JETIR.ORG ISSN: 2349-5162 | ESTD Year : 2014 | Monthly Issue

JOURNAL OF EMERGING TECHNOLOGIES AND INNOVATIVE RESEARCH (JETIR)

An International Scholarly Open Access, Peer-reviewed, Refereed Journal

EXPLAINABLE AI: PROVIDING STUDENT FEEDBACK VIA ONLINE JUDGE

#1**PASUMARTHI AMRUTHA,** M.Tech Student,

#2**DARAPU UMA**, Assistant Professor,

Department of CSE,

PRAGATI ENGINEERING COLLEGE, SURAMPALEM, AP, INDIA.

Abstract: Online Judge (OJ) tools are extensively utilized in programming educational institutions due to their ability to evaluate student submissions swiftly and impartially. This evaluation method typically results in a singular outcome derived from a rubric that often indicates whether the submission has met the assignment's requirements. However, it would be beneficial for both educators and learners to have a greater influence over the overall assessment of the project, as this automated evaluation may not fully capture the nuances of academic performance. This initiative aims to tackle this concern by optimizing the use of OJ data and providing prompt feedback to both teachers and students. By employing learning-based methodologies to simulate student behavior, including Multi-Instance Learning and fundamental machine learning techniques, the accuracy of assessments can be improved. Additionally, Explainable AI is being explored to furnish individuals with comprehensible feedback. The hypothesis was evaluated through a case study involving 2,500 submissions from approximately 90 students enrolled in a computer science program with a focus on programming. The model's ability to accurately predict a student's outcome, such as the likelihood of passing or failing an assignment based solely on behavioral patterns identified in OJ submissions, lends credence to the hypothesis. Moreover, this approach can uncover additional pertinent information, such as student profiles and at-risk groups, thereby facilitating feedback for both educators and learners.

Index Terms - Online Judge, Explainable AI, Assignment, Feedback.

1. INTRODUCTION

The term "online judge" (OJ) was first introduced to describe software that automatically evaluates programming assignments. These systems typically manifest as online testing platforms that gather source code, compile it, assess the results, and generate scores based on various criteria[2]. Such automated systems have garnered significant interest in two interconnected areas: (i) higher education teaching settings, and (ii) programming contests and competitions[3]. This study primarily concentrates on the latter, specifically computer courses provided by colleges and institutions that confer degrees in computer science.

OJ systems offer significant benefits to the educational sector by addressing the primary difficulties associated with manual grading of assignments. Unlike traditional grading methods, which are often labor-intensive and susceptible to errors, these systems provide automatic correction of submissions, regardless of user volume. Additionally, they foster an environment conducive to effective learning by encouraging

healthy competition among students [5].

While OJ systems have clear benefits, their functionality is limited to detecting if the code submitted by the student or instructor matches the assignment requirements [6]. Neither the student nor the instructor receive feedback on the submission. The OJ system's data can be utilized to improve education by independently getting more information on student practices or behavioral patterns related to their work performance (or lack thereof). In this case, Educational Data Mining (EDM), a method that leverages data collected from academic institutions to identify patterns and estimate results, could be used. Machine learning appears to be one of the most essential tools in this industry. Examples include [8]'s efforts, which dealt with measuring a teacher's efficacy; [9]'s method, which worked with predicting students' grades; and [10], which investigated detecting problems in peer-reviewed works. In this initiative, EDM is used to provide educators and students with automatic feedback on programming class assignments given in OJ systems.

When utilizing an OJ to assess programming assignments, students are typically allowed to submit an endless number of them in a specified time period. For most tasks, the quality of the student's best submission decides their final grade. It is probable that information routinely utilized in EDM, such as attendance logs or performance grades, will not be available at that time [9]. Furthermore, because of the possibility for prejudice, some information that is exploited to predict a student's academic success—like their financial condition or past performance in class—may be unethical [11].

It would be beneficial to identify students who might be at danger before the assignment's due date, especially considering the minimal information that is already accessible. An EDM technique was established to provide two separate results: (i) a collection of various student profiles that provide educators and learners with beneficial input, and (ii) a higher likelihood of success for new students. This outcome was made feasible by meta-data that was gathered from the submission procedure. Examples of this meta-data include the number of code submission attempts and the date of the first submission. Remember that by providing meaningful feedback on the activity's progress, this type of data may be utilized, among other things, to make sure that particular activities are not unreasonably tough and to dissuade pupils from adopting unfavorable attitudes.

This problem, which is characterized as a Multi-Instance Learning (MIL) job [12], includes determining the type of student based on the collection of code contributions that the student has displayed. An overview of the concept of a "bag," defined as a collection of unknown items with a common label, is offered by the learning framework [13]. The study [15] that compares MIL and ML for student academic performance prediction verifies the efficacy of MIL in EDM research [14]. The various code contributions that every student in our scenario has generated are housed in each of these tote bags. Students are granted either positive or negative code entries according to how well they score on the OJ system exam.

However, the nature of ML and MIL techniques is frequently "black box" [16], which hampers their application in feedback-based systems. As a result, there is growing interest in the topic of Explainable Artificial Intelligence (XAI). Its purpose is to give human-interpretable tools for computer model judgments in order to address this issue [17]. Compared to the ML domain, the MIL domain has given XAI significantly less attention [18].

This study aims to introduce a method for gaining insights into the various types of students participating in virtual learning environments. The objective is to furnish both educators and students with feedback regarding the progress of their work. Specifically, the methodology relies exclusively on the meta-data extracted from these Online Judging (OJ) systems. Furthermore, an investigation is conducted on a Multiple Instance Learning (MIL) framework designed to automatically identify these characteristics, alongside Explainable Artificial Intelligence (XAI) techniques that enhance the interpretability of predicted behaviors. XAI can be integrated into the MIL framework by proposing a novel approach to transform the MIL representation into a suitable Machine Learning (ML) representation for this purpose. A case study encompassing three academic years of a programming course, which included approximately 2,500 submissions for two distinct objectives, was employed to assess the proposed methodology. A comparative analysis was performed using ML, MIL, and MIL-to-ML mapping techniques across more than twenty learning-based strategies to evaluate the effectiveness of the proposed plan. The findings indicate that the

plan effectively models the student user profile and provides a highly accurate prediction of students' chances of passing or failing the current assignment based on the meta-information derived from the OJ systems.

2. LITERATURE SURVEY

Online judges are algorithms designed to analyze user submitted algorithm source code on a frequent basis. The code then goes through standard testing and compilation. Online tribunals are becoming more popular in certain scenarios. Now let's look at what these instruments can achieve. Online compilers, development platforms integrated into other custom systems, systems for arranging programming competitions, systems for education and recruitment, and systems for solving data mining challenges were all categorized. It also offers an official portrayal of an online court system and a full discussion of the standard work evaluation methodology applied by these systems. Finally, we present a brief example of an online scoring system that has been recommended as a solution to tough optimization problems. We also look at the outcomes of competitions using this methodology. The competition showed that online judging systems, enhanced by public recommendations, can accurately and successfully tackle tough tasks in research and industry [2].

Nowadays, a substantial part of students use programming online judges (POJ) in addition to their coding classes. This is because the POJ software offers a wide variety of programming projects that students can finish. In addition to giving instructions, a POJ will carry out automatic code assembly and verification. Students often claim that they know too much about POJ. It is difficult to choose the most important issue when there are many to take into account. Since ITSs don't include essential information, integrating pre-existing POJs into them can be difficult. The goal of this project is to give students access to a collaborative filtering recommendation system that finds programming problems appropriate for their level of expertise, thereby mitigating the issue of too much information. The roles of the students are more accurately represented the more often the user-problem matrix is used. This makes it easier to identify nearby communities, which results in a more accurate recommendation. Furthermore, a step known as "data preprocessing" has been added to the suggestion process to take into consideration unforeseen user behaviors that can have an impact on the recommendations that are generated. An actual POJ dataset case study shows that the process performs better than previous approaches [3].

Practicing is one of the most crucial aspects of learning to code. Manually determining programming assignments is time consuming and error-prone. The fact that so many students enroll in programming classes exacerbates the problem even further. As a result, fewer programming assignments are typically assigned to students than are strictly required. Automating the evaluation method might be one solution to this problem, allowing students to submit programming tasks online and receive prompt responses. This study examines how the National University of Singapore's School of Computing employs Online Judge, an automated evaluation tool. More than 700 students enrolled in the relevant course, which is a required first-year curriculum that teaches them the fundamentals of programming. It describes the resulting fears, as well as the children's actions and emotions. The Online Judge is used extensively in both an introductory high school course and an advanced college course [4].

In recent years, most programs to introduce active learning into the classroom have leveraged information and communication technology (ICT). Taking use of the flexibility provided by ICT, a large number of these programs have prioritized collaborative situations that can be carried out in person, electronically, or in a hybrid format. In addition to the advantages of virtual collaboration, competitive learning may facilitate learning. The current study will look into the relationship between academic achievement, employment happiness, and competitive learning among telecoms school students. QUEST was used in the college communications network course to increase student involvement and competitiveness during study sessions. A large amount of study has been undertaken on the effects of competitive e-learning technology on university students' academic satisfaction and accomplishment [5].

Most online programmers respond to the submitted code with remarks such as "good enough" or "not good enough." This rule may have an impact on learning environments by banning the use of online judges, who might normally provide more helpful comments to teachers and students who submit incorrect codes.

However, it may be valuable for competitive platforms or recruitment. It would be easy to demonstrate the code's inadequacies. Even if one case isn't unique enough, there may be a sloppy code alteration. This research study looks into the idea of extracting wrong remarks made by judge users in order to create a small number of private test cases that will be valuable to the majority of prospective users. Before coming up with an answer, we reviewed the test files, created a two-part graph, and solved a Set Cover problem with integer linear programming. 100 questions from Jutge.org were utilized to evaluate our responses. These tests demonstrate our method's efficacy, generalizability, and capacity to generate high-quality findings [6].

There are various OJ concepts in the scientific literature, most of which are connected to educational contexts and academic institutions, as well as potential real-world applications. This type of system includes the URI system developed by Universidad Regional Integrada [21], the Peking University Online Judge (POJ) implemented by [22], which is specifically designed for C++ courses, the Course Maker system developed by the University of Nottingham to address general programming tasks [23], and the Youxue Online Judge (YOJ) [24].

OJ systems, in where competitors must solve algorithmic challenges, are being extensively researched for both educational and coding competition applications. This set of examples was evaluated by the International Collegiate Programming Contest [26] and the UVa Informatics Olympiads [27].

Early identification of students who are struggling in a course is seen as critical in education since it provides the instructor with information about the specific support that the student requires. Numerous academic research have examined how external and internal factors influence the previously mentioned issues.

Previous academic research has generally focused on students' socioeconomic background or previous course grades in relation to external factors [11]. The reader is advised to examine the work of [28] for a detailed analysis of these topics, as it is unrelated to the primary goal of this investigation. Numerous other methodologies have been described for the intrinsic components (using task performance data inside a course), as they yield logical estimations. Several notable examples address this issue: [29], which examines this issue across online learning platforms; [30], which investigates preventative failure detection in the Moodle platform; [31], which uses clicker test data to approximate this information in settings involving peer-based instruction; and [9], which uses course attendance as a predictor of academic achievement over the year.

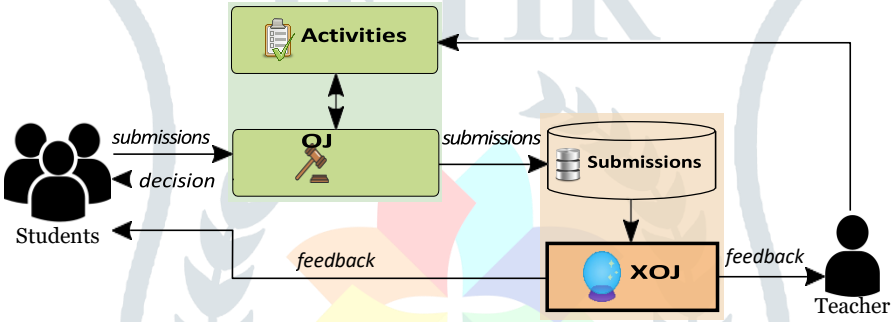
Using programming classes as an example, tailored heuristics are less complex and more efficient than OJ tools. Error Quotient [32] and its enhanced counterpart, Repeated Error Density [33], assess this by examining syntactic errors that occur throughout the compilation process. The Watwin Scoring Algorithm [34] penalizes students depending on how long it takes them, relative to their peers, to correct each sort of error, resulting in equivalent outcomes. Created a system for grading more complex interactions, including bug fixes and syntactically accurate code changes. The final example is from [36]. Linear regression is used to detect at-risk infants by examining compilation errors and other factors.

To analyze student profiles, the bulk of machine learning models in use today must be able to effectively interpret massive amounts of complicated data. Most machine learning models require a large amount of data to deliver accurate predictions. If little data is supplied, the model may be less accurate. The quality of the training data retrieved from the input dataset impacts the accuracy of current machine learning models. An incorrectly labeled sample makes it more difficult for the model to make accurate predictions.

From the above survey, the online judge (OJ) system is used to grade the assignments manually, and it takes a long time depending on the number of users. It is also difficult to use a large amount of data to make accurate predictions and also assign labeling. To overcome these problems, the proposed methodology is discussed in Section III.

3. PROPOSED METHODOLOGY

This research presents a methodology for identifying an individual's learning style through on-the-job (OJ) learning techniques while evaluating all pertinent characteristics. This approach facilitates feedback for both educators and learners. The proposal emphasizes the integration of explainable artificial intelligence (XAI) methods to enhance the understanding of expected behaviors, alongside a multiple-instance learning (MIL) framework that enables the system to autonomously identify these profiles. The meta-data produced by these OJ systems is crucial for accomplishing this objective. A new policy is introduced to convert the MIL representation into a machine learning (ML) representation suitable for this task, thereby allowing the application of XAI to the MIL challenge. A three-year case study involving over 2,500 submissions from two distinct projects in a programming course was conducted to evaluate the proposed methodology. To determine the effectiveness of the concept, a review and comparison of more than twenty learning-based strategies utilizing ML, MIL, and MIL to ML mapping techniques were performed. Based solely on the meta-information derived from the OJ, the results indicate that the proposal accurately models the student user profile and provides a highly precise prediction of the students' likelihood of passing or failing the specific assignment. Transparency strategies, for instance, are methods that facilitate a clear understanding of the model's functioning. Post-hoc explanations are theoretical endeavors aimed at elucidating the model's development process. In contrast to transparency-focused strategies, this study prioritizes the latter scenario, thereby negating the necessity for each learning-based model to be specifically tailored to the task at hand.



The proposed method for quantitatively addressing these questions, which includes the following steps, is depicted graphically in Figure 1.

1) The instructor characterizes the various tasks to be settled by the understudies and designs the OJ framework as needs be.

2) The students respond to the given task and present their solutions.

3) The OJ assesses these entries and gives the understudies a remedy mark solely founded on the assessment of the submitted programming codes.

4) Concurrently, these entries are handled by an extra module XOJ in the plan that gives criticism to both the educator who might adjust the difficulty of the undertaking and the understudies who might accordingly change their obligation to the assignment. Note that this component addresses the center component of the work as it is intended to display the client conduct thinking about a regulated learning system.

This proposed system grades the assignments automatically and very quickly. There are few learning based algorithms to predict he student behavior and also evaluate performance of student. The algorithms are described below.

4. ALGORITHMS

There are various machine learning algorithms to discusses the online judge system are listed below.

4.1 Decision tree classifiers

Decision tree methods appear to be applicable in a variety of circumstances. The most essential aspect about them is that they can learn to use the information you offer to draw complex judgments. Training sets can be used to build decision trees. This demonstrates how to construct an object that resembles a collection of items (S) from the classes C1, C2 ,..., Ck.

Step 1If every element in the S decision tree belongs to the same class, the leaf will be labeled with the class Ci.

Step 2. If T is not else given, define it as a test that returns on. The examination divides S into groups S1, S2,..., Sn, with each item in Si representing a single probable outcome in T. This is the case because each element in S has a unique possible outcome for T. The decision tree starts at point T, and the same procedures are used to build a child decision tree on the set Si for each outcome Oi.

4.2 Gradient boosting

Techniques in machine learning, such as gradient boosting, are employed to address various challenges, including both regression and classification tasks. A prediction model is generally formed by combining several weak predictive models, such as decision trees. When a decision tree acts as the base learner, gradient-boosted trees come into play. In most cases, they demonstrate superior performance compared to random forests. Similar to other boosting methods, gradient-boosted trees are built incrementally. However, they surpass earlier techniques by enabling the optimization of any differentiable loss function.

4.3 K-Nearest Neighbors (KNN)

This straightforward yet highly effective categorization method classifies objects according to their similarities. It operates slowly and is non-parametric, meaning it does not "learn" until it encounters a test case. By utilizing the training data, we identify the K-nearest neighbors of the newly categorized data.

4.4 Logistic regression Classifiers

Logistic regression analysis evaluates a collection of categorical independent variables that contribute to the explanation of a categorical dependent variable. The dependent variable is binary, possessing only two possible outcomes: zero and one, or yes and no. The term "logistic regression" is widely recognized to describe this type of analysis. In cases where the dependent variable encompasses three or more categories, such as married, single, divorced, or widowed, multinomial logistic regression is typically utilized. Although multiple regression employs a different dataset to represent the dependent variable, the fundamental methodology remains consistent. Both discriminant analysis and logistic regression serve as effective techniques for distinguishing between categorical response categories. A significant number of statisticians assert that logistic regression generally surpasses discriminant analysis in modeling various scenarios. However, logistic regression encounters limitations when the independent variables do not adhere to a normal distribution, a situation that is not problematic for discriminant analysis.

This tool allows you to do multinomial and binary logistic regression with independent variables that might be lists or numbers. Along with the degree of fit, it shows the regression equation, chance ratios, confidence bounds, probability, and deviation. The full residual analysis includes maps as well as diagnostic residual reports. Using an independent variable subset search, it is possible to find the best regression model with the fewest independent variables. It gives ROC curves and confidence intervals for anticipated values to assist in determining the optimal categorization threshold. You can direct the system to automatically organize rows that were not used in the study to ensure the correctness of your findings.

4.5 Naïve Bayes

The naïve Bayes method is a supervised learning approach that posits that the presence or absence of a particular attribute within a class does not influence the presence or absence of other attributes. Despite this assumption, it has proven to be both effective and practical. In terms of functionality, various supervised learning methods are comparable. The literature provides numerous explanations for this. In this discussion, we will examine a scenario involving representation bias. Classification of items can be performed using

any linear predictor, including logistic regression, naïve Bayes, linear discriminant analysis, and linear support vector machines (SVM). The distinction is made by employing a technique based on learning bias to ascertain the parameter values of the classifier. Although the Naive Bayes classifier is widely utilized in research, it has limitations when applied to real-world applications. The researchers discovered that it had a number of advantages, including ease of building and use, rapid learning on very large databases, intuitive settings, and fairly high accuracy when compared to other systems. However, because no intuitive paradigm is provided, end users are unable to recognize the advantages of this strategy.

As a result, we show the learning process's outcomes in an innovative way. As a result, the technique is easy to comprehend and implement. This lecture's first segment covers some theoretical elements of the naive Bayes classifier. The technique is then applied to a dataset via Tanagra. The results (model parameters) are contrasted and compared to various linear methodologies, including logistic regression, linear discriminant analysis, and linear support vector machines. It is important to highlight that the findings are consistent. Most of the time, one explains why one approach is superior to another. As described in Section 2, we apply a variety of tools on the same dataset (R 2.9.2, Knime 2.1.1, Orange 2.0b, and RapidMiner 4.6.0). Understanding the findings is really important to us.

4.6 Random Forest

Random forests, often referred to as random decision forests, represent a form of ensemble learning characterized by the generation of numerous decision trees during the training phase. These models are applicable for tasks such as regression analysis, classification, and various other applications. The random forest algorithm determines the classification by selecting the class that receives the majority vote from the trees. In the context of regression, it provides the mean or average prediction derived from all the trees. The design of random decision trees aims to mitigate the risk of errors associated with overfitting that can occur in individual decision trees. While random forests generally outperform single decision trees, they tend to be less precise than gradient-boosted trees. However, the effectiveness of random forests can be influenced by the characteristics of the underlying data. The inception of the random choice forest methodology can be attributed to Tin Kam Ho in 1995. Ho indicated that he employed the random subspace method to implement Eugene Kleinberg's "stochastic discrimination" classification approach. In 2006, Adele Cutler and Leo Breiman trademarked the term "Random Forests," introducing an algorithmic enhancement. As of 2019, this trademark is owned by Minitab, Inc. The innovation merges Breiman's "bagging" technique with Ho's random feature selection method, which was subsequently developed independently by Amit and Geman. This integration aims to create a collection of decision trees with regulated variability. Organizations frequently adopt random forests as "black box" models due to their minimal setup requirements and their ability to yield accurate predictions across diverse data sources.

4.7 SVM

A discriminant machine learning methodology is employed to establish a discriminant function that effectively infers labels for new instances derived from an independent and identically distributed (iid) training dataset. This approach is utilized for job classification. In this classification process, a discriminant classification function is utilized to allocate a data point x to a designated class. However, the implementation of generative machine learning techniques necessitates the prior construction of conditional probability distributions. When incorporating outlier detection into the prediction framework, discriminant methods typically require fewer training samples and computational resources compared to generative methods. This is particularly advantageous in multidimensional feature spaces where only posterior probability needs to be considered. The task of determining the equation for a three-dimensional surface that most effectively segregates multiple classes within the feature space is akin to the process of learning a classifier. Support Vector Machines (SVM) address the convex optimization problem in a systematic manner, yielding the optimal hyperplane value. This differs from perceptrons (Genetic Algorithms) or other classification algorithms commonly utilized in machine learning. The parameters for initialization and termination significantly influence the outcomes of perceptron solutions. Training produces distinct SVM model parameters tailored to a specific training set by employing a particular kernel to transform data from the input space into the feature space. Conversely, the Genetic Algorithm classifier and perceptron models

undergo updates during each training iteration. Throughout the training process, both GAs and perceptron's strive to minimize errors, which can be achieved through the integration of multiple hyperplanes.

5. RESULTS AND DISCUSSION

When performing comparisons among various algorithms, they show various accuracy values with respect to the various algorithms. These are listed in the following Table 1. RF scheme with MIL-to-ML mapping is the best performing strategy XAI scheme provides feedback and high recognition rate and Identifies prone-to-fail student groups and profiles, offering valuable feedback

Table: 1 Comparison of various algorithms with their accuracy

S.No.	Name of the Algorithm	Accuracy %
1	Artificial Neural Network	63.0
2	Naïve Bayes	70.5
3	SVM	66.5
4	Logistic Regression	68.0
5	Gradient Boosting Classifier	67.5
6	Decision Tree Classifier	64.0
7	K-Nearest Neighbors Classifier	58.5

The following figure 2 shows the graphical representation of the above table 1.

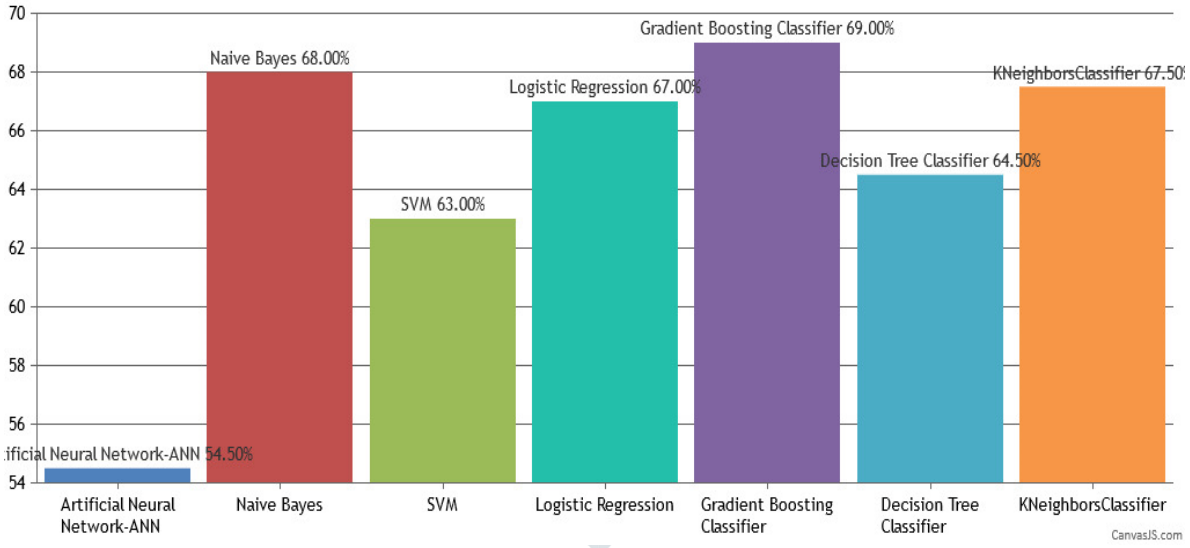


Figure 2: Graphical Representation various algorithms with their accuracy

From figures 3 to 6 shows the resultant screenshots of Online judge system using XAI.

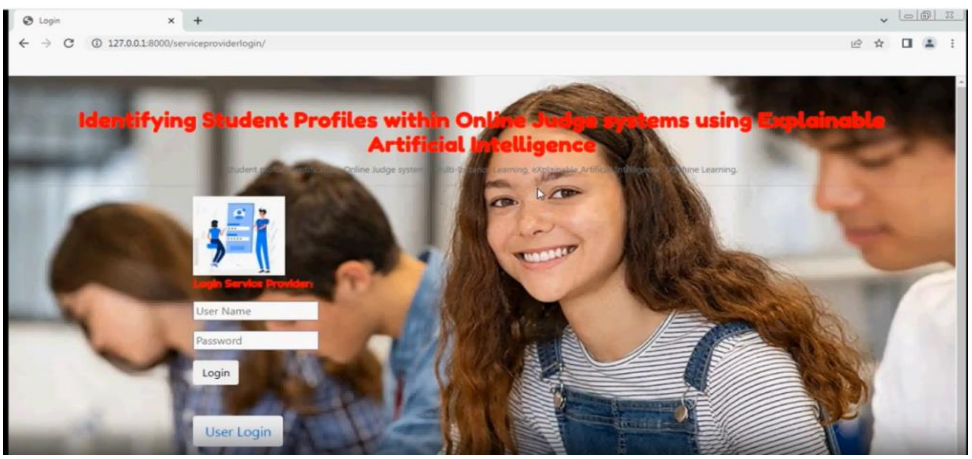


Figure 3: Service Provider Login Page

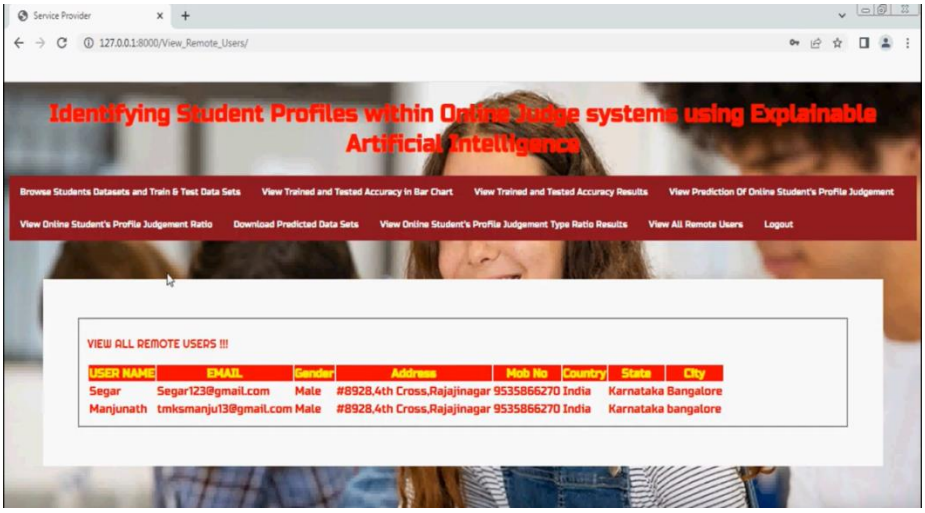


Figure 4: View Remote User Profile

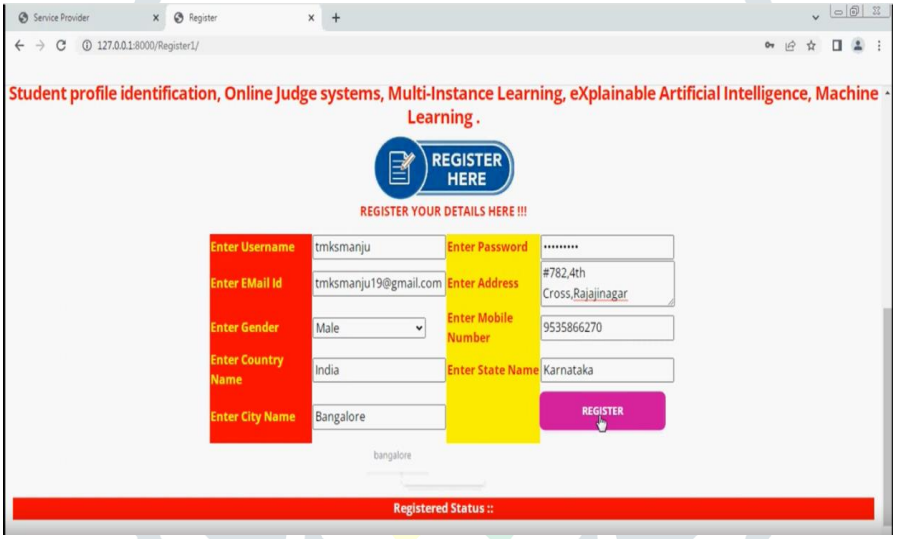


Figure 5: Remote User Register Page

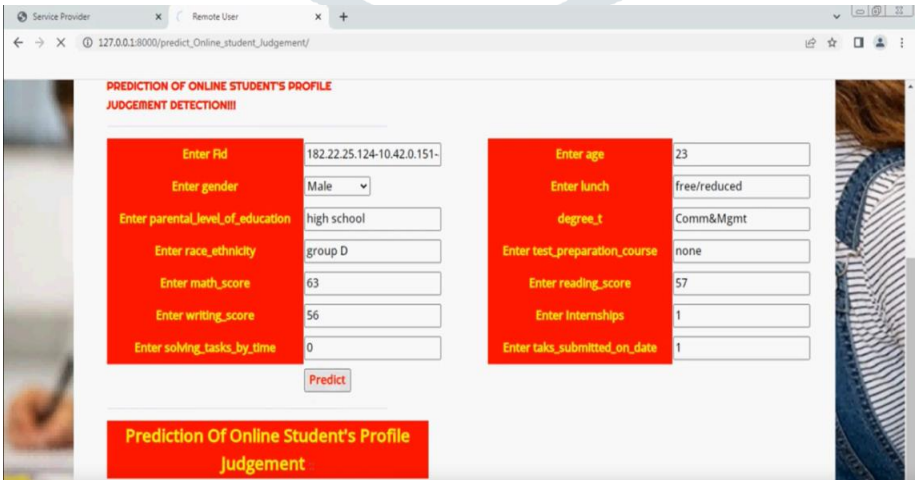


Figure 6: View Prediction Results

6. CONCLUSION

This work discusses about Professors are considering using Online Judge (OJ) tools in programming classrooms to provide fair feedback on student code submissions. However, these systems often only provide feedback on the submission's code, without providing additional information. To address this, a project aims to use Educational Data Mining (EDM) techniques, specifically Explainable Artificial Intelligence (XAI), to collect human-readable feedback. A case study and a computing course for a computer science degree demonstrated the effectiveness of this strategy. The model accurately predicted user outcomes based on submission behavior patterns, identifying students more likely to fail and providing constructive feedback. More research is needed to ensure that the model is accurate. This includes researching more classrooms that use OJ grading systems and expanding on the material provided in the case study. We will also look into how human factor traits acquired from evaluations, such as motivation, self-efficacy, and personality, might be used to improve the system's prediction accuracy.

REFERENCES

1. A. Kurnia, A. Lim, and B. Cheang, “Online judge,” Computers & Education, vol. 36, no. 4, pp. 299–315, 2001.
2. S. Wasik, M. Antczak, J. Badura, A. Laskowski, and T. Sternal, “A survey on online judge systems and their applications,” ACM Computing Surveys (CSUR), vol. 51, no. 1, pp. 1–34, 2018.
3. R. Yera and L. Mart´ınez, “A recommendation approach for programming online judges supported by data preprocessing techniques,” Applied Intelligence, vol. 47, no. 2, pp. 277–290, 2017.
4. B. Cheang, A. Kurnia, A. Lim, and W.-C. Oon, “On automated grading of programming assignments in an academic institution,” Computers & Education, vol. 41, no. 2, pp. 121–131, 2003.
5. L. M. Regueras, E. Verdu, M. F. Munoz, M. A. Perez, J. P. de Castro, and M. J. Verdu, “Effects of competitive e-learning tools on higher education students: A case study,” IEEE Transactions on Education, vol. 52, no. 2, pp. 279–285, 2009.
6. A. Mani, D. Venkataramani, J. Petit Silvestre, and S. Roura Ferret, “Better feedback for educational online judges,” in Proceedings of the 6th International Conference on Computer Supported Education, Volume 2: Barcelona, Spain, 1-3 April, 2014. SciTePress, 2014, pp. 176–183.
7. R. Asif, A. Merceron, S. A. Ali, and N. G. Haider, “Analyzing undergraduate students’ performance using educational data mining,” Computers & Education, vol. 113, pp. 177–194, 2017.
8. X. Zhang and Y. Kang, “Examining and predicting teacher professional development by machine learning methods,” in International Conference on Neural Computing for Advanced Applications. Springer, 2021, pp. 255–269.
9. C. C. Gray and D. Perkins, “Utilizing early engagement and machine learning to predict student outcomes,” Computers & Education, vol. 131, pp. 22–32, 2019.
10. J. R. Rico-Juan, A.-J. Gallego, and J. Calvo-Zaragoza, “Automatic detection of inconsistencies between numerical scores and textual feedback in peer-assessment processes with machine learning,” Computers & Education, vol. 140, p. 103609, 2019.
11. S. Alturki, N. Alturki, and H. Stuckenschmidt, “Using educational data mining to predict students’ academic performance for applying early interventions,” Journal of Information Technology Education: Innovations in Practice, vol. 20, no. 1, pp. 121–137, 2021.
12. J. Foulds and E. Frank, “A review of multi-instance learning assumptions,” The knowledge engineering review, vol. 25, no. 1, pp. 1–25, 2010.
13. M.-L. Zhang, “Generalized multi-instance learning: Problems, algorithms and data sets,” in 2009 WRI Global Congress on Intelligent Systems, vol. 3. IEEE, 2009, pp. 539–543.
14. S. Anupama Kumar and M. N. Vijayalakshmi, Efficiency of Multiinstance Learning in Educational Data Mining. Singapore: Springer, 2018, pp. 47–64.
15. A. Zafra, C. Romero, and S. Ventura, “Multi-instance learning versus singleinstance learning for predicting the student’s performance,” Handbook of Educational Data Mining, pp. 187–200, 2010.
16. T. Komarek, J. Brabec, and P. Somol, “Explainable multiple instance ´ learning with instance selection randomized trees,” in Joint European Conference on Machine Learning and Knowledge Discovery in Databases. Springer, 2021, pp. 715–730.
17. N. Burkart and M. F. Huber, “A survey on the explainability of supervised machine learning,” Journal of Artificial Intelligence Research, vol. 70, pp. 245–317, 2021.

18. A. B. Arrieta, N. D´ıaz-Rodr´ıguez, J. Del Ser, A. Bennetot, S. Tabik, A. Barbado, S. Garc´ıa, S. Gil-Lopez, D. Molina, R. Benjamins ´ et al., "Explainable Artificial Intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI," Information Fusion, vol. 58, pp. 82–115, 2020.
19. J. B. Hext and J. W. Winings, "An automatic grading scheme for simple programming exercises," Commun. ACM, vol. 12, no. 5, p. 272–275, May 1969.
20. R. C. Carrasco, J. R. Rico-Juan, and M. A. Var ´ o, "Aprendizaje de ´ algoritmia mediante desaf˜ıos de programacion," in ´ XVI Jornadas de Ensenanza Universitaria de la Inform ˜ atica ´ . Universidade de Santiago de Compostela. Escola Tecnica Superior d'Enxe ´ nar ˜ ´ıa, 2010, pp. 519– 522.
21. J. L. Bez, N. A. Tonin, and P. R. Rodegheri, "Uri online judge academic: A tool for algorithms and programming classes," in 2014 9th International Conference on Computer Science Education, 2014, pp. 149–152.
22. L. Wen-xin and G. Wei, "Peking university oneline judge and its applications," Journal of Changchun Post and Telecommunication Institute, vol. 2, 2005.
23. C. A. Higgins, G. Gray, P. Symeonidis, and A. Tsintsifas, "Automated assessment and experiences of teaching programming," J. Educ. Resour. Comput., vol. 5, no. 3, p. 5–es, Sep. 2005.
24. H. Sun, B. Li, and M. Jiao, "Yoj: An online judge system designed for programming courses," in 2014 9th International Conference on Computer Science Education, 2014, pp. 812–816.
25. A. Kosowski, M. Małafiejski, and T. Noinski, "Application of an online ´ judge & contester system in academic tuition," in Advances in Web Based Learning – ICWL 2007, H. Leung, F. Li, R. Lau, and Q. Li, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 343–354.
26. R. H. de Boer and C. P. de Campos, "A retrospective overview of international collegiate programming contest data," Data in Brief, vol. 25, p. 104382, 2019.
27. M. A. Revilla, S. Manzoor, and R. Liu, "Competitive learning in informatics: The UVa online judge experience," Olympiads in Informatics, vol. 2, no. 10, pp. 131–148, 2008.
28. S. Alturki, I. Hulpus, , and H. Stuckenschmidt, "Predicting academic outcomes: A survey from 2007 till 2018," Technology, Knowledge and Learning, pp. 1–33, 2020.
29. M. Adnan, A. Habib, J. Ashraf, S. Mussadiq, A. A. Raza, M. Abid, M. Bashir, and S. U. Khan, "Predicting at-risk students at different percentages of course length for early intervention using machine learning models," IEEE Access, vol. 9, pp. 7519–7539, 2021.
30. B. Cabral and A. Figueira, "On the development of a model to prevent ´ failures, built from interactions with moodle," in Advances in Web-Based Learning – ICWL 2019, M. A. Herzog, Z. Kubincova, P. Han, and ´ M. Temperini, Eds. Cham: Springer International Publishing, 2019, pp. 352–356.
31. S. N. Liao, D. Zingaro, K. Thai, C. Alvarado, W. G. Griswold, and L. Porter, "A robust machine learning technique to predict lowperforming students," ACM Trans. Comput. Educ., vol. 19, no. 3, Jan. 2019.
32. M. C. Jadud, "Methods and tools for exploring novice compilation behaviour," in Proceedings of the Second International Workshop on Computing Education Research, ser. ICER '06. New York, NY, USA: Association for Computing Machinery, 2006, p. 73–84.
33. B. A. Becker, "A new metric to quantify repeated compiler errors for novice programmers," in Proceedings of the 2016 ACM Conference on Innovation and Technology in Computer Science Education, ser. ITiCSE '16. New York, NY, USA: Association for Computing Machinery, 2016, p. 296–301.
34. C. Watson, F. W. Li, and J. L. Godwin, "Predicting performance in an introductory programming course by logging and analyzing student programming behavior," in 2013 IEEE 13th International Conference on Advanced Learning Technologies, 2013, pp. 319–323.
35. A. S. Carter, C. D. Hundhausen, and O. Adesope, "The normalized programming state model: Predicting student performance in computing courses based on programming behavior," in Proceedings of the Eleventh Annual International Conference on International Computing Education Research, ser. ICER '15. New York, NY, USA: Association for Computing Machinery, 2015, p. 141–150.
36. E. S. Tabanao, M. M. T. Rodrigo, and M. C. Jadud, "Predicting atrisk novice Java programmers through the analysis of online protocols," in Proceedings of the Seventh International Workshop on Computing Education Research, ser. ICER '11. New York, NY, USA: Association for Computing Machinery, 2011, p. 85–92.