare structurally different but functionally similar code.

- Threshold-Based Classification: Submissions are flagged if their similarity score exceeds a predefined threshold.

C. Student Performance Classification

- Students are classified as slow or fast learners using a KNN model, with plans to transition to Random Forest. The classification is based on the following parameters:

- *Version Time Difference*: The time interval between consecutive code submissions.

- *Plagiarism Score*: Higher scores indicate a greater likelihood of copying solutions.

- *Total Number of Versions*: The number of times a student resubmits a solution.

  Workflow:
  –Data Collection: Student submission history is stored in Firestore DB.
  –Feature Engineering: Extract relevant parameters from submission history.
  –Model Training: KNN is initially used to classify students based on their historical patterns.
  –Model Deployment: The trained model is integrated into the FastAPI backend to generate real-time feedback.

D. Instructor Dashboard

- Assignment Overview: A summary of student submissions and evaluation results.
- Plagiarism Reports: A list of flagged submissions with similarity scores.
- Student Performance Insights: Representation of slow learners and fast learners

CHALLENGES AND FUTURE DIRECTIONS:

Even with the advancements made in KodeKraken, several challenges remain that need to be addressed to fully realize its potential:

Data Availability and Quality: The success of the machine learning models, such as those used for identifying slow and fast learners, heavily depends on the quality and quantity of student data. While KodeKraken processes student assignment data, the models would benefit from more diverse data points, such as behavioral or engagement metrics, to better assess learning patterns and make more informed predictions. Access to high-quality, real-time data on student performance is essential to enhance the model's predictive capabilities.

Model Interpretability: Much like in other fields, the machine learning models used in KodeKraken, especially for classifying learners, operate as "black boxes," making it difficult for educators to understand how the system arrives at certain conclusions. Ensuring that the models used for classifying students as bright or slow learners are interpretable is crucial. Educators need to trust and understand these systems to make informed decisions based on their output.

Handling Diverse Code and Assignment Complexity: Student submissions may vary significantly in complexity, style, and approach. While the current models handle basic code analysis and plagiarism detection, future development should focus on enhancing the platform's ability to handle more complex coding styles and ensure fair assessments across different programming approaches. Additionally, integrating deeper learning models capable of understanding advanced coding concepts or logical structures would be beneficial in providing more accurate feedback and evaluations.

CONCLUSION

This study has highlighted the essential components of KodeKraken in streamlining the process of code submission, plagiarism detection, and personalized learning analysis. The integration of machine learning models, such as Linear SVM and Naive Bayes, has proven to be effective in identifying slow and fast learners, providing valuable insights for educators. Plagiarism detection techniques, using methods like TF-IDF and text summarization, ensure academic integrity while supporting educators in efficiently managing student assignments.

The platform's code editor and versioning features have facilitated smooth code submission, and its integration with machine learning models for learner analysis has demonstrated the potential for a more personalized educational experience. The combined approach of advanced plagiarism detection and learning assessment helps teachers offer targeted feedback, which is crucial for fostering student growth.

In conclusion, KodeKraken provides a robust framework for automating assignment submissions, monitoring student progress, and ensuring fair evaluations. Future improvements should focus on enhancing data quality and model interpretability to make the platform even more reliable, while also expanding the scope of learning indicators to better capture student performance nuances.

REFERENCES

[1] Roy, C., and Cordy, J. R. (2007). A Survey of Source Code Clone Detection Research. Computer Science Review, 2(3), 137-158

[2] Joy, M., and Luck, M. (1999). Plagiarism in Programming Assignments. IEEE Transactions on Education, 42(2), 129-133

[3] Lahtinen, E., Ala-Mutka, K., and Järvinen, H. M. (2005). A Study of the Difficulties of Novice Programmers. ACM SIGCSE Bulletin, 37(3), 14-18.

[4] Edwards, S. H. (2003). Teaching software testing: Automatic grading meets test-first coding. Proceedings of the 35th SIGCSE Technical Symposium on Computer Science Education, 245-249

[5] Leinonen, J., Väätäjä, H., and Pöllänen, P. (2016). Online IDEs in Programming Education: Usage Experiences and Comparison. Proceedings of the 16th Koli Calling International Conference on Computing Education Research, 25-34

[6] Hanks, B., and McDowell, C. (2008). Program quality with pair programming in CS1. Proceedings of the 39th SIGCSE Technical Symposium on Computer Science Education, 205-209

[7] V.L. Miguéi, A. Freitas, P.J.V. Garcia and A. Silva, ”Early segmentation of students according to their academic performance: A predictive modelling approach,” Decision Support System,vol. 6, no. 5, pp. 65-78, 2018.

[8] P. Kamal and S. Ahuja,”An ensemble-based model for prediction of academic performance of students in undergrad professional course,” Journal of Engineering Design and Technology, vol. 98, pp. 654-672, 2019

[9] Chitra, A., and Rajkumar, A. (n.d.). Plagiarism detection using machine learning-based paraphrase recognizer, 2015

[10] Sangeeta, K., Naveen Babu, G. V. S. S., and Madhuri, G. (2020). Classification and prediction of slow learners using machine learning algorithms. International Journal of Computer Trends and Technology (IJCTT), 68(2). © IJCTT Journal.