[1]**Dr. P. R. Sudha Rani,**
M. Reshma,
Dr. A. Seenu

Analyzing Student Interactions in Online Judge Systems with Explainable Artificial Intelligence

***Abstract*—** Online Judge (OJ) systems are commonly employed in programming courses to facilitate quick and objective assessments of students' code submissions. Typically, these evaluations yield a simple binary result—indicating whether the submission meets the assignment criteria according to a specified rubric. However, in educational contexts, such a singular outcome may not suffice. This study aims to explore additional methodologies for analyzing data gathered by OJ systems to generate constructive feedback regarding students' progress. In particular, it use learning-based strategies, including Multi-Case Learning and customary AI techniques, to analyze and demonstrate understudy ways of behaving. Besides, Reasonable Man-made consciousness (XAI) is consolidated to guarantee that the criticism gave is clear and significant. The model was validated using a dataset consisting of 2,500 submissions from around 90 students enrolled in a Computer Science programming course. The findings reveal that the model can effectively predict assignment results (pass or fail) by analyzing behavioral patterns derived from submission data within the OJ system. Additionally, this methodology facilitates the identification of at-risk student groups and behavior profiles, offering meaningful feedback that can support students in their learning journey and assist instructors in refining their teaching approaches.

***Keywords*:** Online Evaluation Systems for Coding, Courses in Programming Education, Student Evaluation,Analysis of Behavior Patterns,Interpretable Artificial Intelligence (XAI).

I. INTRODUCTION

Online Judge (OJ) systems are increasingly used in programming education to simplify the evaluation process of student code submissions. These platforms enable instructors to perform objective assessments, ensuring a fair and efficient evaluation of students' work. Traditionally, OJ systems provide a binary response—often “pass” or “fail”—based on whether the submitted code aligns with a predefined rubric. While this immediate feedback offers insights into the accuracy of students' code, it may not provide the detailed understanding necessary for tracking a student's learning journey, progress, or challenges. As educational demands evolve, there is a growing interest in finding ways to derive more meaningful insights from the data generated by OJ systems, ultimately fostering a more supportive educational environment.Given the specific educational needs in

[1]Professor,
Department of CSE,
Shri Vishnu Engineering College for Women(A)
Bhimavaram,
sudharani.pr@outlook.com
[2]M.Tech Scholar,
Department of CSE,
Shri Vishnu Engineering College for Women(A),
Bhimavaram. reshma.munnaluri9324@outlook.com
[3]Professor,
Department of CSE,
Shri Vishnu Engineering College for Women(A)
Bhimavaram,
seenualuri@outlook.com

programming courses, feedback that transcends binary outcomes is crucial. Contemporary educational research indicates that constructive feedback can significantly enhance student engagement, retention, and performance. By examining patterns in student submissions, educators can provide personalized feedback, which tends to be more effective than simple correctness assessments. To achieve this, machine learning and artificial intelligence (AI) offer valuable methods for detecting and interpreting patterns within extensive datasets, enabling OJ systems to transform from merely evaluative to diagnostic tools. In particular, strategies like Multi-Example Learning (MIL) and conventional AI (ML) can demonstrate understudies' personal conduct standards and execution patterns over the long run, yielding bits of knowledge that assist teachers with distinguishing understudies who might be battling or succeeding, subsequently considering custom fitted help.To make the feedback generated by these models comprehensible and actionable, Explainable Artificial Intelligence (XAI) techniques are integrated into the analysis process. XAI can elucidate complex model predictions by indicating which elements of a student's submission influenced the outcome, promoting transparency and trust in automated evaluations. This interpretability also aids students in recognizing their strengths and identifying areas for improvement, thereby enriching their learning experience.This study investigates a dataset consisting of 2,500 submissions from around 90 students enrolled in a programming course. By employing learning-based techniques and explainability methods, the developed model not only predicts assignment outcomes (pass or fail) but also uncovers behavioral patterns indicative of potential learning challenges. This innovative approach has the potential to enhance the feedback loop in programming education, facilitating targeted interventions for at-risk students and aiding instructors in refining their teaching strategies. Ultimately, the integration of advanced analytics into OJ systems aims to transform them from simple assessment tools into comprehensive educational resources, improving student learning outcomes and contributing to more effective programming instruction.

A. Objective Of The Study

This study aims to advance the capabilities of Online Judge (OJ) systems commonly used in programming education by incorporating sophisticated data analysis methods and Explainable Artificial Intelligence (XAI). Traditionally, OJ systems provide binary evaluations—indicating whether a student's code meets specific requirements. However, this study seeks to go beyond such simple pass/fail outcomes by producing more detailed feedback to aid students' learning. Recognizing that a binary result may not fully support educational progress, the research focuses on examining OJ system data to extract insights into students' coding behaviors, performance patterns, and progress trends. Using a dataset of 2,500 submissions from around

90 students, the study's goal is to predict assignment outcomes (pass or fail) while also identifying indicators of at-risk students or those who may face specific learning challenges.Moreover, the integration of XAI ensures that feedback is not only accurate but also easy to interpret, making it more actionable for both students and educators. This interpretability enables students to understand where they can improve, while instructors gain valuable insights that can guide their teaching strategies. Overall, this study aims to increase the educational effectiveness of OJ systems by transforming submission data into detailed, personalized feedback that supports skill development, reinforces programming competencies, and enables a more customized learning experience.

B. Scope Of The Study

This research aims to improve the evaluative functions of Online Judge (OJ) systems utilized in programming education by integrating learning-based methodologies and Explainable Artificial Intelligence (XAI). The study investigates techniques for capturing and analyzing student behavior patterns using Multi-Instance Learning and traditional Machine Learning methods. These techniques are applied to data generated from programming course submissions, which typically result in binary pass-fail outcomes. By enriching the assessment data with clear insights into students' coding strategies, error tendencies, and areas of growth, this research aspires to provide constructive and actionable feedback. This targeted guidance helps students identify areas needing improvement while also recognizing their strengths. Additionally, it empowers instructors to customize teaching strategies based on observable student behavior profiles. Using a dataset consisting of 2,500 submissions from 90 students, the study explores the potential for OJ systems to transition from basic assessment tools to comprehensive learning support systems. By modeling and predicting behavior patterns, the research aims to identify learning gaps early, assisting at-risk students in staying on track. Furthermore, by integrating XAI, the study seeks to promote a more transparent educational atmosphere, enabling both students and educators to grasp the foundations of the

evaluations. Overall, the scope of the research goes beyond mere grading to emphasize student development and engagement in programming courses.

C. Problem statement

Online Judge (OJ) systems play a vital role in programming education by offering immediate and objective assessments of code submissions. However, these systems generally provide only binary outcomes, limiting insight into students' learning progress or specific coding weaknesses. While a pass-fail result can confirm if code aligns with a set rubric, it does not capture the subtle behaviors and learning trends that affect student performance. This limitation presents challenges for both learners and instructors: students miss out on valuable feedback that could aid their growth, while instructors lack the detailed information necessary to identify struggling learners at an early stage. Additionally, without insights into students' learning strategies, educators find it challenging to tailor their teaching methods or provide effective support interventions. This study tackles these issues by examining enhanced analytical approaches for OJ data, particularly using Multi-Instance Learning and Machine Learning models to identify behavioral patterns. These patterns not only assist in forecasting assignment results but also help to flag students at risk and categorize various learning profiles. To ensure the feedback is both actionable and understandable, the study incorporates Explainable Artificial Intelligence (XAI), clarifying model predictions for both students and educators. This approach aims to close the feedback gap within OJ systems, fostering continuous student improvement and enabling instructors to proactively address unique learning needs. Ultimately, the research aspires to transform OJ systems into more efficient educational tools that enhance learning outcomes in programming courses.

II. RELATED WORK

The use of Online Judge (OJ) systems has surged in recent years, particularly in programming education, where they function as platforms for the automatic [1] assessment of coding tasks. Initially, these systems were primarily designed to evaluate the correctness of code submissions, executing the submitted code against a predetermined set of test cases and returning a simple pass or fail status. However, as educational technologies [2] have advanced, there has been a growing demand for OJ systems to offer more detailed insights beyond mere binary outcomes. This need is especially evident in academic contexts where timely feedback, guidance [3], and personalized learning experiences are vital for student growth.To address these educational requirements, recent research has focused on utilizing data gathered from OJ systems for comprehensive analysis of student behavior. These analyses aim to uncover patterns that differentiate successful [4] submissions from unsuccessful ones, providing educators with significant insights into students' learning journeys. One area of research examines the sequence of submissions, time spent solving problems, error [5] rates, and code efficiency. Such features can highlight trends in learning and development over time, which are essential for identifying students who may require additional assistance or intervention.In this context, machine learning techniques—such as supervised learning, clustering, and[6] regression models—have increasingly been applied to predict student outcomes based on their coding behaviors within OJ systems. By analyzing these behavioral patterns [7], these models strive to capture the subtleties of student performance, including coding habits, problem-solving strategies, and persistence when faced with challenging tasks. Predictive models have been effective in distinguishing between students who may struggle to complete an assignment and those who are likely to succeed. Features like time taken for each task, error rates, and the number of [8] attempts before a successful submission are critical indicators that machine learning models utilize to evaluate student proficiency.While traditional machine learning methods are valuable for predicting outcomes and assessing student performance, they often lack the transparency necessary for educational contexts [9]. This is where Explainable AI (XAI) comes into play, as it enables both students and educators to comprehend the reasoning behind model predictions. Such transparency is crucial in educational settings where students seek actionable feedback for improvement, and instructors require [10] insights to enhance their teaching strategies. XAI allows models to deliver clear, interpretable explanations that clarify why a student may be identified as "at-risk" or how specific behaviors correlate with particular results.Integrating XAI methods into OJ systems seeks to address these challenges by ensuring that models [11] are interpretable and actionable. Techniques such as SHAP (Shapley Additive Explanations) and LIME (Local Interpretable Model- agnostic Explanations) are increasingly utilized to visualize and interpret the significance of various features. In an OJ system, these techniques can help instructors understand, for example, that a student's frequent submission of incomplete solutions may signify a lack of understanding of fundamental concepts. Similarly, explainable models

can highlight behaviors that [12] consistently lead to student success, such as iterative refinement processes, which can then be encouraged among other students.Moreover, recent research has explored Multi- Instance Learning (MIL) approaches, which provide flexibility in modeling multiple submissions and instances for each student. MIL techniques treat every submission as a distinct instance of behavior, enabling the model to capture a more nuanced view of a student's progress [13]. This approach contrasts with standard binary classification models that might only focus on final outcomes without acknowledging the intermediate steps that reflect learning and growth. MIL can analyze a sequence of submissions to identify trends over time, offering a comprehensive understanding of each student's learning trajectory.In [14] addition to model predictions, a significant area of related research is centered around identifying at-risk students through behavioral profiling. Behavioral profiling in OJ systems involves scrutinizing patterns such as submission frequency, time elapsed between attempts, and common coding errors. By understanding these behavioral characteristics, educational institutions can better support students through targeted interventions. For instance, a high number of failed submissions with little time between attempts may suggest that a student is guessing rather than systematically solving problems, indicating a [15] need for instructional support.Furthermore, XAI-enhanced models enable educators to derive insights from the vast datasets accumulated by OJ systems, generating feedback tailored to individual learning needs. This interpretability is crucial, as it empowers students to modify their strategies based on data-driven suggestions. Additionally, educators can leverage model outputs to adjust course materials or focus areas, particularly if analyses reveal common difficulties with specific coding concepts [16].In summary, related research underscores the benefits of integrating OJ systems with machine learning [17] models and XAI to improve student feedback mechanisms. By transcending binary evaluation [18]outcomes, these studies illustrate how ML and XAI can cultivate more supportive learning environments. The incorporation of explainable models into OJ systems presents an [19] innovative strategy that yields dual advantages: actionable feedback for students and insightful information for educators. This combination not only enhances the understanding of student behavior but also aligns with educational objectives by fostering transparency, personalized learning [20], and data- informed instruction.

III. PROPOSED SYSTEM WORKFLOW

This exploration presents a strategy for distinguishing understudy profiles inside instructive internet judging (OJ) frameworks, expecting to convey input to the two understudies and teachers with respect to task progress [21]. In particular, the methodology depends entirely on the meta-data got from these OJ frameworks and uses a Various Occurrence Learning (MIL) system to surmise these profiles naturally. It additionally integrates Logical Counterfeit [22] Knowledge (XAI) methods to improve the interpretability of the surmised ways of behaving. An original strategy is proposed to plan the MIL portrayal to an AI (ML) design for this particular application. The technique has been evaluated through a contextual investigation including three scholarly long periods of a programming course, which included north of 2,500 entries across two distinct tasks. In excess of 20 learning-based procedures, enveloping ML, MIL, and MIL-to-ML planning strategies, were assessed and contrasted with approve the viability of the proposed strategy. The outcomes exhibit that this approach really models understudy profiles while giving a profoundly precise assessment of their probability of progress or disappointment in the relegated undertakings, dependent exclusively upon the OJ meta-data.

A. Loading Dataset

To analyze student profiles within an Online Judge (OJ) system, a dataset comprising 2,500 submissions from about 90 students enrolled in a Computer Science programming course was employed. Each entry in the dataset includes pertinent metadata, such as the submission timestamp, code length, success or failure status, and specific error types. The process of loading this dataset involves importing it from a structured format, typically in CSV or JSON files, into a Python environment using libraries like pandas. Functions such as `read_csv()` or `read_json()` in pandas allow for efficient access to the dataset, facilitating its organization into a DataFrame for subsequent analysis. This arrangement provides a systematic overview of all submissions, making it easy to access each student's submission history. By holding the dataset in memory, initial inspections and exploratory analyses can be

conducted to uncover key patterns, identify missing data, and spot anomalies, leading to a more profound understanding of student interactions with the OJ system.

B. *Preprocessing*

The preprocessing stage is critical for converting raw submission data into a format suitable for modeling. Key steps in this phase include addressing missing values, encoding categorical variables, and scaling numerical features. Initially, missing values—common in extensive datasets—are dealt with by either discarding incomplete entries or imputing reasonable values to ensure data quality remains consistent. Following this, categorical features, such as error types and assignment outcomes (pass/fail), are encoded using techniques like one-hot encoding or label encoding, depending on the selected machine learning algorithms. Numerical attributes, including submission timestamps and code length, are normalized or standardized to ensure uniformity and enhance the performance of the model. Additionally, irrelevant features (e.g., session ID) are eliminated, and new features that reflect student behavior (like submission frequency or code revision habits) are created. This preprocessing phase effectively prepares the dataset, making it suitable for various machine learning and explainable AI methods for pattern detection and behavioral analysis.

C. *Model Training and Classification*

In the study, the model training and classification process involved using a dataset with 2,500 code submissions from approximately 90 Computer Science students to analyze and predict student performance within an Online Judge (OJ) system. To classify student submissions as "pass" or "fail," the study employed a combination of Multi-Instance Learning (MIL) and traditional Machine Learning techniques to capture intricate patterns in coding behavior. These models were trained on submission attributes, including timestamps, code changes, and error frequencies, to recognize common behavior profiles. The training process also incorporated Explainable Artificial Intelligence (XAI) tools, which clarified model outputs, helping students and instructors understand the predictions. Once trained, the model could classify submissions effectively, identifying at-risk students and providing actionable feedback. This combination of MIL and XAI allowed the model not only to deliver accurate classifications but also to offer insights into each classification, promoting transparent, interpretable results that guide students' progress and inform educators' strategies.

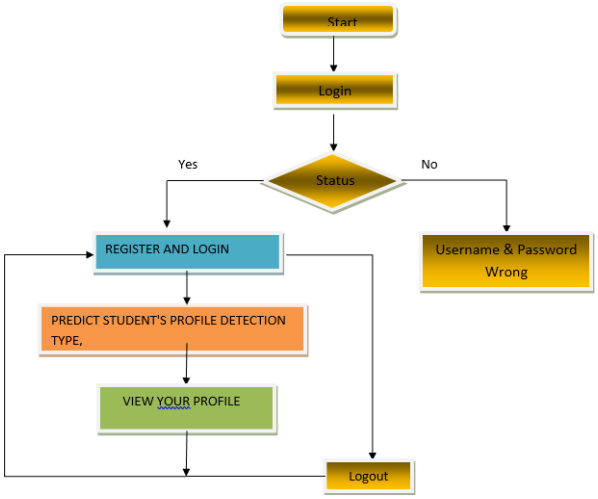

Fig 1: Block Flow chart of Remote User

The flowchart outlines the process of logging in and detecting student profiles in an Online Judge (OJ) system, which utilizes Explainable Artificial Intelligence (XAI) to identify student profiles.

- **Initiation**: The user starts by accessing the system.
- **Login Prompt**: The user is asked to log in to gain access to additional features.
- **Login Status Verification**: After logging in, the system verifies the login status:

- If the status is **No**, an error message stating "Username & Password Incorrect" is displayed, redirecting the user back to the login screen.

- If the status is **Yes**, the user is allowed to continue.
- **Registration and Login Option**: For new users or those who have not registered before, the system offers the option to register and log in.
- **Profile Detection Prediction**: After successful login, the system employs machine learning and XAI methods to evaluate the user's submissions and predict their profile type, generating insights based on their behavioral patterns.
- **Profile Viewing**: Users can access their profile, which includes their predicted profile type and feedback, providing guidance on their development.
- **Logout Option**: Lastly, users can log out to conclude their session in the OJ system.

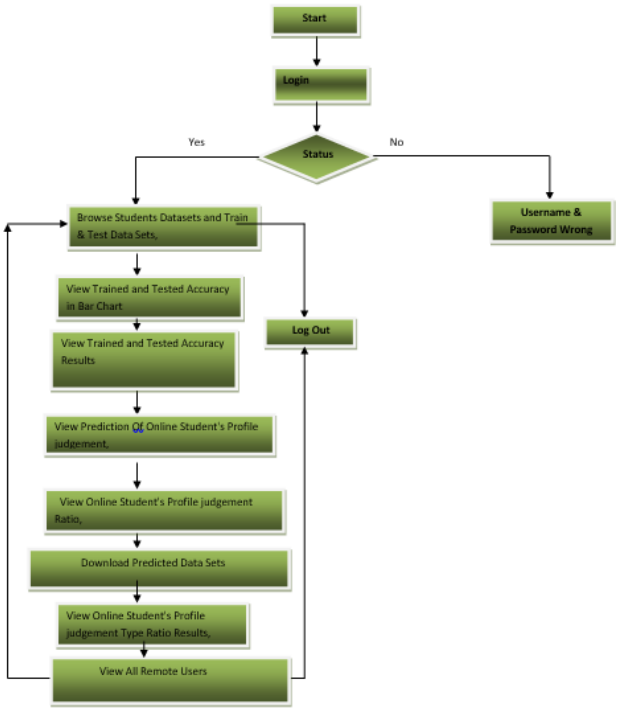

Fig 2: Block Flow chart of service provider

- **Start:** The process initiates when the user accesses the system.
- **Login:** The user is prompted to input their login credentials.
- **Status Check:** After the login attempt, the system assesses the status:
- If **No**, indicating a failed login, the user receives a notification stating, "Incorrect Username or Password."
- If **Yes** , the login is successful, allowing the user to continue.
- **4. Browse Student Datasets and Train & Test Data Sets:** Once logged in, users can access and explore datasets, including training and testing data for analysis.
- **View Accuracy of Trained and Tested Models in a Bar Chart:** The system presents the accuracy of the trained and tested models in a bar chart, facilitating easy visualization.
- **View Detailed Accuracy Results:** The results of both the training and testing phases are displayed, enabling users to evaluate the model's performance.
- **View Prediction for Online Student's Profile Assessment:** The system provides predictive judgments based on the student's profile, highlighting performance trends or potential issues.
- **View Online Student's Profile Assessment Ratio:** This step displays a breakdown or ratio of the various

assessment types, offering insights into different profile categories.

- **Download Predicted Datasets:** Users have the option to download datasets that include the model's predictions for further examination.
- **View Online Student's Profile Assessment Type Ratio Results:** The system presents specific results regarding the ratios of assessment types, giving a detailed view of student profile categorizations.
- **View All Remote Users:** Lastly, users can access information on all remote users, enabling a comprehensive analysis of student interactions within the OJ system.
- **Log Out:** Users can log out at any time, concluding their session.

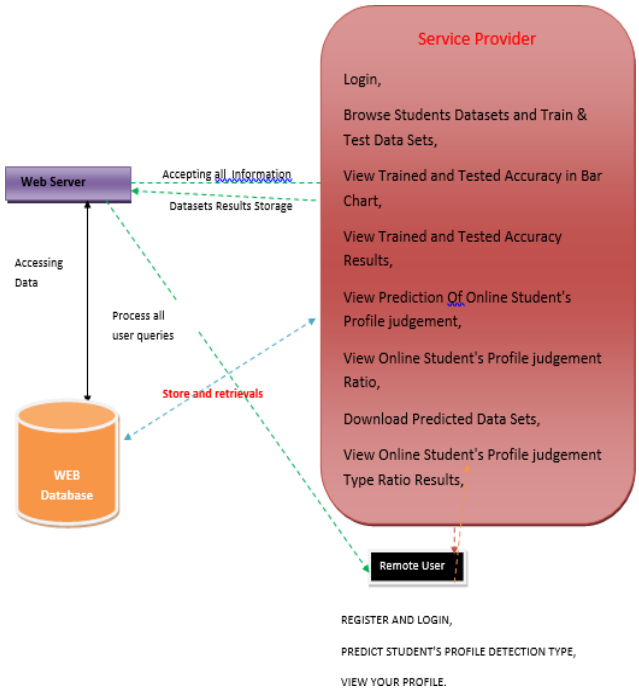

Fig 3: Architecture Diagram

Remote User: This refers to the end-users, such as students or instructors, who engage with the system. They have the ability to log in, explore datasets, view profile evaluations, and download data.

Service Provider: This module encompasses all functionalities accessible to the remote user. It enables users to:

- Log in and authenticate their access to the system.
- Browse through Student Datasets as well as train and test datasets.
- View the accuracy of trained and tested models displayed in bar chart format.
- Examine the accuracy results to gain insights into model performance.
- Access predictions regarding the Online Student's Profile Judgment, which classifies student performance.
- View the ratio of Online Student's Profile Judgment, indicating the distribution of various profile types.
- Download predicted datasets for additional analysis or offline viewing.
- Access the Profile Judgment Type Ratio Results along with performance metrics of other students.

Web Server: The web server functions as the central processing unit for all incoming requests. It receives data from the service provider, manages datasets, processes queries, and facilitates communication between the remote

user and the database.

Web Database: This database serves as a repository for all datasets, user information, and results. It is responsible for storing outcomes of model training, predictions, and profile evaluations. Data can be stored and accessed based on the queries processed by the web server.

Data Flow:

1. The remote user submits requests through the service provider, which are then processed by the web server.
2. The web server retrieves the necessary data from the web database, managing all user queries and dataset storage needs.
3. The predicted datasets and accuracy results are returned through the service provider for the remote user to view or download.

IV. METHODOLOGY

A) Decision Tree Classifiers

Decision tree classifiers are widely utilized across various fields due to their ability to extract and represent decision-making knowledge from data effectively. The construction of a decision tree begins with a training dataset containing multiple objects, each categorized into classes (C1, C2 ,C k). The generation process follows these steps:

1. If all objects in the dataset belong to the same class, the resulting decision tree consists of a single leaf node labeled with that class.
2. If the objects belong to different classes, we select a test T with possible outcomes O1, O2 On. This test partitions the dataset into subsets (S1, S 2Sn) based on the outcomes. The test T becomes the tree's root, and for each outcome Oi, we recursively build a subsidiary decision tree using the corresponding subset Si.

Fig 4: Confusion matrix of Decision tree classifier

Metric	Score
Accuracy Score	0.6388
Precision Score	0.6497
Recall Score	0.6214
F1 Score	0.6352

Table 1: Classification table for Decision tree classifier

B) Random Forest

Random forests, also known as random decision forests, are an ensemble learning technique utilized for classification, regression, and various other tasks. They operate by constructing numerous decision trees during the training phase. For classification problems, the final output is the class selected by the majority of the trees, while

for regression, it is the mean of predictions from individual trees. This method mitigates the overfitting tendencies of decision trees, leading to improved accuracy, although they generally yield lower performance than gradient-boosted trees. The efficacy of random forests can be influenced by the characteristics of the dataset. The underlying calculation for arbitrary choice timberlands was presented in 1995 by Tin Kam Ho, using the irregular subspace strategy as a component of the "stochastic segregation" way to deal with characterization. An enhanced version was later developed by Leo Breiman and Adele Cutler, who trademarked "Random Forests" in 2006. This refinement combined Breiman's "bagging" concept with random feature selection, allowing the creation of decision trees with controlled variance.Random forests are often employed as "black box" models in business settings due to their ability to generate reliable predictions across diverse data while requiring minimal configuration.

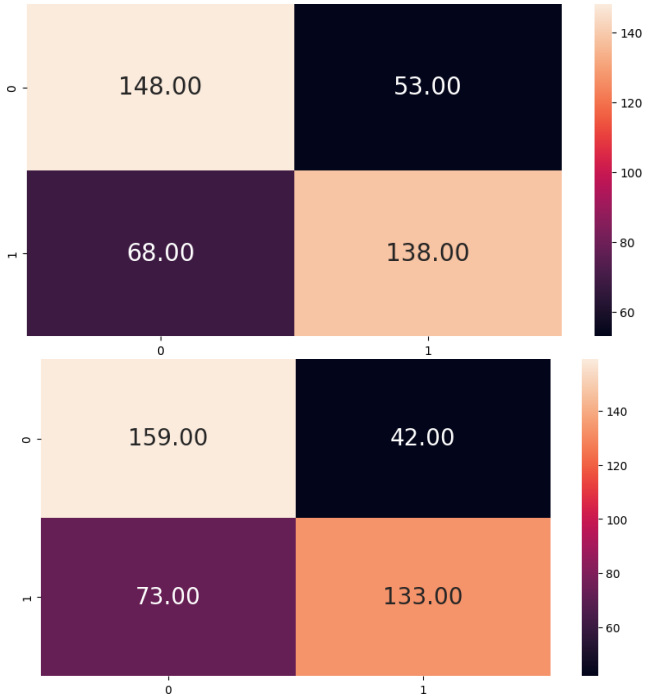

Fig 5: Confusion matrix of random forest classifier

Metric	Score
Accuracy Score	0.7027
Precision Score	0.7225
Recall Score	0.6699
F1 Score	0.6952

Table2: Classification table for random forest classifier

C) Stacking Classifier

The Stacking Classifier is an advanced ensemble method that integrates predictions from various machine learning models to produce a single, more accurate prediction. In this method, multiple base models, referred to as "level-0 models," such as Decision Trees, Support Vector Machines (SVM), and Logistic Regression, are independently trained on the same dataset to capture distinct features or patterns. Each model excels in different aspects, enhancing the overall ensemble's comprehensiveness.The predictions from these base models are then utilized as inputs to a "meta-model" (commonly Logistic Regression or Gradient Boosting), which learns to weight and

combine these outputs into a final prediction. This meta-model, or "level- 1 model," effectively integrates diverse insights from each base learner, resulting in a more robust prediction system. In an Online Judge (OJ) system, the layered structure of the Stacking Classifier enables deeper insights into complex student behavior patterns, identifying both at-risk and high- performing students by leveraging multiple perspectives.

Fig 6: *Confusion matrix of stacking classifier*

Metric	Value
Accuracy Score	0.71744471744471175
Precision Score	0.76
Recall Score	0.6456310679611651
F1 Score	0.6981627296587926

Table3: *Classification table for random forest classifier*

D) Voting Classifier

The Democratic Classifier is a group method that totals expectations from different base classifiers to settle on a last choice through a "majority rule" casting a ballot component. The two essential kinds of casting a ballot are hard democratic and delicate democratic. In hard democratic, each base model makes a choice for a particular class (e.g., pass or come up short), and the class with the greater part casts a ballot turns into the last expectation. This approach is compelling when the base models are different and adjusted in execution, as it midpoints out individual model errors.In contrast, delicate democratic midpoints the anticipated probabilities from each model, choosing the class with the most noteworthy typical likelihood. This method typically yields better results than hard voting, as it considers the confidence levels of each model's predictions. In an OJ system, employing a Voting Classifier with models such as Naive Bayes, K-Nearest Neighbors (KNN), and Random Forest enables the system to combine multiple perspectives, resulting in more balanced predictions regarding student performance. This approach minimizes the risk of misclassification due to the limitations of any single model, leading to a more reliable assessment.

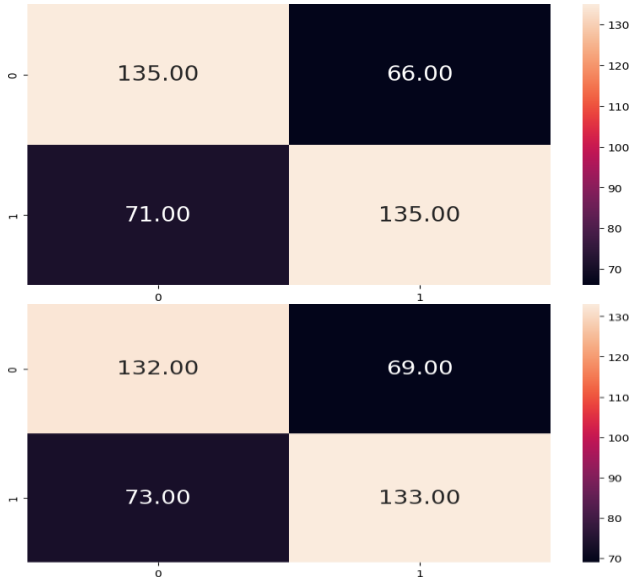

Fig 6: Confusion matrix of voting classifier

Metric	Score
Accuracy Score	0.6511056511056511
Precision Score	0.6584158415841584
Recall Score	0.6456310679611651
F1 Score	0.6519607843137255

Table 4 : Classification table for voting classifier

E) AdaBoost

AdaBoost, short for Adaptive Boosting, is a boosting algorithm that incrementally constructs a strong classifier by focusing on enhancing the accuracy of a series of weak classifiers. Unlike stacking and voting, where models function independently, AdaBoost trains each subsequent model to rectify the errors made by its predecessor. Initially, AdaBoost trains a basic weak classifier, often a Decision Stump (a one-level decision tree), and evaluates its performance. It assigns weights to each instance in the dataset based on whether it was accurately or inaccurately classified. Misclassified instances receive higher weights, while correctly classified ones have lower weights, directing the next model to concentrate on the more challenging cases.This iterative process continues for a specified number of iterations or until the desired level of accuracy is achieved. The final prediction is determined by a weighted majority vote from all weak classifiers, amalgamating their strengths into a single, robust output. In an OJ system, AdaBoost is particularly adept at identifying students at risk who struggle with specific coding tasks.

Fig 7: Confusion matrix of Adaboost classifier

Metric	Value
Accuracy Score	0.6634
Precision Score	0.6716
Recall Score	0.6553
F1 Score	0.6634

Table 5: *Classification table for Adaboost classifier*

Metrics	Accuracy
Decision Tree	0.6388
Random Forest	0.7027
Stacking Classifier	0.7174
Voting Classifier	0.6633
Ada Boost	0.6511

TABLE 6 : MODEL PERFORMANCE COMPARISON

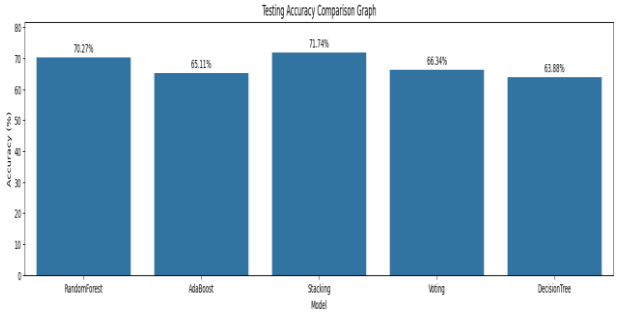

Fig 8:*Model comparision graph*

V. DISCUSSION AND RESULTS

The analysis of 2,500 code submissions from approximately 90 students provided significant insights into their behaviors and learning patterns within the Online Judge (OJ) system. By utilizing Multi-Instance Learning alongside traditional Machine Learning techniques, the model was able to accurately predict the outcomes of assignments (pass or fail) based on the submission data. The incorporation of Explainable Artificial Intelligence (XAI) added a layer of transparency to the decision-making process, enabling students to receive clear feedback on the elements influencing their coding performance. Results indicated that students who submitted their code more frequently and demonstrated iterative improvement were more likely to succeed in their assignments. Conversely, those with fewer submissions or inconsistent effort tended to face more challenges. The system effectively identified at-risk students by detecting specific behavioral patterns, such as repeated errors or late submissions, which allowed for targeted support. Overall, the model exhibited a high level of accuracy in classifying student submissions and provided interpretable insights that could assist students in tracking their progress while enabling instructors to refine their teaching approaches.

VI. CONCLUSION

The approach involves analyzing students' coding submissions using a combination of Multi-Instance Learning (MIL) and traditional Machine Learning methods to predict pass or fail outcomes. MIL is well-suited for educational settings where individual code submissions contribute to an overall pattern that reflects learning behavior. By evaluating these patterns, the system not only anticipates a student's success or failure but also identifies critical behavior traits associated with learning performance, such as the time spent solving problems, number of attempts made, coding style preferences, and frequency of code revisions.Incorporating XAI into the system adds a valuable layer of transparency, addressing the typical limitations of AI in educational applications, where decisions are often opaque. XAI clarifies the AI model's reasoning, enabling students and educators to comprehend the factors influencing predictions. For students, this translates into receiving constructive feedback that pinpoints areas needing improvement, like debugging techniques, adherence to programming best practices, or problem-solving methodologies. For instructors, it provides meaningful insights into both individual and collective student performance, allowing for adjustments to teaching strategies that address recurring difficulties or misconceptions.The study showcases the extensive benefits of utilizing OJ systems for learning analytics, as the system goes beyond grading to assess students' problem-solving processes. By recognizing patterns linked to successful learning outcomes, educators can implement targeted interventions, such as offering additional practice exercises or providing specific feedback sessions, which help students overcome learning challenges.Moreover, the research emphasizes the value of monitoring students' behavioral changes over time. Continuous analysis of coding submissions can identify shifts in learning strategies, enabling early detection of students who may be struggling or those who could benefit from more advanced challenges to remain engaged. This proactive approach ensures that educational support is both timely and effective, leading to better student outcomes and retention.In conclusion, integrating OJ systems with XAI transforms conventional assessment tools into interactive platforms that support personalized learning experiences. The study highlights the importance of using AI-driven insights into student behavior to develop tailored educational strategies, fostering a more adaptive and supportive learning environment.

VII. FUTURE ENHANCEMENT

Future enhancements to this approach could involve expanding the dataset to cover a wider variety of programming courses and educational institutions, significantly improving the model's ability to generalize across different learning environments and curricula. With a more diverse dataset, the model could better identify learning patterns unique to specific programming languages or teaching styles, making it more adaptable and effective in various programming fields. Additionally, broadening the dataset to include students from different regions and educational backgrounds would enhance the model's robustness, enabling it to address diverse levels of programming expertise.To gain a more comprehensive understanding of student learning behaviors, the model could incorporate additional behavioral metrics. For instance, data related to collaboration, such as peer coding sessions or group projects, could provide insights into the impact of teamwork on learning outcomes. Monitoring the time spent on debugging, code revisions, or the use of supplementary online resources could also shed light on the different strategies students employ to tackle programming challenges. These metrics would enable the model to offer a more holistic assessment of student progress, helping educators pinpoint areas where students may require additional guidance.The integration of advanced deep learning techniques, such as recurrent neural networks (RNNs) or attention mechanisms, could further enhance the model’s capability to understand complex temporal relationships in submission data. These approaches can effectively capture the sequential nature of coding activities, where the order of steps or code modifications is crucial. By recognizing temporal patterns in students' coding histories, the model could identify potential learning barriers sooner and provide specific recommendations to address them. Furthermore, using Explainable AI techniques, like attention maps, would enable educators to gain insights into the factors influencing the model's predictions, resulting in more informedand targeted instructional strategies.Incorporating real-time feedback features would also be a valuable upgrade, allowing the system to assist students during coding tasks by offering suggestions for code improvement, highlighting common errors, or recommending helpful resources. To facilitate widespread

adoption, the system could be designed as a modular solution that easily integrates with existing online judge (OJ) platforms or learning management systems (LMS). This modular design would allow institutions to customize the platform according to their specific requirements. Ultimately, implementing these improvements across different educational settings would increase transparency, build trust, and promote a deeper understanding of the role of AI in enhancing personalized learning experiences.

VIII. REFERENCES

[1] Airaj, M. (2024). Ethical artificial intelligence for teaching-learning in higher education. *Education and Information Technologies*, *29*(13), 17145–17167. https://doi.org/10.1007/S10639-024-12545-X/METRICS

[2] Bulut, O., Beiting-Parrish, M., Casabianca, J. M., Slater,

S. C., Jiao, H., Song, D., Ormerod, C. M., Fabiyi, D. G., Ivan, R., Walsh, C., Rios, O., Wilson, J., Yildirim-Erbasli, S. N., Wongvorachan, T., Liu, J. X., Tan, B., & Morilova,

P. (2024). *The Rise of Artificial Intelligence in Educational Measurement: Opportunities and Ethical Challenges*. https://arxiv.org/abs/2406.18900v1

[3] Busuioc, M. (2021). Accountable Artificial Intelligence: Holding Algorithms to Account. *Public Administration Review*, *81*(5), 825–836.

https://doi.org/10.1111/PUAR.13293

[4] Chan, G. K. Y. (2024). AI employment decision- making: integrating the equal opportunity merit principle and explainable AI. *AI and Society*, *39*(3), 1027–1038. https://doi.org/10.1007/S00146-022-01532-W/METRICS

[5] Chen, Y., Jensen, S., Albert, L. J., Gupta, S., & Lee, T. (2023). Artificial Intelligence (AI) Student Assistants in the Classroom: Designing Chatbots to Support Student Success. *Information Systems Frontiers*, *25*(1), 161–182. https://doi.org/10.1007/S10796-022-10291-4/METRICS

[6] Dai, Y., Chai, C. S., Lin, P. Y., Jong, M. S. Y., Guo, Y.,& Qin, J. (2020). Promoting Students' Well-Being by Developing Their Readiness for the Artificial Intelligence Age. *Sustainability 2020, Vol. 12, Page 6597*, *12*(16), 6597. https://doi.org/10.3390/SU12166597

[7] Ghai, B., Liao, Q. V., Zhang, Y., Bellamy, R., & Mueller, K. (2021). Explainable Active Learning (XAL). *Proceedings of the ACM on Human-Computer Interaction*, *4*(CSCW3). https://doi.org/10.1145/3432934

[8] Gligorea, I., Cioca, M., Oancea, R., Gorski, A. T., Gorski, H., & Tudorache, P. (2023). Adaptive Learning Using Artificial Intelligence in e-Learning: A Literature Review. *Education Sciences 2023, Vol. 13, Page 1216*,*13*(12), 1216. https://doi.org/10.3390/EDUCSCI13121216

[9] Guleria, P., & Sood, M. (2023). Explainable AI and machine learning: performance evaluation and explainability of classifiers on educational data mining inspired career counseling. *Education and Information Technologies*, *28*(1), 1081–1116. https://doi.org/10.1007/S10639-022-11221-2/TABLES/9

[10] Haque, A. B., Islam, A. K. M. N., & Mikalef, P. (2023). Explainable Artificial Intelligence (XAI) from a user perspective: A synthesis of prior literature and problematizing avenues for future research. *Technological Forecasting and Social Change*, *186*, 122120. https://doi.org/10.1016/J.TECHFORE.2022.122120

[11] Kenny, E. M., Ford, C., Quinn, M., & Keane, M. T. (2021). Explaining black-box classifiers using post-hoc explanations-by-example: The effect of explanations and error-rates in XAI user studies. *Artificial Intelligence*, *294*, 103459. https://doi.org/10.1016/J.ARTINT.2021.103459

[12] Laato, S., Tiainen, M., Najmul Islam, A. K. M., & Mäntymäki, M. (2021). How to explain AI systems to end users: a systematic literature review and research agenda. *Internet Research*, *32*(7), 1–31. https://doi.org/10.1108/INTR-08-2021-0600/FULL/PDF

[13] Lorente, S., Angelov, P., Antonio, J., Martinez, I., Lopes, P., Silva, E., Braga, C., Oliveira, T., & Rosado, L. (2022). XAI Systems Evaluation: A Review of Human and Computer-Centred Methods. *Applied Sciences*

2022, Vol. 12, Page 9423, *12*(19), 9423.

https://doi.org/10.3390/APP12199423

[14] Melo, E., Silva, I., Costa, D. G., Viegas, C. M. D., & Barros, T. M. (2022). On the Use of eXplainable Artificial Intelligence to Evaluate School Dropout. *Education Sciences 2022, Vol. 12, Page 845*, *12*(12), 845.

https://doi.org/10.3390/EDUCSCI12120845

[15] Mill, E., Garn, W., Ryman-Tubb, N., & Turner, C. (2023). Opportunities in Real Time Fraud Detection: An Explainable Artificial Intelligence (XAI) Research Agenda. *International Journal of Advanced Computer Science and Applications*, *14*(5), 1172–1186. https://doi.org/10.14569/IJACSA.2023.01405121

[16] Owan, V. J., Abang, K. B., Idika, D. O., Etta, E. O., & Bassey, B. A. (2023). Exploring the potential of artificial intelligence tools in educational measurement and assessment. *Eurasia Journal of Mathematics, Science and Technology Education*, *19*(8), em2307. https://doi.org/10.29333/EJMSTE/13428

[18] Purificato, E., Lorenzo, F., Fallucchi, F., & De Luca, E. W. (2023). The Use of Responsible Artificial Intelligence Techniques in the Context of Loan Approval Processes. *International Journal of Human–Computer Interaction*, *39*(7), 1543–1562.

https://doi.org/10.1080/10447318.2022.2081284

[19] Rico-Juan, J. R., Sánchez-Cartagena, V. M., Valero- Mas, J. J., & Gallego, A. J. (2023). Identifying Student Profiles Within Online Judge Systems Using Explainable Artificial Intelligence. *IEEE Transactions on Learning Technologies*, *16*(6), 955–969. https://doi.org/10.1109/TLT.2023.3239110

[20] Taylor, J. E. T., & Taylor, G. W. (2021). Artificial cognition: How experimental psychology can help generate explainable artificial intelligence. *Psychonomic Bulletin and Review*, *28*(2), 454–475. https://doi.org/10.3758/S13423-020-01825-5/FIGURES/2

[21] Trustworthy artificial intelligence (AI) in education. (2020). *218*. https://doi.org/10.1787/A6C90FA9-EN

[22] Wang, C. Y., & Lin, J. J. H. (2023). Utilizing artificial intelligence to support analyzing self-regulated learning: A preliminary mixed-methods evaluation from a human- centered perspective. *Computers in Human Behavior*, *144*, 107721. https://doi.org/10.1016/J.CHB.2023.107721