Improved Explainable Educational Data Mining System for Enhancing Programming Skills

BY

MOHAMED ABDULLE MOHAMUD
ID: 0242310004213017

This Report Presented in Partial Fulfillment of the Requirements for the Degree of master of Science in Management Information System

Supervised By

Dr. Sheak Rashed Haider Noori
Professor & Associate Head
Department of CSE
Daffodil International University

DAFFODIL INTERNATIONAL UNIVERSITY

DHAKA, BANGLADESH

JULY 2024

APPROVAL

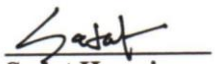
This Thesis titled .**Improved Explainable Educational Data Mining System for Enhancing Programming Skills ,** submitted by **Mohamed Abdulle Mohamud** to the Department of Computer Science and Engineering, Daffodil International University, has been reviewed in terms of style and substance and acknowledged as satisfying for the partial completion of the criteria for the MS in Management Information System degree. The presentation was held on 06 july Saturday, 2024.

BOARD OF EXAMINERS

Narayan Ranjan Chakraborty — **Board Chairman**
Associate Professor and Associate Head
Department of CSE
Faculty of Science & Information Technology
Daffodil International University

Md. Sadekur Rahman — **Internal Examiner 1**
Assistant Professor
Department of CSE
Faculty of Science & Information Technology
Daffodil International University

Dr. Ohidujjaman Tuhin — **Internal Examiner 2**
Assistant Professor
Department of CSE
Faculty of Science & Information Technology
Daffodil International University



Sadat Hossain — **External Examiner 3**
Data Scientist
Risk Management Division
BRAC Bank Limited

DECLARATION

I hereby declare that, this thesis has been done by me under the supervision of **Dr. Sheak Rashed Haider Noor**, Professor & Associate Head, Department of CSE Daffodil International University. I also declare that neither this thesis nor any part of this thesis has been submitted elsewhere for award of any degree or diploma.

Supervised by:

Dr. Sheak Rashed Haider Noori
Professor & Associate Head Department of CSE
Daffodil International University

Submitted by:

Mohamed Abdulle Mohamud
ID: **0242310004213017**
Department of CSE
Daffodil International University

ACKNOWLEDGE.

First and foremost, I would want to express my profound gratitude to the divine power of God for His abundant blessings, which have greatly facilitated the successful completion of my final year thesis.

I would like to express my heartfelt gratitude to **Dr. Sheak Rashed Haider Noori**, an **Associate Professor in the Department of Computer Science** and Engineering at Daffodil International University in Dhaka. During the last phases of completing my thesis, I really appreciate my supervisor's extensive expertise and passionate dedication in providing study opportunities and technical help. The completion of this thesis was made possible by his exceptional patience, intellectual guidance, persistent motivation, firm and meticulous supervision, constructive critique, practical recommendations, thorough examination of several subpar versions, and rectification of any identified issues at every stage.

I have exerted much effort on this thesis. However, the achievement would have been unattainable without the invaluable assistance and support of other individuals. I would want to express my gratitude to those individuals who provided me with the chance to complete my report.

I express my gratitude to all the other students of Daffodil International University who engaged in this discourse while completing their projects.

Finally, I would want to express my sincere gratitude to my parents for their unwavering support and patience.

ABSTRACT

Forecasting student academic performance benefits from the extremely effective method known as educational data mining (EDM), which also helps to find important links within educational data. Evaluating and improving students' programming competency has been the main emphasis of many recent studies. Still, there are chances for constant development in this field. In this work, we provide an improved and understandable Educational Data Mining (EDM) approach for spotting and improving students' programming capacity. This proposed EDM system seeks to investigate a very effective feature engineering approach, a suitable classification technique, and the use of Explainable Artificial Intelligence (XAI) tools for model explanation. We do ablation study to find the best feature engineering method. The categorizing process decides students' current programming state. Six basic Machine Learning (ML) algorithms—decision tree, Support Vector Machine, Random Forest (RF), artificial neural network, Naive Bayes Classifier, k-Nearest Neighbor, and Ensemble method—are the main subjects of this module. Many criteria—including accuracy, precision, recall, f1-score, ROC curve, McNamar test, and others—are used to assess the performance of these algorithms. The experimental results show that among the many models, the Random Forest (RF) and the Stacking-SRDA ensemble technique can classify the students with more accuracy than others. To improve the interpretability of the model, we have finally used XAI technologies like Eli5, SHAPASH, and Local Interpretable Model Agnostic Explanations.

Topics: Programming Performance, Explainable Artificial Intelligence, Machine Learning Educational Data Mining.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

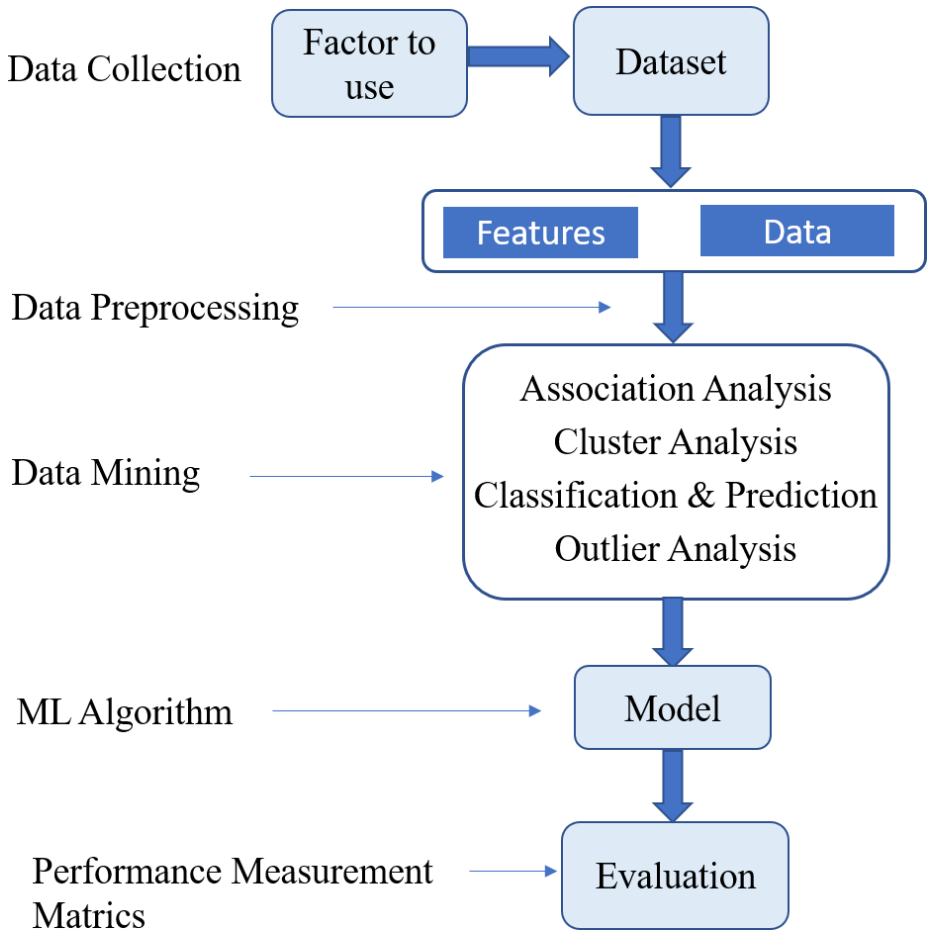
Chapter 1

Introduction

1.1 Introduction

EDM, or educational data mining, is a developing field that encompasses several disciplines and seeks to create approaches for evaluating data gathered from educational settings. EDM employs statistical, machine learning (ML), and methods for data mining (DM) to discover previously undisclosed patterns in educational data [1]. Data mining, sometimes referred to as DM, is the methodical process of dividing a large dataset into separate clusters by detecting patterns within the data. These clusters are then used to extract noteworthy and invaluable information (2). The objective of EDM, or educational data mining, aims to develop tools for evaluating various forms of data in educational contexts to get a more profound comprehension of learners and the learning environments they experience [3]. Studying is the main goal of educational research. this particular kind of data in order to answer the current problems [4]. The ecosystem has the capacity to systematically gather, evaluate, report, collaborate on, and engage with diverse educational data with the aim of consistently enhancing teaching and learning [5].The popularity of EDM has increased in the modern era of education because to the notable enhancement in the standard of the educational system. EDM applications are often used in higher education because they have the ability to enable the development of student-centered solutions and provide crucial real-time predictive information [5, 6]. EDM applications prioritize the improvement of teaching methods via the creation of precise models that assess students' academic achievement and conduct. Figure 1.1 illustrates the essential

elements of the EDM process.The instructional strategy used involves constructing accurate models to analyze students' performance and behavior. Figure 1.1 depicts the



fundamental basis of the EDM process. performance and behavior. Figure 1.1 illustrates the EDM process's foundation

Figure 1.1: Basic EDM Process1

The three main foundations of EDM are computer science, education, and statistics, as seen in figure 1.2. The interaction of these aspects is giving rise to new areas associated with EDM, including computer-based learning environments, machine learning (ML) systems, and learning analytics systems [9].

EDM, such as machine learning (ML) systems, computer-based learning environments, and learning analytics systems [9].

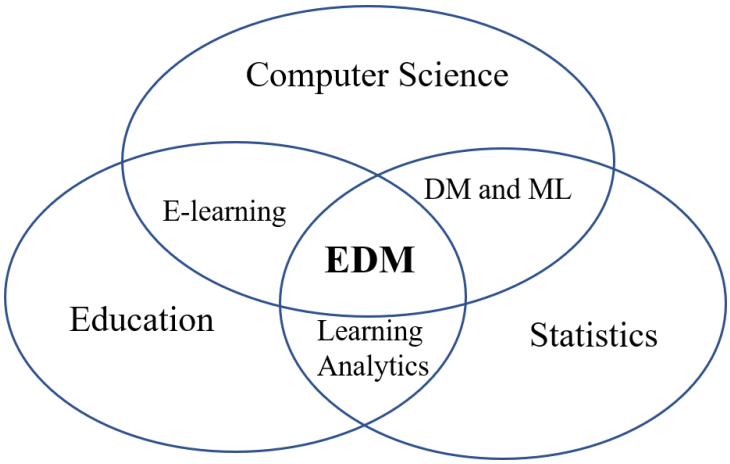


Figure 1.2: Components of EDM2

In recent years, a multitude of distinctive DM methods and classification algorithms have been used. The EDM program is used to augment pedagogical approaches, decrease the incidence of academic underachievement, scrutinize essential attributes, and consider student accomplishment and proficiency. Furthermore, it assists in the creation of accurate predictive models for forecasting student performance [7][8]. In addition to furnishing facts and concepts for the development of the curriculum for the next year, it also facilitates immediate improvement for vulnerable children.

Higher education is crucial for equipping Students equipped with the requisite skills and information to thrive in the highly competitive work market. Employment opportunities in the area of computer science and engineering (CSE) include expanding rapidly, leading to a strong need for expertise in this sector. Students may enhance their programming skills to effectively compete with their peers. Programmers are those who possess expertise in programming, which is a clear indication of their ability to write computer programs. An adept programmer need a varied range of abilities that combine technical and analytical aspects with a considerable level of ingenuity. proficiency in many computer programming languages, Algorithms, problem-solving ability, mathematical prowess, technical expertise, writing skills, creativity, passion for learning code, and other similar attributes are all crucial for any programmer. In this work, we have created an improved Educational

Data Mining (EDM) method to classify student programming ability into four categories: excellent, good, average, or terrible. In addition, we use Explainable Artificial Intelligence (XAI) methods to detect the most important features, enabling less skilled programmers to concentrate on their areas of deficiency and improve their abilities.

1.2 EDM System for Predicting Students' Performance

Improving student performance and the quality of teaching are the primary goals in the education industry. A detailed examination of their past performance records or data may be very beneficial in projecting their success and advising them on areas in which they may use EDM to improve, all in the name of giving kids a high-quality education. The concept of EDM has been used in many recent investigations. To improve the quality of tertiary education, for instance those on academic performance prediction, dropout and student behavior analysis [15][16][17][18][19], and dropout rates. [11][10][12][13][14], and so There have been a few studies published that assess programming skills using the EDM approach. For instance, [38] forecasts undergraduate students' progress in the programming language C using presentations. Predictions on undergraduate students' performance in a certain Java programming language course were made by the authors of [39]. A separate researcher in [40] presents an improved method for skill development and forecasts students' tertiary-level programming ability. However, there are still a number of issues with the current EDM technique in this area. To predict programming performance, the authors of these EDM studies assess a number of significant machine learning classifiers; they did not, however, use data representation techniques. To address these shortcomings Regarding the present educational data mining (EDM) methodology for evaluating students' computer programming abilities, we have developed an improved EDM system that incorporates explainability.

1.3 Motivation of The Thesis

Technology and computers are an aspect of contemporary life. Computer science (CS) and other STEM (science, technology, engineering, and math) fields have seen a rise in interest in recent years. Students are enrolling in computer science courses, often without taking into account their aptitude for or familiarity with the required subject, because of the

positive employment market in computer science and associated fields. The ability to program is seen as the key to success in the field of computer science. Learning computer programming is a difficult and intimidating subject, and some of the more fortunate ones that enrolled in it eventually dropped out. The first computer science course, CS1, is predicted to be successfully completed by 67% of students globally [20]. Several academics have endeavored to ascertain the elements that render computer programming instruction and learning captivating, with the goal of enhancing student motivation and course completion. In particular, computer scientists have been looking at methods of using student data to detect underachievers. Oyelere et al. [21] looked at one such case in which students received programming training with the use of mobile learning. The success of students in a certain college course on programming in C and Java, respectively, was predicted by the authors of [38] and [39]. From this perspective, We have suggested an enhanced Educational Data Mining (EDM) method that aims to categorize students with more precision. and pinpoint the essential components required for them to excel in programming courses. This study will help those who have trouble programming get past their challenges.

1.4 Objectives of The Thesis

The primary aims of the research are as follows:

In order to enhance the precision of predictions, it is advisable to use a feature engineering approach and ensure the appropriate formatting of the data.

To investigate machine learning methods for classification and group learning models in an effort to outperform previous models' accuracy.

To clarify the idea and look at the main factors that determine how students' programming proficiency is categorized

1.5 Contributions of The Thesis

The following are the research's main contributions:

This work presents a very efficient approach to feature engineering. We conduct an ablation research to evaluate the efficacy of the feature engineering approach that has been provided.

We used fundamental machine learning methods in conjunction with an ensemble learning model (Stacking) to construct the model. The Stacking-SRDA method we have presented has shown superior accuracy compared to other approaches. We evaluate the significance level of our proposed Stacking-SRDA model using fundamental machine learning methods.

In addition, we have used the explainable AI (XAI) tools LIME, SHAPASH, and Eli5 to enhance interpretability for our proposed EDM system. We analyze the fundamental components and evaluate the characteristics that influence the prediction capability of the model for a certain category.

1.6 Thesis Outline

The first chapter included a consolidation of the issue domain, a rationale for the study's need, an attainment of the research's objective, a depiction of the study's objectives, and a summary of the results. The thesis's remaining components framework are shown below.

The second chapter provides An extensive investigation of the fields of education, machine learning, and decision making process mining. The research primarily focuses on the current methodologies used to forecast pupils' academic performance. In addition, this review will identify and emphasize the limits and shortcomings of the current literature, thereby setting the stage for future research. The subsequent sections of the systematic literature review include the evaluation procedure, elucidation of several data modalities, and the use of data mining and machine learning methods to predict students' programming success.

Chapter 3 provides a thorough description of the basic methodologies used in all tests carried out for this inquiry. The explanations include the many kinds of features, datasets, and the process of feature engineering. Instructional guidelines, assessment methodologies, explainable artificial intelligence (XAI) resources, and descriptions of the different topics. Additionally, a description is provided of the specific hardware and environment that were used. Detailed information on the techniques used in the research may be found in the corresponding chapters.

The results of the research experiments are presented in Chapter 4. This section presents an ablation study of the feature engineering process and an examination of the classification outputs utilizing performance assessment criteria. The explanation of the models is provided in the last part using XAI (Explainable Artificial Intelligence) tools.

Chapter 5 explores the results and consequences of the investigation. The study finishes by presenting an analysis of the results and proposing recommendations for further investigation.

Chapter 2

Literature Review

2.1 Introduction

The most pertinent studies on the subject of EDM is presented in this chapter. Among the most popular applications include recognizing undesirable student behaviors, categorizing students, predicting student progress, and modeling pupils. This chapter highlights the best practices for EDM apps, with a focus on those that have an impact on students. Also discussed are the datasets and techniques used in these studies. A comparison of some of the studies conducted in this field is also given.

2.2 Related Works

EDM is one of the study areas that has been active recently, and in this part, we cover the majority of relevant EDM research projects.

2.2.1 Predicting student performance

The technique of assessing and forecasting student progress is of great interest to EDM experts. Using data mining (DM) methods, the authors of [22] forecast and assess students' academic development based on their participation in online discussion forums and prior academic performance. They have information gathered from pupils in two different college courses. The Naive Bayes (NB), Neural Network (NN), and Decision Tree (DT) classification algorithms for Data Mining (DM) were used to train the dataset. The results of their experiment show that, with an overall prediction accuracy of 86%, the Naive Bayes classifier outperforms the other two classifiers.

Using three distinct machine learning algorithms—Decision Tree (DT), Random Forest (RF), and K-Nearest Neighbors (KNN)—the research [23] improves our understanding of the impact of students' involvement in extracurricular activities on their academic achievement. The Decision Tree (DT) approach achieved an accuracy of 85% and an F1-score of 84%, outperforming all other methods.

For every subject in the dataset, Sudais et al. [24] assessed the students' grades and Cumulative Grade Point Average (CGPA). Finding and predicting the students who were

most likely to fail the semester's examinations was the major goal of this project. Neural networks, decision trees, support vector machines, and the naive Bayes classifier were used to predict the children's outcomes. The most trustworthy approach is to use NB. Overall, the accuracy and dependability of the ML approaches used are inadequate.

The majority of students get education on how to utilize online learning platforms, according to Lonia Masangu et al. [25]. This makes it easier to make sure that students understand what accurate result prediction is. Owing to their methodology, which mandates that each student learn the same amount of material for every course, students who neglect to turn in an assignment or show up for an assigned forum will get a zero on that particular evaluation. To predict academic achievement, the authors analyze student demographic data and online log data, with an emphasis on student interaction, frequency of message checking, number of hands raised in class, number of visited forums, and number of resources available. Decision tree (DT), support vector machine (SVM), random forest (RF), Perceptron classifiers, and logistic regression (LR) are a few of the methods that are used. This study demonstrates the SVM technique's advantage over other approaches for precisely predicting students' academic progress.

Researchers predict students' final academic accomplishment in using video learning analytics and data mining methods [26]. This research forecasted students' final semester grades using video learning analytics and data mining techniques. Eight distinct classification techniques were used to examine the data from the learning management system, mobile applications, and student information system. Their analysis indicates that the RF classifier predicts with an accuracy level of 88.3%.

Reduced training vector-based support vector machine (RTV-SVM) is a unique technique that the researchers suggest be used to identify students who are ostracized or at danger within the academic community. It also removes superfluous training vectors, which cuts down on training time and the need for support vectors. A research that examined the academic performance of 32,593 college students in seven different courses was carried out to evaluate the effectiveness of the suggested RTV-SVM. The results showed that the suggested

Overall accuracy for the method ranged from 92.2% to 93.8% for kids who were at-risk

and from 91.3% to 93.5% for children who were borderline.

The researchers in [28] offer a two-stage model that utilizes DM methods and has the potential to forecast a student's total academic achievement. The model relies on data obtained at the conclusion of a student's first year of their academic journey. Unlike much of the existing research on educational data mining (EDM), academic accomplishment is determined by two factors: the duration required to get a degree and the average grade attained. Moreover, this research suggests a categorization of pupils according to the model's projected levels of achievement and indications of either exceptional performance or lack of success at the beginning of the academic program.

The authors of [29] used machine learning methods to examine the possibility of predicting academic accomplishment using EMIR (Enrollment Management/Institutional Research) data. They discovered that the RF classifier achieved an accuracy rate of almost 90%. This enables the tutor to enhance their instructional approach and provide students with more opportunities for practice outside the classroom. Incorporating other variables into the training dataset may enhance the result [30].

2.2.2 Student's Dropout Predication

One of the main uses of EDM in educational settings is to forecast student attrition. Forecasting dropout rates is essential for educational institutions as it may assist in promptly identifying students who are at risk and implementing proactive actions to ensure their continued enrollment.

The main goals of this research [31] conducted by Janka Kabat ová et al. are to evaluate the efficacy of several ML classifiers, including LR, RF, NB, SVM, and neural networks (NN), and to emphasize the deficiencies of the existing educational data datasets. NB was used to assess access factors, examinations, assignments, evaluations, and projects that were accumulated over a span of four years. It had a peak accuracy level of 93%.

The objective of this work, as stated in [33], is to construct a highly efficient prediction model using simply scholarly data and machine learning techniques like GB, RF, and SVM. However, GB and RF used a dataset collected over a two-year period and were able

to predict student dropout with 94% accuracy. Many factors, including the degree name, success rate, dropout rate, admission exam scores, and median marks, were taken into account while making the forecast. Anjana Pradeep, a researcher at Kerala's M.G. University in India, used several classification algorithms from 2013 to 2018 to predict the rates of student failure and dropout. She used many classification methods, including induction rules and decision trees. Based on the results, the AD Tree that had the most favorable qualities had a judgment accuracy of 92%.

The authors of [35] argue that the behavioral, intellectual, and demographic features of students are the primary factors that form the training dataset for supervised machine learning algorithms. The experiment indicates that the LR classifier is the most efficient technique for accurately predicting a student's course pass status solely based on their final grades.

The research published in [36] used Support Vector Machines (SVM), Decision Trees (DT), Artificial Neural Networks (ANN), and a Naive Bayes Classifier (NBC) to identify students who were at risk of struggling in a future demanding course. The output from NBC is sufficiently accurate at a rate of 85%.

The research described in [37] focuses on the problem of classifying underperforming pupils. The suggested method offers a solution to the absence of past course data, which is often used for training machine learning models.

2.2.3 Predicting students Programming Performance

The goal of assisting CSE students in developing as computer programmers is highly prioritized. To support the teacher Referencing [38], a decision tree-based approach was presented for this task. It divided the students into three groups based on their proficiency in C programming: Good, Average, and Poor. The study's authors used a very small dataset (70) and 16 characteristics. The accuracy of the DT algorithm is 87%.

The Java-based "Introduction to Computer Programming" course provided the dataset for the experiment described in [39]. We looked at the student log data from the Department of Mathematics and Computer Science Unit, which contained details on completed

assignments (ASC), test results (CTS), attendance (CATT), and lab work (CLW). They examined this data using data mining methods such as the ID3 and J48 Decision Tree Algorithms. According to their test findings, J48 is generally 87% correct. Researchers predict university-level students' programming skills and provide a means of enhancing them in a separate study [40]. Using the following criteria, they divided the class into four groups: Excellent, Good, Average, and Weak. Here, the authors provide an actual dataset relevant to this objective, which they then examine to examine six fundamental machine learning methods. According to their test findings, RF may get an accuracy rating of 93%.

2.2.4 Research on AI That Explains

Although there are many similarities between the function and requirements for XAI in education and more general applications XAI is especially needed in education, and the kind of data it uses presents unique difficulties. These include privacy management in the context of AI, justice, and responsibility for accuracy. In particular, learning data is noisy in many ways, and reasoning about it is also noisy [92].

Important findings and debates on the use of eXplainable Artificial Intelligence (XAI) tools like LIME and SHAP to detect characteristics of kids in school dropout situations are discussed in the paper [91].

Giving students more authority and control over their education is one important function of this kind of XAI [93].

Explaining AI, in particular—including the Parents, educators, and students should be able to understand the personal importance of the language and information used [94]. They will be able to employ AI to its full potential and choose whether or not to embrace it thanks to this ability to make independent decisions.

Table 2.1: Related Works on Educational Data Mining

Reference	No. of Sample Used	No. of AttributeUsed	Applied MLClassifier	Best Classifier	Accuracy
[22]	Not stated	38	NB, NN, DT	NB	86%
[23]	500	Not stated	DT, RF, KNN	DT	85%
[24]	900	11	NB, NN, SVM, DT	NB	70%
[25]	480	16	SVM, DT, RF, LR	SVM	71%
[26]	772	Not stated	SVM, DT, RF, LR, ANN, kNN	RF	88%
[29]	1155	118	RF	RF	90%
[30]	5148	20	NBC, GLM, DL, DT, LR, RF and XGboost	RF	86%
[32]	2260	11	ANN, SVM, DT(C4.5), NBC, and *k*-NN	DT (C4.5)	90%
[36]	2973	6	k-NN, ANN, SVM, DT, NBC, and Ensemble	Ensemble	85%
[33]	30217	8	DT(CART), LR, SVM, and NBC	LR	89%
[31]	262	Not stated	SVM, (NN), RF, LR, and NB	NB	93%

[34]	472	13	GB, RF, and SVM	LR	94%
[35]	499	Not stated	NB, NN, SVM, DT, RF, LR	LR	89%
[38]	70	16	DT(ID3)	DT(ID3)	87%
[39]	239	6	DT(ID3), J48	J48	87%
[40]	1720	36	SVM, ANN, RF, DT (ID3, C4.5, CART), KNN, and NBC	RF	93%
Proposed EDM	1720	36	SVM, ANN, RF, DT(ID3, C4.5, CART),KNN, and NBC	Stacking-SRDA	96%

2.3 Summary

There is a categorization issue with our suggested EDM system. This chapter includes a summary of our work on categorization in EDM, and in the end, a comparative study in this area is also offered. Our suggested study on EDM is connected to the research in [14], [15], and [16]. However, certain aspects are still being improved. For example, in [14] and [15], the authors took into account a particular programming language and a tiny dataset. Only essential machine learning techniques were used in [14], [15], and [16]; no ensemble approach was utilized to increase accuracy. Taking into account every shortcoming of earlier studies in this field, we provide an enhanced and comprehensible EDM that surpasses all of the earlier models

Chapter 3

Research Methodology

3.1 Introduction

This study looked at deeper impressions of students' success in programming using DM approaches in an educational setting. Based on the input data, our suggested EDM method essentially forecasts a student's performance status into one of four classes (outstanding, good, average, or weak). An overview of the suggested EDM system is given in this chapter, along with a description of the main procedures and supplies used in each experiment that the research has used. The feature types, datasets, and feature engineering methodology are all covered. Along with the training protocols and assessment techniques, Additionally, it gives a summary of the environment and applications. that were used. In addition, this chapter describes explainable AI tools to help you understand the model.

3.2 Proposed EDM System

With the rapid advancement of technology in today's society, computer science is one of the most in-demand topics for students. The main issue in the area of computer science is programming abilities. Therefore, in order to project themselves as knowledgeable professionals, students must sharpen their programming abilities. We provide our study to better accurately categorize the students' programming based on this inspiration. In order to complete this work, we first properly preprocess the real-world dataset that we have collected from students enrolled in CSE courses at different Bangladeshi institutions, applying feature engineering techniques. After that, we look into the fundamental machine learning algorithm and ensemble method to make predictions about the students' programming skills. We determine which aspects are most crucial for predicting a student class and how much each feature contributes to the models using explainable AI technologies. With computer programming, the teacher will be better able to predict where students would struggle and provide well-informed suggestions based on those students' areas of weakness and potential skill strengthening.

Though it could seem that our suggested EDM paradigm is similar to the ones discussed in [38], [39], and [40], we go into more detail about the distinctions and our unique approach below. In [38], the authors investigated classifying students into three groups based on their performance in a particular college course in programming language C (Good, Average, and Poor). Consequently, they

intended for a small dataset (70 samples) and the minimum amount of attributes (16) connected to instruction in C programming. Data from the "Introduction to computer programming" course's first semester lecture (in Java) was taken by the writers in reference [39]. They only review the log data from the Computer Science Unit of the Mathematics Department, including the Completed Assignment (ASC), Lab Work (CLW), and Class Test Score (CTS), and Class Attendance (CATT) of the students. The dataset contains 36 characteristics and about 1700 data points, to which six important ML models are applied [40There isn't a single, universally applicable technique to evaluate pupils' programming. skills since it necessitates the fusion of technical and analytical knowledge with the creativity and inventiveness of the individual student, according to research [38] [39] [40]. To improve the students' existing programming state, appropriate feature engineering using the ensemble technique and XAI tools is also recommended. As a result, we provide significant elements and characteristics and suggest an enhanced and understandable AI EDM paradigm for assessing the students' overall programming abilities. To do this, we use an appropriate feature engineering approach to our gathered dataset together with appropriate data representation. In order to increase prediction accuracy, we compare and examine important machine learning models as well as a group strategy. Last but not least, to enhance the model's interpretability, we have made use of XAI technologies like SHAPASH, Eli5, and Local Interpretable Model Agnostic Explanations (LIME).

3.3. Approach Overview

The suggested EDM approach's working stages are shown in Figure 3.1. The first and most crucial stage Dataset preparation is part of our suggested EDM paradigm. After the required preprocessing and feature engineering, datasets are collected for various training and use different ML models to evaluate ratios (DT, SVM, ANN, k-NN, GNB,

RF), and a stacking ensemble model is used. Subsequently, the experimental results and evaluation criteria are used to complete the performance evaluation and assessment. Lastly, in order to improve the model's interpretability, we have made use of XAI technologies like SHAPASH, Eli5, and Explanations that are Local Interpretable Model Agnostic.(LIME).Justifications.(LIME).

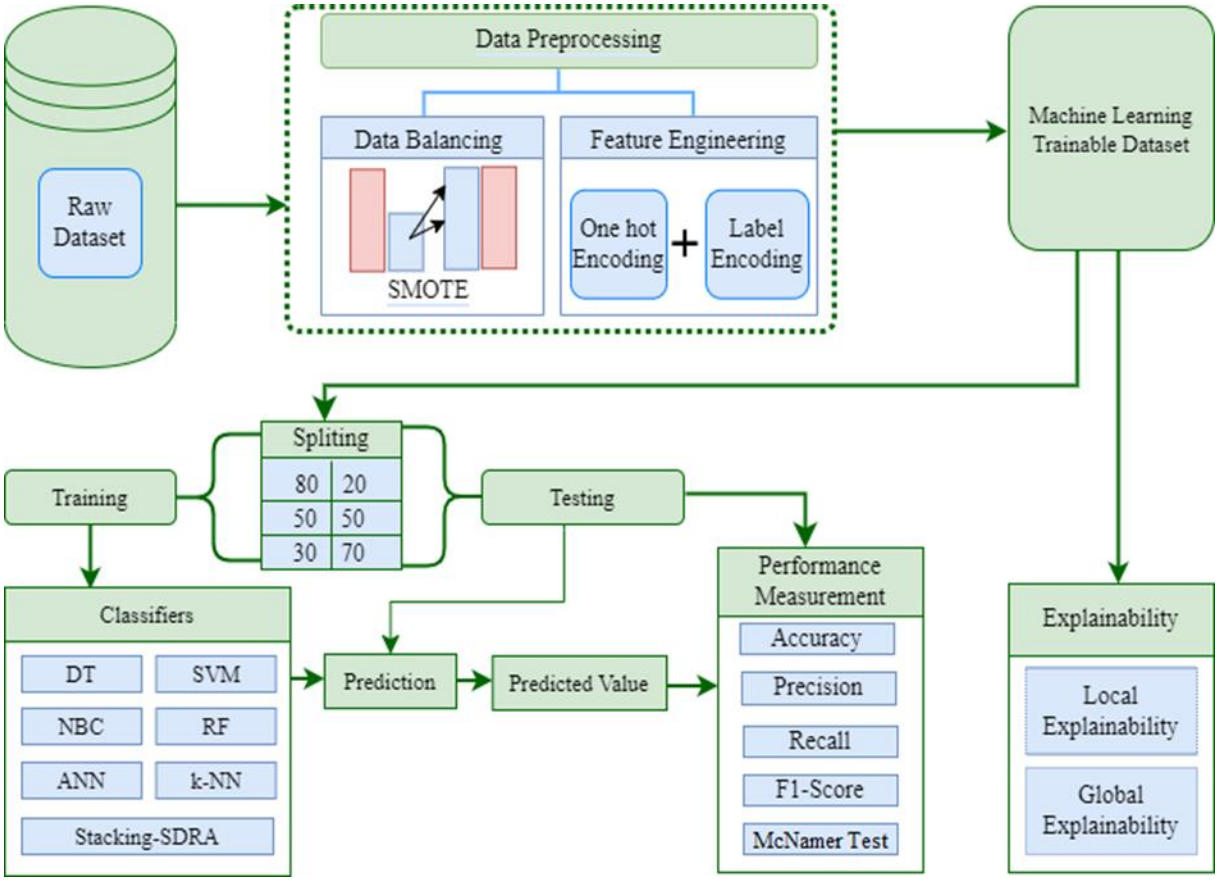


Figure 3.1: The suggested EDM system's work flow.

3.4. Methods And Materials

3.4.1 Feature Engineering and Dataset Preprocessing

Among the 1,720 student participants' records is our gathered dataset [40]. The majority of students are majoring in engineering and computer science. Certain contestants possess the necessary experience from participating in national and international onsite

programming competitions, which equips them to tackle the challenges posed by online contest platforms. There is one target label and 36 characteristics in the dataset. Four classifications make up the target label: Excellent, Good, Average, and Weak. Table 3.1 shows the amount of values for each characteristic, whereas Table 3.2 provides a brief explanation of each feature. Unstructured data for ML tasks may be found in this dataset. Every piece of data is a string value. It is necessary to turn this data into numerical values. In order to accomplish this translation and suitable data representation, we look into a productive feature engineering procedure.

Algorithm 3.1: Proposed Feature Engineering

Input: Raw Dataset (S), Number of Features (n)
Output: Preprocessed dataset after Feature Engineering S_p

Procedure *Feature Engineering (S, n)*
for i=1 to n do
Cμ[i]=Unique(i) // Calculate number of unique values, Cμ for each feature,
end for
for i=1 to n **do**
if (Cμ[i]<7(trial and error manner) **then**
R_b ← binary mapping (OHE) (Cμ[i])
else
R_{le} ← label encoding (Cμ[i])
end if
end for
S_p=Concat (R_b, R_{le})
end Procedure

The suggested feature engineering procedure is represented by Algorithm 1. During the feature engineering process, we convert text and category values to numerical values using One Hot Encoding (OHE) and label encoding approaches. First, for each of the 36 characteristics, we compute the Cμ number of unique values. To prevent the model from overfitting, we conduct several trials to determine the optimal value for each Cμ in order to encode the features as binary mappings. We found that binary mapping the feature works best when Cμ is smaller than 7. One hot encoding is used for binary mapping (OHE). We use label encoding for additional values of Cμ that are more than or equal to 7. During this phase, we use binary mapping for the remaining 28 features and label encoding for 8

features (Feature Nos. 14, 18, 19, 20, 21, 32, 33, and 34 from Table 1). Ultimately, we merge this dual-encoding outcome. This is our last dataset after the feature engineering process.

A prevalent issue with DM systems is data inconsistency. In particular, data imbalance indicates that one class has more samples than the other, which has an impact on the performance of the classifier [41]. To address this kind of imbalance in data, preprocessing methods including undersampling, synthetic minority oversampling, oversampling, and others are often used [42]. To create a fair dataset, the Synthetic Minority Oversampling Technique was combined. (SMOTE) with NearMiss. Table 3.3 displays the outcomes of applying NearMiss and SMOTE on the dataset. This is the final trainable dataset for machine learning.

Table-3.1: Feature name with feature levels

SL	Features	Values (Levels)	No.of Values
1	SSC HSC Result	Excellent, Very Good, Good, Average, Poor	5
2	BSc Result	Excellent, Very Good, Good, Average, Poor	5
3	Problem solved number	Expert, Advance, Intermediate, Learner, Very Beginner, Online Performance	6
4	Onsite Participation	Irregular, Regular, No, Very Regular	4
5	Programming concept before BSc level	Yes, No	2
6	Mathematics on SSC Level	Excellent, Good, Average, Very Good	4
7	"Mathematics" in HSC Level	Excellent, Good, Average, Very Good	4
8	knowledge on "Physics"	Excellent, Good, Average, Very Good	4

9	problem understanding capability	Good, Average, Excellent, Better, Very Good	5
10	Real-Life Problem-Solving Skills	Yes, Of Course Yes, Maybe No	4
11	Patience in Problem solving	Very Good, Good, Average, Low	4
12	Learning Speed	Very Fast, Fast, Average, Slow, Very Slow	5
13	Programming Language Diversity	Very High, High, Average, Low	4
14	Programming Learning period	Before University Admission, Level 1, Level 2, Level 3, Level 4, Undergrad Passed	22
15	Programming Experience	Very High, High, Average, Low	4
16	Learning Method	Of Course Yes, Yes, No, Maybe	4
17	Technical Experience	Excellent, Very Good, Learning, Average, Low	4
18	Computer Architecture Knowledge	Excellent, Very Good, Good, Learning, Average, Low, Yes, No	8
19	Knowledge on Algorithm	Excellent, Very Good, Good, Learning, Average, Low, Don’t Know, None	8
20	Knowledge on Data Structure	Excellent, Very Good, Good, Learning, Average, Low, Don't know, Zero, None	9
21	Use of STL	Yes, No, Maybe, Sometimes, something, Use Python Library, Don’t know what is this?, Never used, sometime, I am a beginner, Yes i do.And in python the default library is very useful, Rare, I did use but not now	14
22	Searching ability	Excellent, Very Good, Good, Average, Low, Very Low	6
23	Mentor	Yes, No	2

24	Ask help	Yes, Sometime, No	4
25	Improvement policy	Of Course Yes, Yes, No, Maybe	4
26	Learning from failure	Of Course Yes, Yes, No, Maybe	4
27	Coding Curiosity	Very Much Curious, Curious, Not Interested	3
28	Mentoring to others	Of Course Yes, Yes, No, Maybe	4
29	Communication Skills	Excellent, Very Good, Good, Average, Low, Very Low	6
30	passion on learn code	Excellent, Very Good, Good, Average, Low, Very Low	6
31	General Knowledge	Excellent, Very Good, Good, Average, Low, Very Low	6
32	Creativity Level	Excellent, Very Good, Good, Average, Low, Very Low, I cannot compare, No creativity	8
33	Coding Understand ability	Excellent, Very Good, Good, Average, Low, Very Low, No Knowledge	7
34	Debugging skills	Excellent, Very Good, Good, Average, Low, Very Low, No skills	7
35	Weekly Practice	Very Regular, Regular, Irregular, Weakly Regular	4
36	Reading habit	Some Time, Regularly, When Needed, No	4
37	Target	Excellent, Good, Average, Weak	4

Table-3.2: Short description of the features.

SL	Features	Short description
1	SSC HSC Result	Result in GPA at Secondary School Certificate (SSC) and Higher Secondary Certificate (HSC)
2	BSc Result	Result in CGPA at B.Sc.
3	Problem solved number	No. of programming problems solved in different online platform and from the text book
4	Onsite Participation	No. of onsite programming contest (NCPC/ICPC/ IUPC) participations
5	Programming concept	Programming coding knowledge before B.Sc. level
6	Mathematics on SSC Level	Result of Mathematics in grade point (GP) at SSC
7	"Mathematics" in HSC Level	Result of Mathematics in GP at HSC
8	knowledge on “physics"	Result of Physics in GP at SSC
9	Problem understanding capability	Capability of problem solving and understanding the problem
10	Real-Life Problem-Solving Skills	Capability to solve the real-life problems
11	Patience in Problem solving	Patience level with solving a difficult problem
12	Learning Speed	Learning speed of the participant
13	Programming Language Diversity	No. of programming languages that the participant knows
14	Programming Learning period	Time period when the participant learns most programming languages
15	Programming Experience	Experience of programming coding of the participant programmer in years
16	Learning Method	Learn by hands-on coding
17	Technical Experience	Technical experience of the participant besides programming

18	Computer Architecture Knowledge	Basic knowledge of hardware configuration and computer architecture
19	Knowledge on Algorithm	Experience of using different algorithms to solve the problem
20	Knowledge on Data Structure	Knowledge level on data structure
21	Use of STL	Experience of using Standard Template Library (STL)
22	Searching ability	Ability to search on google or other search engines
23	Mentor	Participant has any mentor or not
24	Ask help	Participant asks help or not when facing difficulties
25	Improvement policy	Methods of tracking records for further improvement
26	Learning from failure	Participant has the practice to learn from failure or not
27	Coding Curiosity	Coding curiosity level
28	Mentoring to others	Participant teaches other programmers or not
29	Communication Skills	English language proficiency level
30	Passion on learn code	Patience level when solving a difficult problem
31	General Knowledge	Knowledge of scientific methods, laws, and theories
32	Creativity Level	Creativity level of the participant
34	Debugging skills	Ability to compiling the code and solve the error
35	Weekly Practice	Weekly Practicing routine to improve coding skills
36	Reading habit	Habit to read programming blogs or books
37	Target	Students' category (Excellent, Good, Average, Weak)

Table-3.3 State of the dataset before and after data balancing using SMOTE and NearMiss.

	Excellent	Good	Average	Weak
Original Dataset	564	307	346	503
After usingSMOTE	564	564	564	564
After usingNearMiss	307	307	307	307

3.4.1.1 One Hot Encoding (OHE)

Textual information is inaccessible to ML algorithms. Numbers are required for data. This research's data preprocessing involved encoding textual information as one-hot vectors. OHE increases the model's comprehension of the data [43]. In this encoding approach, a new variable is established for each unique level of a categorical property or feature, and each class or category is mapped to a binary variable that may have two possible values: 0 or 1. As a result, the numbers 0 and 1 denote the existence and absence of that class or category, respectively. [44]. With OHE, feature vectors will have a large number of dimensions if their cardinality is high. However, one-hot encoding is widely used because of its ease of use [45, 46]. OHE works well on models that have high smoothing properties, and it is most often used on tweets or sentences that not have many repeated elements. A one-hot vector is a 1 N matrix (vector) in which every cell but one is set to 0 to indicate that it is not part of a word. OHE enables a more evocative portrayal of categorical data. Figure 3.2 represent the result of applying OHE to a single feature3.2.

ID	Problem_solved _number
1	Expert
2	Advanced
3	Expert
4	Beginner
5	Advanced

ID	Problem_solved _Expert	Problem_solved _Advanced	Problem_solved _Beginner
1	1	0	0
2	0	1	0
3	1	0	0
4	0	0	1
5	0	1	0

Figure 3.2: Example of applying OHE to a single feature

3.4.1.2 Label Encoding

Label encoding is a technique used in data analysis and machine learning to convert category variables into numerical labels. This method assigns a distinct integer value to every category [44]. For instance, the target label in our dataset is divided into four groups: poor, mediocre, excellent, and great. In order to label encode this variable, you would give each potential value—3,2,1,0 for great, good, average, and weak—a unique integer value. Label encoding is a helpful method in situations when the category variable follows a natural sequence, such "low," "medium," and "high." Assigning integer numbers according to the natural order might be useful in certain.

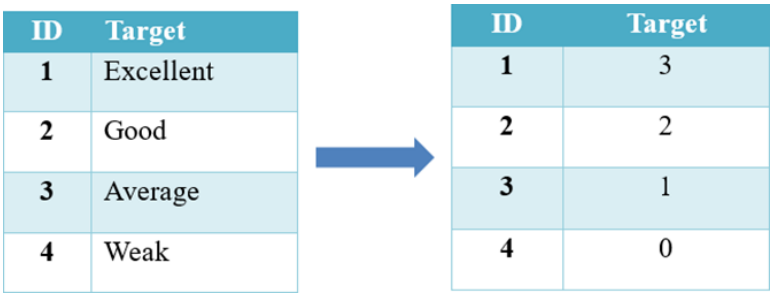


Figure -3.3 Example of applying Label Encoding to a single feature

3.4.1.3 Synthetic Minority Oversampling Technique (SMOTE)

In many application fields, SMOTE is the most popular and efficient oversampling technique for addressing imbalanced data [47]. One class may have substantially fewer instances than the other classes in a class-imbalanced dataset, which might result in a model that is skewed to benefit the dominant class. SMOTE raises the quantity of occurrences in the minority class by creating artificial samples that resemble the already available minority samples. The SMOTE algorithm creates new samples by interpolating between the minority class samples that are already available. SMOTE creates new samples for each minority sample by interpolating between the selected sample and its neighbors. by choosing the k closest neighbors from the minority class [48]. A sampling rate parameter that determines how many fresh samples must be produced controls the degree of interpolation. The number of minority class instances is then increased by adding the additional samples to the dataset. Data balance is shown using SMOTE on

the dataset in Figure 3.4.

SMOTE can efficiently balance the dataset and raise the proportion of occurrences of minority classes, enhancing the model's performance. Oversampling, however, may also lead to overfitting, particularly if the artificial samples produced are very similar to the minority class samples already in existence. Consequently, it's critical to thoroughly assess the model's output after the application of the SMOTE approach.in Figure3.4

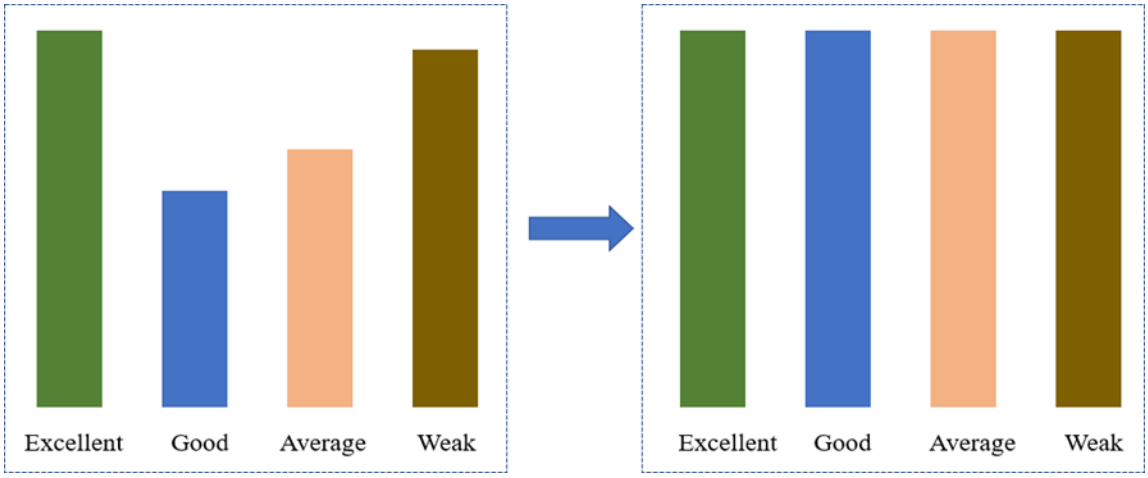


Figure 3.4: Data balancing representation applying SMOTE to the dataset

3.4.1.4 NearMiss

Machine learning uses the popular undersampling technique known as NearMiss in order to deal with the issue of class disparity [47]. One class in an unbalanced dataset may contain much less instances than the other classes, which might result in a biased model that benefits the majority class. NearMiss minimizes the number of examples in the majority class in order to balance the dataset and enhance model performance.

The NearMiss method is used to choose samples from the majority class that most closely resemble examples from the minority class. This technique preserves the minority class's decision border by removing just the cases that are near the majority class. Following the selection of the majority class samples, the final dataset will have a balanced class distribution. However, undersampling may also result in a loss of information and accuracy, especially if large samples are included in the majority class. Therefore, it is

essential to analyze the model's performance in-depth after the use of the NearMiss method.

3.4.2 Model Training

The ML algorithms are trained after the dataset has been created to determine which classifier is better at predicting students' programming prowess. Many experiments are carried out in an effort to reduce the discrepancy between the results as they were and as planned. In particular, we concentrate on k-NN, DT, ANN, NBC, SVM, and RF ML models. To find the optimal values for C and gamma while training the SVM model with one of four different kernel functions—polynomial, Gaussian, linear sigmoid, and Radial Basis Function (RBF)—we use a grid search technique as required. We construct an ANN classifier with five hidden layers on top of the Multi-Layer Perceptron (MLP) architecture. Next, we experimented with various hyperparameter settings to evaluate the RF algorithm's performance. such as alpha, random state, max-depth, etc. We use two commonly used construction strategies, ID3 and CART, to produce the DTs. Additionally, we train k-NN and NBC, two additional popular ML algorithms, for DM training. The Gaussian distribution is used by NBC to predict student performance. Lastly, we use an ensemble learning approach called stacking to improve the prediction accuracy. An ANN serves as the base model in our proposed stacking approach, Stacking-SDRA, while SVM, DT, and RF serve as the submodels.

3.4.2.1 DecisionTree(DT)

A well-liked supervised One machine learning method that may be used to issues with both regression and classification is the decision tree.. classifier [49]. By recursively dividing the feature space into smaller areas depending on the feature values, the DT algorithm predicts the class of a sample until each zone only includes samples of a single class [50]. In the tree structure, Each branch leads to the test's outcome and a child node, and each internal node represents a test on a feature. that represents a subset of the decision tree classifier's samples. For every subset, the anticipated class is represented by the leaves of the tree [51].

The following is a mathematical representation of the decision tree classifier:

Assume that X = (x_1, x_2,..., x_n) is the feature vector of size n, and that y is the class label. A function f() that translates from class labels y to feature vectors X may be used to describe a decision tree classifier.

If X ∈ R_k~, then f(X) = C_k~(3.2)

where the class label given to the area R_k is C_k. Recursive partitioning of the feature space according to feature values defines R_k, a subset of the feature space.

A sequence of binary tests on the features define how the feature space is partitioned. A binary test is run on a feature x_j at each internal tree node to determine if the characteristic value falls within a certain range or meets a particular condition.

The binary function h_m(x_j), which returns 1 if the feature x_j meets the requirement and 0 otherwise, defines the test at node m. The following is a representation of the binary function:

If x_j < t_m, then h_m(x_j) = 1, and 0 otherwise,

A The decision tree classifier may be trained using dataset D = {(X_1, y_1), (X_2, y_2),..., (X_n, y_n)}. The best feature for dividing the data at every node is chosen in a recursive manner to build the tree. A splitting criteria that assesses the quality of the split is used to determine which feature is the best. Information gain, entropy, and Gini impurity are examples of common splitting criteria.

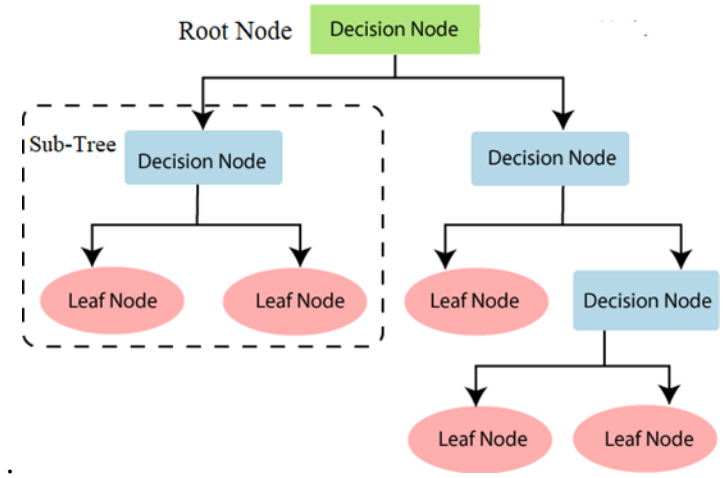


Figure 3.5 Top-down representation of DT classifier.

The top-down form of the DT classifier is shown in Figure 3.5. After the tree is constructed, a new sample's class label may be predicted by going through the tree from the root node to a leaf node, making predictions about the class label at each stage by selecting which child node to pursue based on the binary tests at each node. The projected class label is shown after a leaf node is reached. ID3: The ID3 method often relies on classes that have no missing values in their nominal properties. To construct the DT from the training data, ID3 uses information gain [52]. In order to explore the tree of possibilities, this technique takes a top-down, greedy search strategy, ignoring branches that have already been studied. A hierarchy is established using the data that has been acquired. The exam information labeling process uses this information that is extracted from the training examples [53]

Algorithm 3.2 ID3.

Input: Training dataset

Output: DT

1: procedure *ID3*():

2: Compute entropy of dataset, $E(\mathbf{S})$

3: **for** each feature **do**

4: Find $I(\boldsymbol{A})$

5: Calculate $IG(\boldsymbol{A}) = E(\mathbf{S}) - I(\boldsymbol{A})$

6: end for

7: Assign highest $IG(\boldsymbol{A})$ attribute as root node Prepare subdataset for each value of root

8: node

9: Repeat steps 2-8 until getting the desired tree

10: end procedure

CART: Besides the benefits of C4.5: CART is able to manage multiple-output problems, non-linear connections, and missing data [54, 55]. It finds dataset patterns automatically via self-validation, hence avoiding the overfitting issue [56]. The CART trees are constructed using Hunt's approach, and the splitting criteria are determined by means of the Gini index. [57].

3.4.2.2 Support Vector Machine (SVM)

For The Support One popular machine learning technique for binary and multi-class classification issues is the Vector Machine (SVM) classifier. The SVM classifier divides the data into discrete groups by locating a hyperplane in a high-dimensional feature space, as shown in Fig. 6 [58][59]. The following is a mathematical representation of the SVM classifier.

The training data is split into two classes by a hyperplane generated by the SVM classifier. +1 and -1, given a set of training data D = {(X_1, y_1), (X_2, y_2),..., (X_n, y_n)}, where y_i is the corresponding class label and X_i is the feature vector for the i-th sample. The hyperplane is defined by the equation w^T x + b = 0 (3.3).

where w is the weight vector, b is the bias term, and x is the feature vector.

By calculating the distance between the hyperplane and the closest points, the SVM classifier decides which hyperplane widens the gap between the two classes. from each class. An SVM classifier may be shown as a hyperplane or as a line separating the two classes in a two-dimensional feature space. Support vectors are the points from each class that are closest to the hyperplane and are used to calculate the margin. The position of a new sample on the hyperplane, which acts as the SVM classifier's decision boundary, determines its projected class.

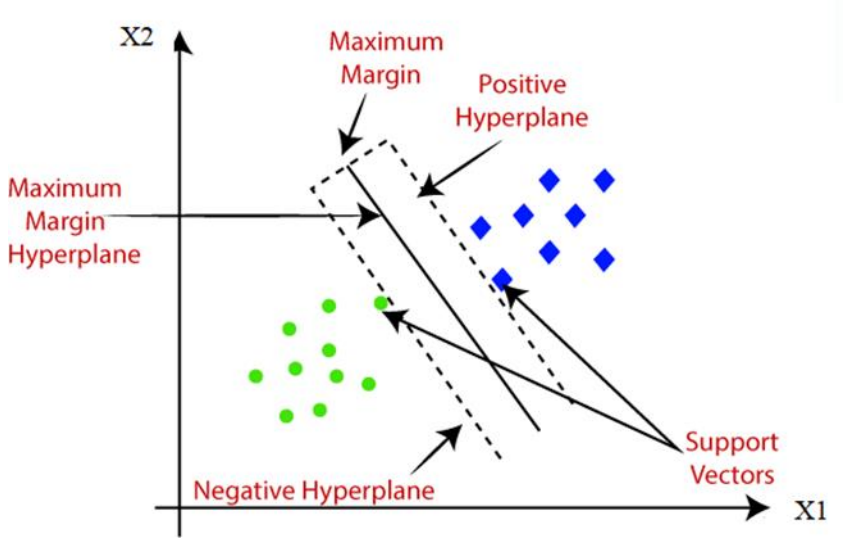


Figure 3.6: SVM classifier with decision boundary and hyperplane

The SVM approach is more sophisticated since it requires a high dimensional dot product for both training and classification. To lessen this expensive computation, a Kernel (), when the kernel function, Kernel (), is used [60]. The kernel function affects the properties of SVM, such as linearity and non-linearity. We use four distinct kernels in this work: polynomial, sigmoid, linear, and RBF.

3.4.2.3 Naïve Bayes Classifier (NBC)

The foundation of the Naive Bayes model is the Bayes theorem. A probabilistic classifier classifier [61]. This paradigm states that an attribute's existence in a class does not entail the existence of any other attribute. This assumption simplifies probability calculations and allows for effective feature-rich data categorization.

The following is a mathematical representation of the Naive Bayes Classifier:

After receiving a set of training data The Naive Bayes Classifier computes the posterior probability of each class given a new sample x, where D is a set of tuples (X_i, y_i), representing the feature vector X_i and its associated class label y_i for the i-th sample.

The number 3.4 is represented as (3.4). The expression P(y) P(x | y) / P(x) may be simplified to P(y).

The equation represents the relationship between the posterior probability of class y given sample x (P(y)), the marginal probability of sample x (P(x)), the prior probability of class y (P(y)), and the likelihood of sample x given class y (P(x|y)).

The Naive Bayes Classifier assumes that the probability of each feature, given its class, is independent of the other features. This is the method by which it can be computed.

The expression for the conditional probability P(x | y) is obtained by multiplying the individual probabilities P(x_1 | y) and P(x_2 | y). The expression * P (x_n | y) *... * (3.5) represents the probability of feature i given class y, where P (x_i | y) is the probability of feature i given class y. One may estimate the prior probability of each class by utilizing the training data, then estimate the likelihood of each feature given the class using methods such as maximum likelihood estimation and Bayesian estimation. However, the Naive Bayes Classifier presupposes that eachOne may estimate the prior probability of each class by utilizing the training data, then estimate the likelihood of each feature given the class

using methods such as maximum likelihood estimation and Bayesian estimation..

However, the Naive Bayes Classifier assumes that every attribute is independent of the others, which may not always be the case. Furthermore, if a feature is missing from a certain class, the Naive Bayes Classifier can face 0 probability. This issue could be resolved by using methods like Laplace smoothing or Bayesian smoothing.

3.4.2.4 Artificial Neural Network (ANN)

Artificial Neural Network (ANN) classifiers are machine learning approaches that are designed to mimic the structure and functioning of the human brain. The number 64. The objective is to use the acquired knowledge from data analysis to predict future events. The ANN classifier is composed of three layers: an input layer, an output layer, and one or more hidden layers. Once the input layer collects the feature vectors from the input data, the hidden layer analyzes the data. Finally, the output layer generates the desired class label [65].

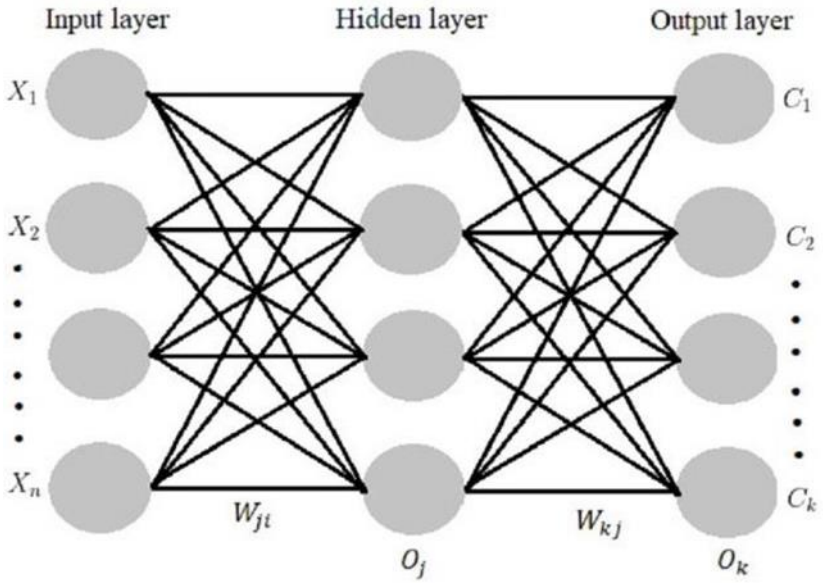


Figure 3.7: Basic Neural Network Architecture including hidden layer

Layers of neurons make up the artificial neural network (ANN) classifier. Each layer adds the weighted sum of its inputs and generates an output via the application of an activation function. During training, To lessen the discrepancy between the predicted

and actual outputs, the weights and biases of the neurons are adjusted. The following is the mathematical expression for the ANN classifier's activation function: The weight matrices for the input-to-hidden and hidden-to-output layers are denoted by w_1 and w_2. The ANN classifier can be mathematically represented as follows:

y_hat = f(w_2 * f(w_1 * x + b_1) + b_2) (3.6)

The ANN classifier modifies the weights and biases of the network's neurons during training by using the backpropagation approach [67]. The gradient descent method is used in backpropagation to update the neurons' biases and weights in reaction to the mistake that derives from the difference between the expected and actual outcomes.

3.4.2.5 k--Nearest Neighbor (KNN)

Based on instance-based machine learning, KNN is a non-parametric supervised learning algorithm [68, 69]. KNN has a quicker training phase than competing classifiers, but its testing phase uses more resources and is slower. The k-value

determines the categorization in k-NN. As shown in Fig. 6. Using a majority vote among its closest neighbors, k-NN ascertains the class of a sample data point; the number of neighbors to be assessed is indicated by the k-value. The sample data point's distance is used to calculate the neighbors [70]. However, since it utilizes memory, rapid

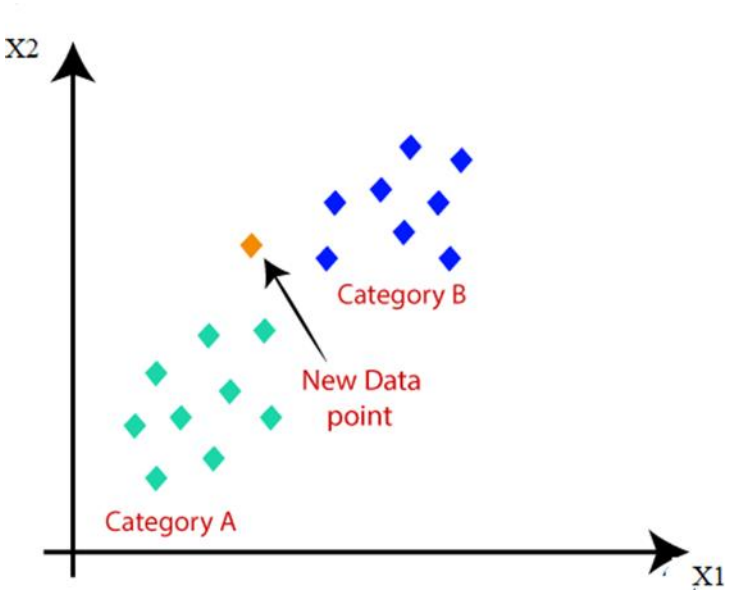


Fig 3.8: Representation of a new data point classing in KNN

3.4.2.6 Random Forest (RF)

One ensemble machine learning technique [72] that increases prediction accuracy is RF. of a dataset by means of decision trees. It averages the results by using many decision trees on several subsets of the primary data. By use of classification and regression techniques, this program can forecast continuous as well as categorical data. By means of a random subset of the provided dataset as ML trainable data, each decision tree selects at random the major relevant characteristic of data samples using a simple deterministic probability [73]. To raise the prediction accuracy of testing data, the data is therefore divided into essentially similar groups for every tree—that is, nodes. The value of the predictor vector helps one to find these splitting nodes. Another name for the vectors separating the datasets is crucial vectors. Many pre-defined bootstrapped datasets on several decision trees are fit using the RF classifier. Every decision tree class answers categorically based on the expected value. The mean of the fitted response of each independent tree in the bootstrapped sample is the anticipated outcome for an unbroken response from every independent tree. Stated differently, RF forecasts the result based on the majority vote or the average of all the predictions generated by each decision tree, hence transcending the dependence on one decision-making tool. {7} [RF is a technique of ensemble machine learning [72] that increases the prediction accuracy of a dataset by means of decision trees. It averages the results by using many decision trees on several subsets of the primary data. By use of classification and regression techniques, this program can forecast continuous as well as categorical data. By means of a random subset of the provided dataset as ML trainable data, each decision tree selects at random the major relevant characteristic of data samples using a simple deterministic probability [73]. To raise the prediction accuracy of testing data, the data is therefore divided into essentially similar groups for every tree—that is, nodes. The value of the predictor vector helps one to find these splitting nodes. Another name for the vectors separating the datasets is crucial vectors. Many pre-defined bootstrapped datasets on several decision trees are fit using the RF classifier. Every decision tree class answers categorically based on the expected value. The mean of the fitted response of each independent tree in the bootstrapped sample is the anticipated outcome for an unbroken response from every

independent tree. Stated differently, RF forecasts the result based on the majority vote or the average of all the predictions generated by each decision tree, hence transcending the dependence on the one decision tree. [74].

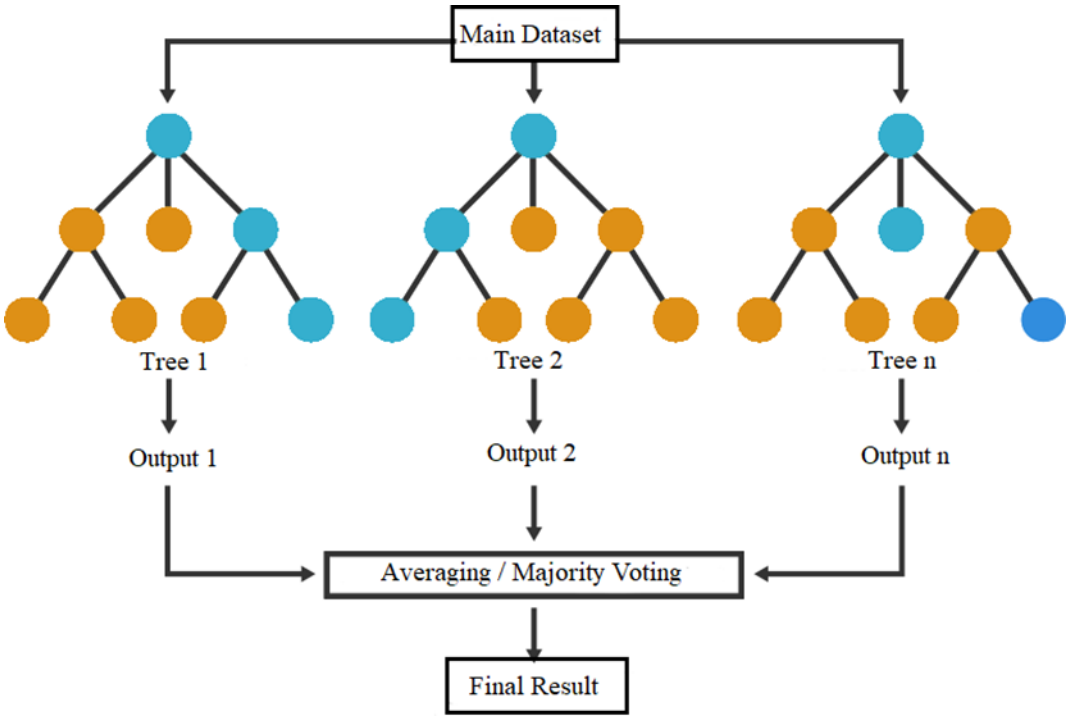


Figure 3.9. Working process of RF classifier in the top-down approach

3.4.2.7 Ensemble Learning-Stacking

By merging the results of many models, a common machine learning approach called ensemble stacking may increase prediction accuracy [75].

Using various techniques and/or hyperparameters, many base models are trained on the same training set of data in an ensemble stacking technique. Following that, the outputs of various models are integrated, often by training a meta-model atop the underlying models' outputs [76].

Using the basis models' outputs as inputs, the meta-model generates a final prediction. Since the meta-model may learn to combine the several base models' strengths and reduce their shortcomings, this technique can often provide greater performance than any of the base models alone.

Ensemble stacking may be implemented in a variety of ways. Using One often used approach is k-fold cross-valuation to split the training data into k subgroups. The

fundamental models then go through k-1 subsets of training, and the meta-model gets its inputs from their predictions on the other subset. Once through the k rounds of this process, every subset is used as the validation set once. The final predictions are then derived by averaging the findings of the meta-model across k iterations. We have shown for this study the Stacking-SRDA model, which bases ANN on SVM, RF, and DT.

3.4.3 Performance Evaluation

We utilize the four most used performance assessment criteria to evaluate the effectiveness of the ML classifier in predicting students' programming progress, including accuracy, precision, recall, and F1-score [77]. In an ML model, the testing accuracy is defined as the percentage of the dataset's actual value that corresponds with the projected value [78]. This helps with student categorization. Precise calculations are used to determine the positive predictive value, or chance of a positive test result [79]. Based on their programming performance, we put students into four categories: excellent, good, average, and poor. Precision shows the percentage of students who are accurately placed into each category. Recall indicates the likelihood of true positives (TP) from the machine learning model's total projected positive values [80]. F1-score methods for acting quickly in relation to related methods. The accuracy and recall components add up to the F1-score's total value. In the case of an imbalanced dataset, precision, F1-score, and recall are sometimes seen to be more helpful performance metrics than accuracy [81]. The ML model generates True Positive (TP), True Negative (TN), False Positive (FP), and False Negative (FN) results while providing guidance. TP, TN, FP, and FN data are used in the following ways to create ML performance monitoring tools [82]:

ROC Curve: Another performance metric for machine learning algorithms is the Receiver Operating Characteristic (ROC) curve. The connection between the true-positive rate (sensitivity) and the false-positive rate (specificity) is shown by the ROC curve. [83]test of significance (McNemar test): We use the McNemar Test to evaluate the Stacking-SRDA significant level with that of another ML classifier. A statistical test called the McNemar test compares the proportions of two paired binary samples to ascertain whether or not they are representative of the same population [84].

When there are only two potential outcomes for a given observation or variable, like in the case of a medical diagnosis or the assessment of a machine learning algorithm's performance, the McNemar test is often used [85]. It is often used to determine if two models or algorithms' classification performance on a certain dataset differs noticeably .The way the McNemar test works is by comparing the percentage of instances in which two models predict the same thing and the percentage in which they predict something differentThe test statistic is obtained by dividing the sum of the proportions by the square of the difference between the two proportions. According to the null hypothesis, which holds that there is no difference between the two models, the test statistic is described by a chi-squared distribution with one degree of freedom. If the estimated test statistic is greater than the critical value of the chi-squared distribution, we reject the null hypothesis and conclude that there is a significant performance difference between the two models.

3.4.4 Explainable AI

In order to improve model transparency, Explainable Artificial Intelligence (XAI) has emerged as a means of converting AI from a black-box model to a gray-box one. XAI seeks to provide a range of techniques that produce highly performant and more understandable models [86]. Educational institutions may find this approach quite useful when making decisions. A variety of techniques, including decision trees, rule-based systems, and neural network visualization techniques, may be used to develop XAI [87]. These approaches aim to give a range of explanations for the actions of AI models, from local explanations that offer detailed knowledge of particular decisions or predictions made by the model to global explanations that provide a comprehensive comprehension of the model's behavior [88]. XAI has become a more active area of research as a result of an increase in publications and conferences on the topic in recent years. It is interesting to note that the literature contains XAI explainers customized for certain models, in addition to global and local explanations.

3.4.4.1 Global Explainability

Understanding the model's overall behavior and average performance across various input types is made easier with the help of global explanations. They are also crucial for detecting any biases or mistakes in the model that may need to be fixed or enhanced. We use ELI5 and SHAPASH for global explainability in this work.

SHAPASH: An open-source Python library called Shapash XAI offers a number of resources and methods for creating explainable artificial intelligence (XAI) models. Data scientists and developers may rapidly and simply provide explanations for their machine learning models using Shapash because of its very user-friendly architecture [89]. Shapash's capability to provide machine learning models both local and global explanations is one of its primary strengths. Local explanations provide specific insights into each prediction or choice the model makes, while global explanations give a broad grasp of how a model functions and which variables influence its outputs most.

In addition, Shapash offers a variety of interactive tools and visualizations for investigating and deciphering machine learning models. Among them are bar charts, scatter plots, and interactive dashboards that may all be tailored to a certain application's or use case's requirements. Shapash's great degree of modularity and flexibility is one of its main advantages. This implies that developers may tailor it to their own requirements and include it with ease into their current processes. Furthermore, a multitude of machine learning frameworks frameworks and libraries like TensorFlow, XGBoost, and scikit-learn, are compatible with Shapash.

ELI5ELI5 is a subfield of explainable artificial intelligence (XAI). strategy designed to provide comprehensible and unambiguous insights into the major variables influencing a model's predictions [90]. It is a helpful tool for a variety of applications as it is made to utilize language that is simple enough for non-experts to understand. A range of tools and methods are available in the eli5 package to help implement ELI5 in Python.

3.4.4.2 Local Explainability

Local explanations may be useful in understanding how the model arrived at a certain prediction or judgment as well as in pointing out situations in which the model could have erred or made errors. For local explainability, in this study, we use LIME.

Theories that are LIME (Local Interpretable Model Agnostic): In order to offer clear and intelligible explanations for the model's predictions, another popular XAI technique called LIME involves creating a local approximation of the model [90]. Like ELI5, LIME is implemented using the lime package, a customized package that provides a wide range of tools and functions for the creation and analysis of LIME explanationsThe capability of LIME is one of its main advantages. to provide personalized local explanations for each prediction. In scenarios where understanding the reasoning behind each decision a machine learning model makes is crucial, such as credit scoring or medical diagnosis, this may be quite useful.

3.4.5 Software

Python programming, the scikit-learn module, and the Jupyter notebook were used for all phases, including data pre-processing, categorization, and explainability. Scikit-learn is a Python machine learning toolbox that has seen significant growth in popularity lately. The IPython Notebook is no longer recognized by its previous moniker, Jupyter Notebook. It's an online interactive computational environment program that is available as open-source software. Live code, execution, math, graphs, equations, visualizations, and rich media may all be integrated in this setting. This program was used for ML, statistical modeling, data transformation, and cleaning, among other things.

3.5 Summary

The main goals of this study are to characterize the strategy and predict the students' success in programming. Giving an explanation of the steps taken in the process of doing this study was the aim of this chapter. In addition to providing details on the data sources themselves, this chapter covers the three datasets that came from various sources of information. It also outlines the main methods that were used to each experiment carried out for this investigation. Additionally, it explains the environment, the hardware and software that were used, the training protocols, and the assessment techniques.

Chapter 4

Result and Discussion

4.1 Introduction

The subject of this chapter will be the experimental results of our proposed EDM system. Several machine learning algorithms are used to categorize students based on their different programming abilities, including SVM, NBC, DT, ANN, RF, KNN, and Stacking-SRDA. These techniques are built using Python's scikit-learn package, which provides a flexible and reliable framework for developing and evaluating ML models. A range of techniques are used to assess the models' efficacy, including as performance assessment parameters like recall, accuracy, accuracy, and F1-score, together with the quality of fit criteria like the ROC curve. The dataset that contains 36 different variables that may be utilized to predict a student's programming skill is divided into three various dataset split ratios for training the machine learning algorithms. These attributes are acquired by classifying students into four groups (weak, average, excellent, and outstanding) using supervised approaches. Ultimately, our methodology leverages many robust machine learning algorithms and assessment metrics to provide a comprehensive and effective means of assessing and categorizing students based on their programming skills.

4.2 ML Classification Result without Proposed Preprocessing Pipeline

We used an 80:20 ratio to divide the dataset and classify student programming proficiency using machine learning techniques. The performance of the ML classifier without the suggested pretreatment approach (data balancing and feature engineering) is shown in Table 4.1. Among all the ML algorithms, RF performs the best in this situation. Impressive results include 91% f1-score, 92% recall, 92% precision, and 92% accuracy. The other algorithms continue to function as expected, with NBC demonstrating at least 79% accuracy and f1-score. A bar chart illustrating the performance of the other algorithms can be seen in Figure 4.1.

Table 4.1: ML classification result without proposed processing pipeline

Model	Accuracy	Precision	Recall	F1-score
SVM	0.82	0.82	0.82	0.82
DT	0.80	0.80	0.80	0.80
NBC	0.79	0.80	0.79	0.79
ANN	0.82	0.82	0.82	0.82
KNN	0.80	0.81	0.79	0.79
RF	0.92	0.92	0.92	0.91

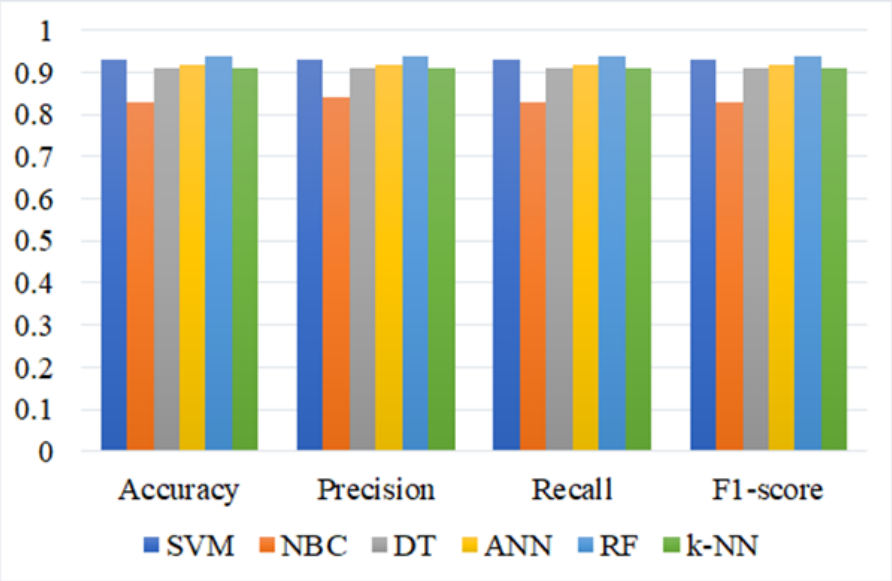


Figure 4.1: Performance of the ML classifier without proposed processing pipeline

In comparison to the present outcome, previous work [40] has already achieved superior accuracy in predicting students' programming ability. We use SMOTE and NearMiss as data balancing approaches with individual feature engineering to enhance prediction performance. The classifier's performance is shown in the following result after the use of feature engineering and data balancing strategies.

4.3 ML Classifier Performance after Applying SMOTE and Feature Engineering

To improve the performance of the classifiers, we apply feature engineering to the dataset and employ SMOTE as a data balancing technique. The results of the experiment are shown

in Table 4.2. All of the classifiers perform better overall than the outcomes shown in table 4.1. In this case, RF obtains 94% recall, accuracy, precision, and f1-score. Over 90% accuracy is shown by SVM, ANN, and DT; over 85% accuracy is achieved by NBC and KNN. The performance of the classifier is shown as a bar chart in Figure 4.2.

Table 4.2: Results of classification utilizing machine learning algorithms with an 80:20 training to testing ratio using SMOTE and feature engineering

Model	Accuracy	Precision	Recall	F1-score
SVM	0.93	0.93	0.93	0.93
DT	0.91	0.91	0.91	0.91
NBC	0.85	0.86	0.85	0.85
ANN	0.93	0.93	0.93	0.93
KNN	0.87	0.87	0.87	0.87
RF	0.94	0.94	0.94	0.94

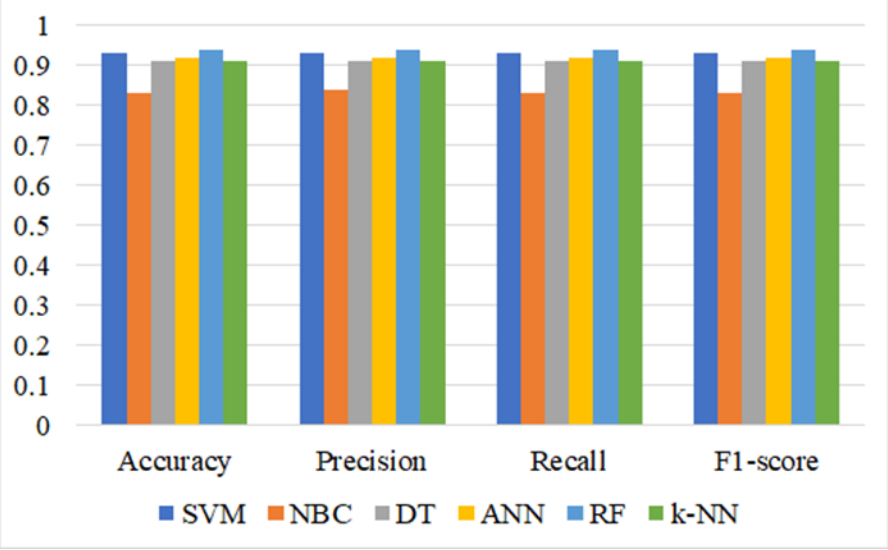


Figure 4.2: Performance of the ML classifiers with after applying SMOTE and feature engineering, training and testing ratio is 80:20.

The performance of the ML classifiers in an 80: The table displays the training to testing ratio of 20. 4.2. We test the classifiers' tolerance by doing the same analysis in 50:50 and 30:70 ratios.

Table 4.3 displays the effectiveness of algorithms for machine learning with a 50:50 training to testing ratio. This experiment's result shows that RF achieves a 91% f1-score as well as 92% accuracy, precision, and recall. When RF, SVM, and ANN outperform other classifiers, SVM and ANN attain an accuracy of more than 90%.

Figure 4.3 displays the classifier's performance as a bar chart.Table 4.3: Classification results using ML algorithms after applying SMOTE and feature engineering, training and testing ratio is 50:50

Model	Accuracy	Precision	Recall	F1-score
SVM	0.91	0.91	0.91	0.91
DT	0.88	0.88	0.88	0.88
NBC	0.82	0.83	0.82	0.82
ANN	0.90	0.90	0.90	0.90
KNN	0.86	0.86	0.86	0.85
RF	0.92	0.92	0.92	0.91

According to the experimental findings shown in Tables 4.2, 4.3, and 4.4, ML classifiers outperform other methods on the 80:20 dataset. With 94% accuracy, RF performs the best in this instance. Training data drops while testing data rises in 50:50 and 30:70 ratios. The fact that the ML classifiers' performance varies little indicates how tolerant they are to small amounts of training data.

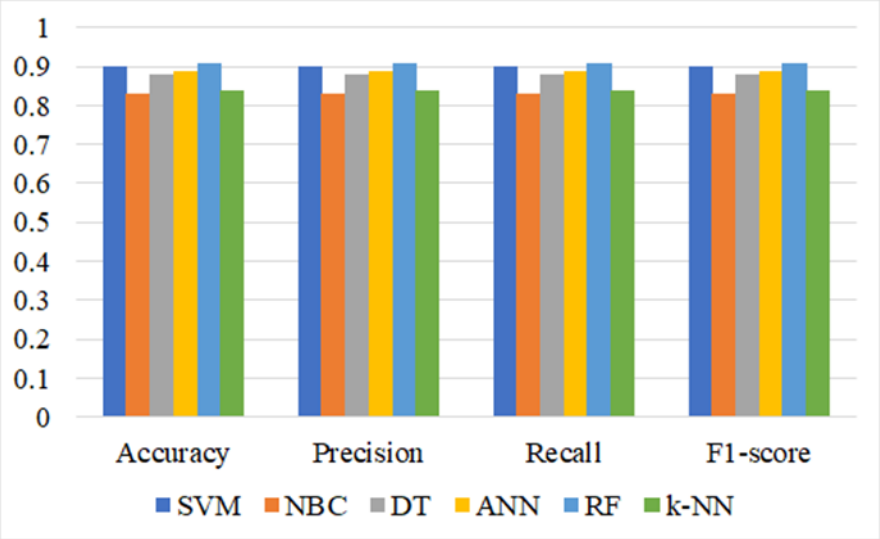


Figure 4.4: Performance of the ML classifier with after applying SMOTE and feature engineering, training and testing ratio is 30:70.

4.4 ML Classifier Performance after Applying NearMiss and Feature Engineering

We use SMOTE, an oversampling approach, in section 4.3. When compared to section 4.2, we have a superior performance. We also use feature engineering and the NearMiss undersampling approach. Table 4.5 displays the results of the experiment. In this case, RF attains 93% recall, f1-score, accuracy, and precision. Similar performance is shown by the 92% accuracy, precision, recall, and f1-score that both SVM and ANN attain. For DT, KNN, and NBC, the corresponding figures are 89%, 87%, and 85%. Figure 4.5 displays a bar chart illustrating the efficacy of the machine learning classifiers that use NearMiss and feature engineering.

Table 4.5: Classification results using ML algorithm after applying NearMiss and feature engineering, training and testing ratio is 80:20

Model	Accuracy	Precision	Recall	F1-score
SVM	0.92	0.92	0.92	0.92
DT	0.89	0.89	0.89	0.89
NBC	0.85	0.85	0.85	0.85
ANN	0.92	0.92	0.92	0.92

KNN	0.87	0.87	0.87	0.87
RF	0.93	0.93	0.93	0.93

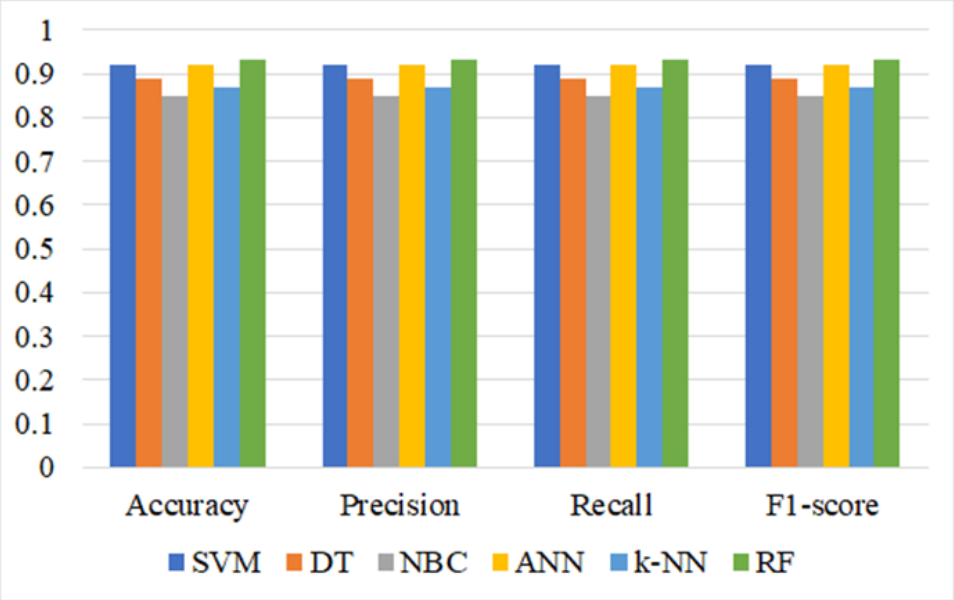


Figure 4.5: Performance of the ML classifier with after applying NearMiss and feature engineering, training and testing ratio is 80:20.

The performance of the ML classifiers in an 80: The table displays the training to testing ratio of 20. 4.5. We test the classifiers' tolerance by doing the same analysis in 50:50 and 30:70 ratios.

The performance of ML algorithms with a 50:50 The ratio of training to testing is shown. in Table 4.6. The outcome of this experiment demonstrates showed the highest 91% f1-score, accuracy, precision, and recall are achieved using RF and SVM. Whereas other classifiers perform worse than RF, SVM, and ANN, ANN achieves 90% accuracy, almost equal to that of RF and ANN. The classifier's performance is shown as a bar chart in Figure 4.6.The ratio of training to testing is shown.

Table 4.6: Classification results using ML algorithm after applying NearMiss and feature engineering, training and testing ratio is 50:50

Model	Accuracy	Precision	Recall	F1-score
SVM	0.91	0.91	0.91	0.90
DT	0.83	0.83	0.83	0.83
NBC	0.86	0.86	0.86	0.86
ANN	0.90	0.90	0.90	0.90
KNN	0.84	0.84	0.84	0.84
RF	0.91	0.91	0.91	0.90

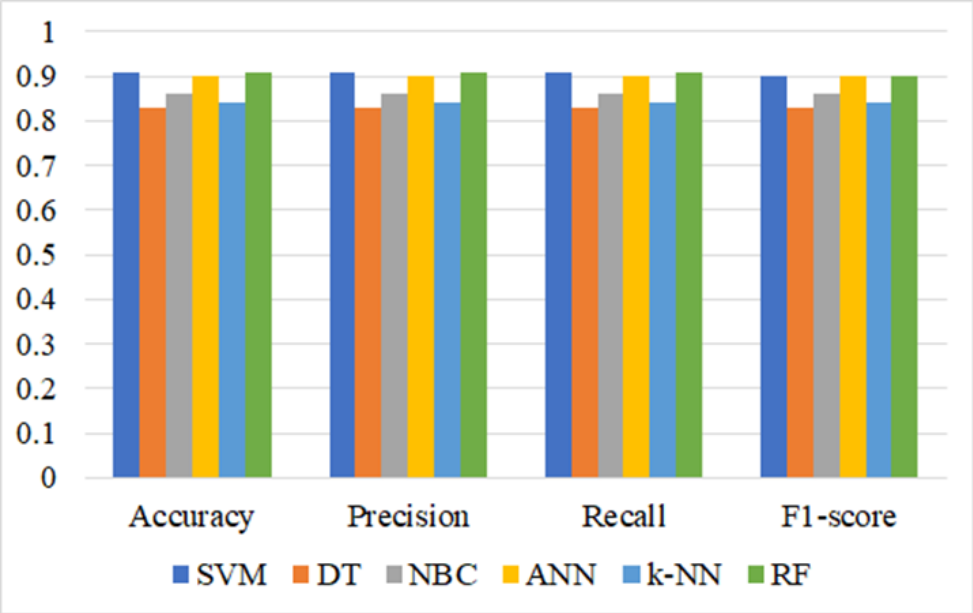


Figure 4.6: Performance of the ML classifier with after applying NearMiss and feature engineering, training and testing ratio is 50:50.

The performance of ML algorithms with a 30:70 The ratio of training to testing is shown. in Table 4.7. According to the testing results, RF obtains the maximum f1-score, accuracy, precision, and recall at 91%. ANN and SVM achieve 89% accuracy. NBC and DT both get 84% and 86%, respectively. In this experiment, the KNN performs the worst. A bar chart illustrating the ML classifiers' performance is shown in Figure 4.7.

Table 4.7: Classification results using ML algorithm after applying NearMiss and feature engineering, training and testing ratio is 30:70

Model	Accuracy	Precision	Recall	F1-score
SVM	0.89	0.89	0.89	0.89
DT	0.84	0.84	0.84	0.84
NBC	0.86	0.86	0.86	0.86
ANN	0.89	0.89	0.89	0.89
KNN	0.81	0.81	0.81	0.81
RF	0.91	0.91	0.91	0.91

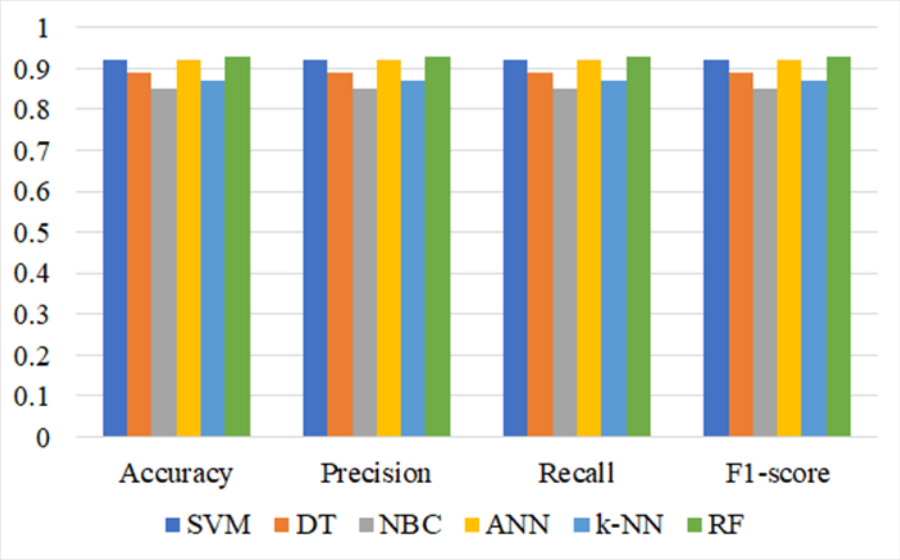


Figure 4.7: Performance of the ML classifier with after applying NearMiss and feature engineering, training and testing ratio is 30:70.

Tables 4.5, 4.6, and 4.7 provide the experimental results, which indicate that ML classifiers outperform 50:50 and 30:70 ratios on 80:20 datasets. With 93% accuracy, RF likewise demonstrates the greatest performance in this testing. Training data drops while testing data rises in 50:50 and 30:70 ratios. The fact that the ML classifiers' performance varies little indicates how tolerant they are to small amounts of training data.

4.5 Performance of Proposed Ensemble Model

We conclude from the aforementioned sections 4.2, 4.3, and 4.4 that the classifier performs better when SMOTE and feature engineering are used. In order to increase accuracy, we examine several classifier combinations to create an ensemble approach.

Ultimately, we arrive at the optimal combination, which we use to create our suggested ensemble Stacking-SRDA. We test our dataset in 80:20, 50:50, and 30:70 training and testing ratios using this Stacking-SRDA. Table 4.8 presents a tabulation of the categorization findings. The suggested ensemble beats the benchmark ML algorithms in every scenario. The maximum 96% The f1-score, recall, accuracy, and precision are achieved by stacking-SRDA in 80:20. This model achieves 93% The model achieved a f1-score of 92% with equal emphasis on accuracy, precision, and recall in a 50:50 ratio. In a 30:70 ratio, stacking-SRDA obtains the maximum 92% Retrieve, F1 score, accuracy, and precision. The bar that is based on accuracy chart depicting the stacking-SRDA performance with a three-dataset split ratio is shown in Figure 4.8.

Table 4.8: Classification results using Stacking-SRDA with 80:20, 50:50, 30:70 training and testing ratio

Training and Testing Ratio	Accuracy	Precision	Recall	F1-score
80:20	0.96	0.96	0.96	0.96
50:50	0.93	0.93	0.93	0.92
30:70	0.92	0.92	0.92	0.92

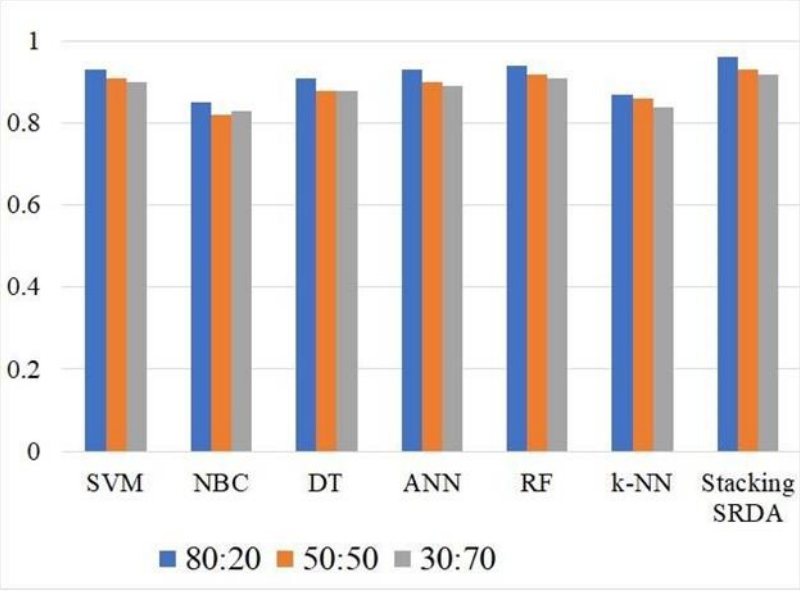
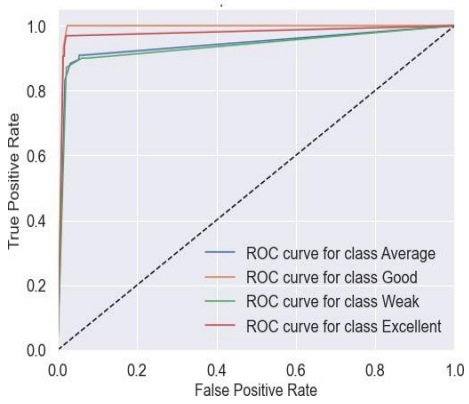


Figure 4.8: Performance of Stacking-SRDA with three-dataset split ratio using accuracy We also analyze the ROC curve to assess the fit's quality. The The Receiver Operating Characteristic (ROC) curve for each individual class of the Key Machine Learning (ML) classifier is available. shown in figure 4.9. The average ROC curve for The SVM and DT shown in Figures 4.10(a) and 4.10(b) show how well the dataset fits the data. Furthermorewe provide the ROC curve in Fig. for every top performance model. 4.6(d), which shows that RF and Stacking-SRDA outperform the other ML modelsused in.



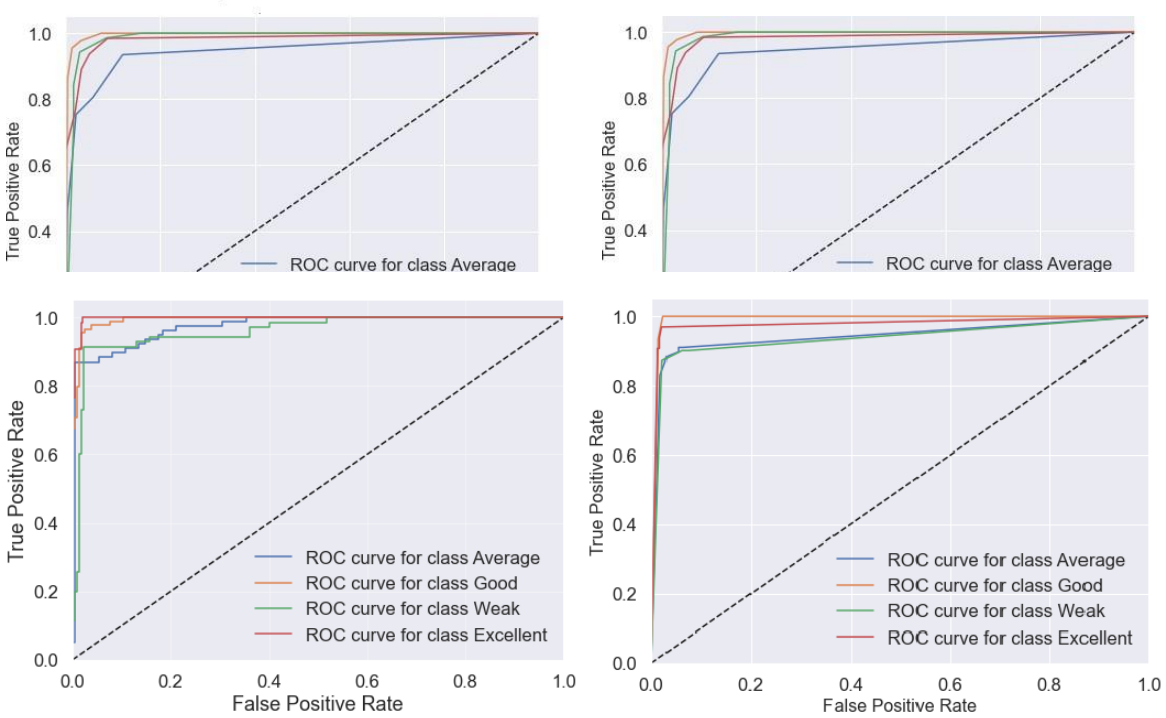


Figure 4.9: ROC curves performance of (a) ANN, (b) DT, (c) GNB, (d) k-NN, (e) SVM and Rf

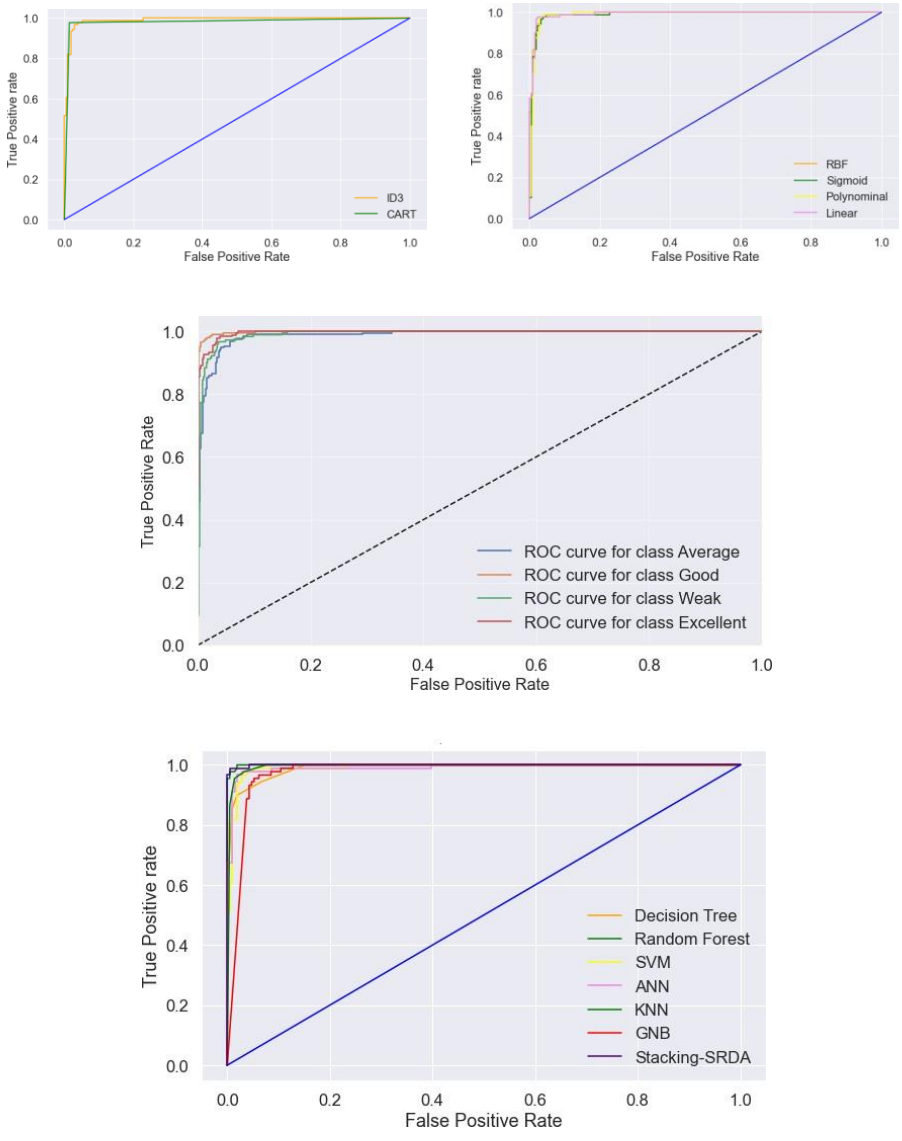


Figure 4.10: ROC curves, (aUtilizing several decision tree algorithms, such as ID3 and CART, (b) Employing Support Vector Machines (SVM) with linear, RBF, polynomial, and sigmoid kernels, The Stacking-SRDA algorithm. (d) Employing NBC, DT, SVM, ANN, RF, k-NN, and Stacking-SRDA

4.6 Significance Test (McNemar Test)

We also use McNemar's Test for the significance test. When two classifiers are compared, McNemar's Test often determines how much more important one classifier is than the other. Thus, contrast Staking-SRDA with the six major machine learning classifiers (SVM, ANN, NBC, DT, KNN, and RF). The McNemar The test results are shown in Table 4.9. The hypothesis model is 1% significant if the P-value in the McNemar test is less than 0.0001, and greater than 0.05, the hypothesis model is 5% significant. when compared to other models. Staking-SRDA is 1% more significant than SVM, DT, NBC, and KNN, and 5% more significant than RF and ANN, according to the results. Where 1% denotes the greatest degree of significance and 5% denotes a lower level of significance than 1%.

Table 4.9: Significance Test (McNemar Test)

Model	P-value
SVM	0.0078125
NBC	$8.364584573428147e\text{-}^{06}$
DT	$3.5881996154785156e^{-05}$
ANN	0.03515625
RF	0.0390625
KNN	$1.6242265701293945e^{-06}$

4.7 Ablation Study for Feature Engineering

For each part of our proposed feature engineering, we conduct ablation experiments at the data distribution and algorithm levels to examine its impact. During the feature engineering process, we convert the text or category value into a numerical value using the One Hot Encoding (OHE) and Label Encoding (LE) approaches. We started by

removing the OHE from the suggested feature engineering and using the LE approach to train the ML algorithms. After that, we take LE out of the suggested model and use the OHE approach to train the ML algorithms. We also provide the results of the suggested feature engineering model. Table 4.10 provides the findings. OHE achieves 95% accuracy whereas LE achieves 94% accuracy. It is evident from the findings that, in terms of performance assessment measures, our suggested model performs better than the two single data conversion methods. Our suggested model has an accuracy of 96%.

Table 4.10: Ablation study for feature engineering

Model	Encoding Techniques	Accuracy	Precision	Recall	F1-score
SVM	LE	0.84	0.83	0.84	0.83
	OHE	0.81	0.82	0.81	0.80
	LE+ OHE	0.93	0.93	0.93	0.93
DT	LE	0.84	0.83	0.84	0.83
	OHE	0.90	0.91	0.90	0.90
	LE+ OHE	0.91	0.91	0.91	0.91
NBC	LE	0.64	0.69	0.64	0.64
	OHE	0.70	0.77	0.70	0.67
	LE+ OHE	0.85	0.86	0.85	0.85
ANN	LE	0.74	0.73	0.74	0.71
	OHE	0.88	0.88	0.88	0.88
	LE+ OHE	0.93	0.93	0.93	0.93
KNN	LE	0.74	0.75	0.74	0.73
	OHE	0.78	0.78	0.78	0.77
	LE+ OHE	0.87	0.87	0.87	0.87
RF	LE	0.91	0.91	0.91	0.91
	OHE	0.94	0.94	0.94	0.94
	LE+ OHE	0.94	0.94	0.94	0.94
Stacking-SRDA	LE	0.94	0.95	0.94	0.94
	OHE	0.95	0.95	0.95	0.95
	LE+ OHE	0.96	0.96	0.96	0.96

4.8 Result of XAI

Lastly, we discuss the model's operation using ML trainable datasets and explainable AI tools including LIME, SHAPASH, and ELI5.

4.8.1 Result using Global XAI

We have used ELI5 and SHAPASH as a global XAI. Initially, feature significance is determined using SHAPASH. The model's feature significance is shown using SHAPASH in Figure 4.10. This feature significance bar chart shows the total absolute contribution values of all the features. In decreasing order, the most significant characteristics are shown. This chart shows that the most significant attribute is the "Problem solved number," followed by onsite involvement, experiences, technical expertise, and STL utilization, in that order.

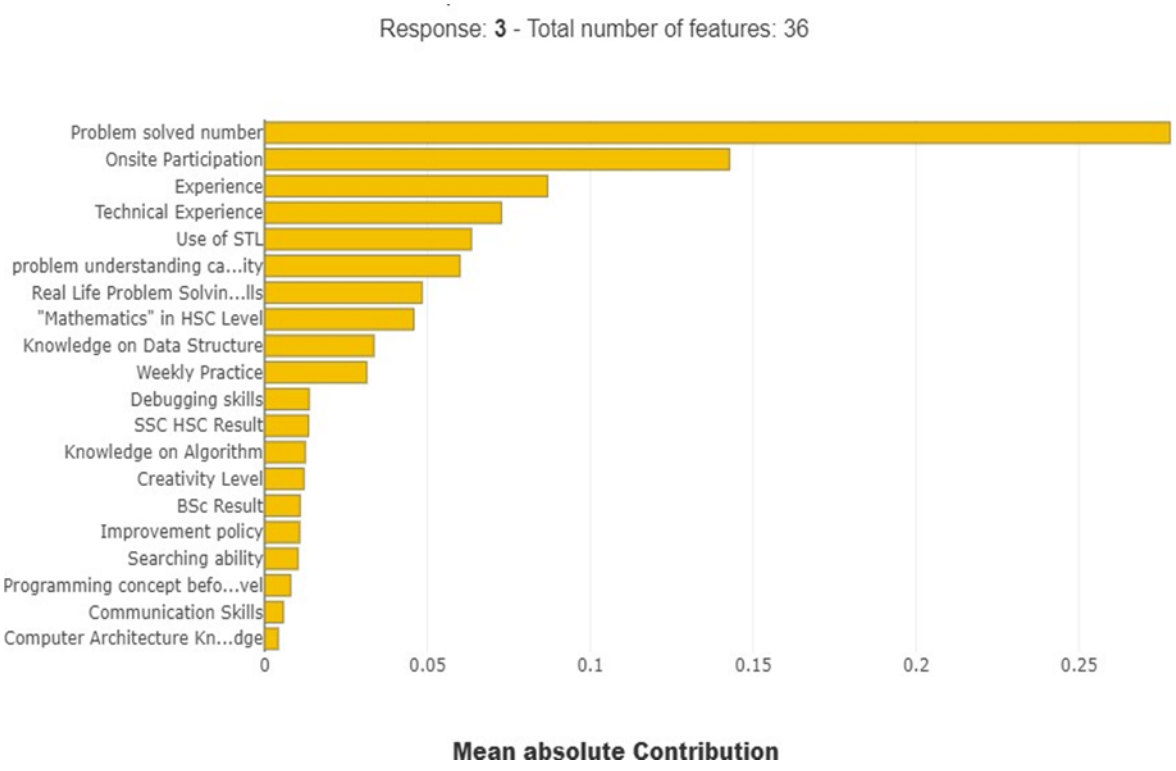


Figure 4.10: Feature importance of the dataset using SHAPASH

We also use ELI5, a program that calculates "Mean Decrease classifiers" or "permutation importance," and for the "Onsite Participation" feature, the chance of decreasing accuracy is 0.0935 ± 0.0363.

Table 4.11: Permutation importance using ELI5

Weight	Feature
0.1015 ± 0.0383	Problem solved number
0.0935 ± 0.0363	Onsite Participation
0.0241 ± 0.0173	Knowledge on Data Structure
0.0241 ± 0.0280	Coding Understand ability
0.0231 ± 0.0121	"Mathematics" in HSC Level
0.0171 ± 0.0080	Real-Life Problem-Solving Skills
0.0151 ± 0.0142	Use of STL
0.0090 ± 0.0075	problem understanding capability
0.0080 ± 0.0049	Debugging skills
0.0080 ± 0.0102	Programming Learning period
0.0080 ± 0.0121	Experience
0.0070 ± 0.0259	Patience in Problem solving
0.0070 ± 0.0197	Searching ability
0.0070 ± 0.0080	Programming concept before BSc level
0.0070 ± 0.0080	Communication Skills
0.0070 ± 0.0080	Learning Speed
0.0060 ± 0.0195	passion on learn code
0.0060 ± 0.0148	Weekly Practice
0.0050 ± 0.0142	Knowledge on Algorithm
0.0050 ± 0.0000	Learning Method

4.5.2 Local XAI using LIME

Then, for local explainability, we used LIME. As shown in Figure 4.12, we first examined a scenario in which the model inferred that a learner was not proficient in programming. It is evident that the orange-colored elements (such as onsite participation, poor practice, and real-life problem-solving skills) or right-side features in this image, when assigned a specific weighted value, may accurately be classified as a weak student in programming. These weighted characteristics significantly affect the model's capacity to predict someone who is not good at programming. Conversely, the model is unaffected by the lower values of Problem Solved Number, Experience, and Use of STL.

Figure 4.12: Local explainability using LIME for class Weak

The explainability of the model in forecasting the "Excellent" programming performance of students is shown in Figure 4.13. It shows that the model's ability to forecast someone who would succeed at programming is greatly influenced by the weighted values of Problem Solved Number, Experience, and Onsite Participation.

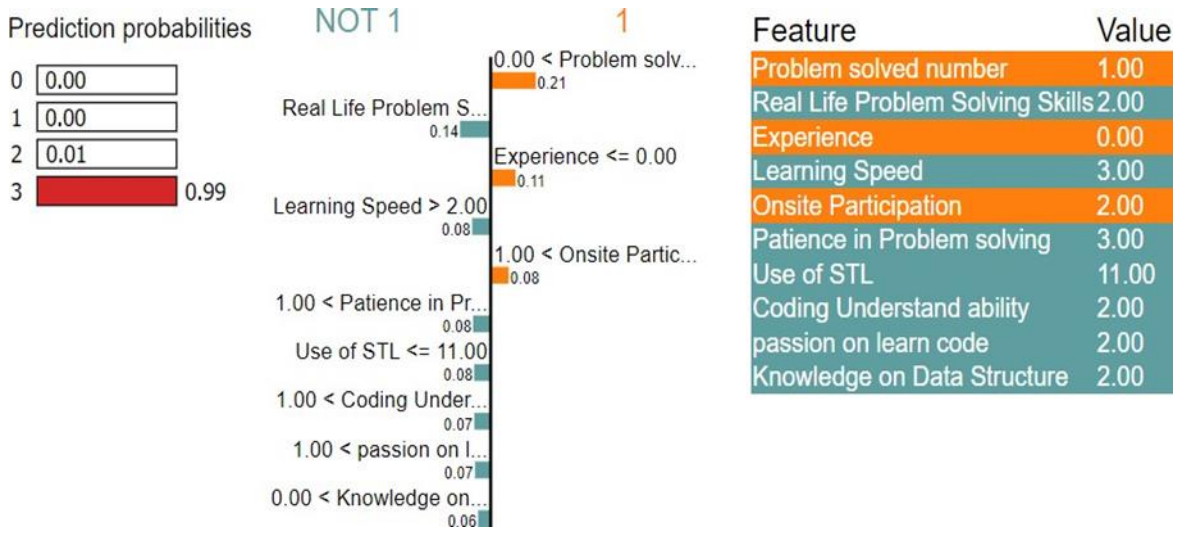


Figure 4.13: Local explainability using LIME for class Excellent

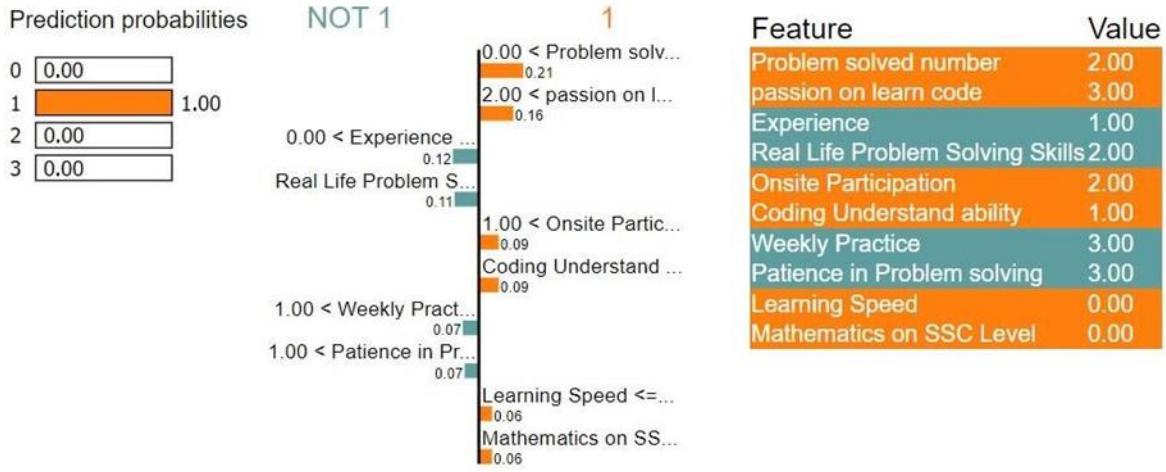


Figure 4.114: Local explainability using LIME for class Average

Figure 4.14 represent the explainability the model to forecast the pupils programming performance as “Average” while Figure 4.15 represent the explainability Using the model to forecast students' proficiency in programming as “Good”.

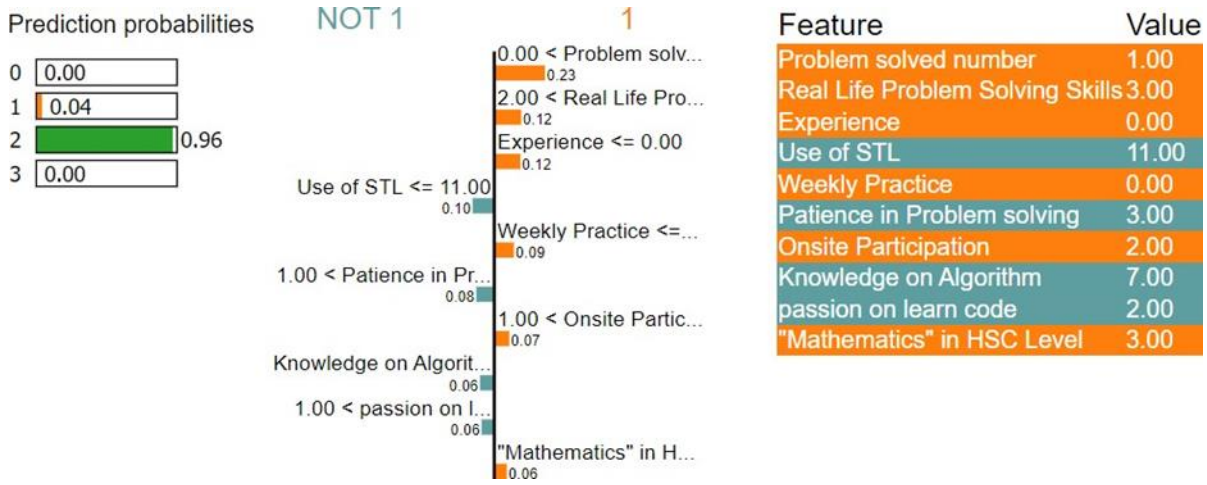


Figure 4.15: Local explainability using LIME for class Good

4.6 Summary

This study's main objective is to explain the model and predict students' programming proficiency more accurately than earlier models. This chapter's goal was to provide an explanation of the study's experimental findings. The chapter includes specifics on the experimental findings and the model's explainability. With the aid of XAI tools and all experimental results, we provide an enhanced and comprehensible EDM system.

Chapter 5

Conclusion and Future Work

5.1 Conclusion

Within this study, we provide an enhanced EDM system that introduces the model's interpretability and forecasts students' programming skill more precisely than earlier models. We look at an efficient feature engineering procedure and an ensemble learning model in an effort to discover a higher accuracy than the earlier models. We use ablation study in the feature engineering process, and our results indicate that the combination of OHE and LE works well. A student's present status is predicted by the categorization module to be outstanding, good, average, or weak. We use the stacking ensemble learning model and train the dataset using six important machine learning classifiers to categorize student performance. We conduct our research using three training and testing ratios within for the purpose of assessing the performance. All of the experiment's results indicate that the RF classifier and stacking-SRDA can both predict students' success in computer programming with an accuracy level of 94% and 96%, respectively. Last but not least, we have used the XAI tools ELI5, SHAPASH, and LIME to provide our suggested EDM system with interpretability. Weak programmers will be able to enhance their programming skills by paying greater attention to their areas of weakness according to the XAI results.

This section offers a concise overview of each chapter of the research and a detailed analysis of its content. the findings. The rationale, goals, and overview of the EDM system are reflected in the first chapter. The thesis offers a summary of the literature on current techniques for predicting students' academic success in the fields of education process mining, machine learning, and education domains. Chapter 3 provides a comprehensive description of the study process and materials used. The data sources, training protocols, performance assessment techniques, a In the domains of education, process mining, machine learning, and education nd a description of the hardware and environment employed are all covered in this chapter. Using performance measurement measures, the experimental results are described in detail in Chapter 4. Additionally, it presents the results

of the XAI tools, talks about the key components, and evaluates the elements influencing the model's capacity to forecast a certain class. Based on the testing data, our suggested EDM system operates more accurately than any of the preceding models.

5.2 Future Work

Future efforts to enhance the Improved Explainable Educational Data Mining (EDM) System for Enhancing Programming Skills might concentrate on many crucial domains to further enhance its effectiveness and practicality.

Investigation of Supplementary Ensemble Learning Models:

Although the present approach utilizes specific machine learning models, future research should explore other ensemble learning methods that may enhance prediction accuracy. One might analyze techniques such as Gradient Boosting Machines (GBM), Extreme Gradient Boosting (XGBoost), and Light Gradient Boosting Machine (LightGBM). These models, renowned for their durability and exceptional performance across several fields, have the potential to greatly enhance the identification and prediction of student performance in programming activities.

Utilization of Advanced Explainable Artificial Intelligence (XAI) Tools:

By incorporating more sophisticated Explainable Artificial Intelligence (XAI) technologies, we may get a more profound understanding of how the utilized models make decisions. Further investigation might be conducted to study the use of SHAP (SHapley Additive exPlanations) and LIME (Local Interpretable Model-agnostic Explanations) tools for the purpose of explaining the results generated by intricate models. By using these tools, instructors may get a more detailed comprehension of the elements that impact student performance, therefore enabling them to customize interventions with greater effectiveness.

Creation of an Internet-based system for providing recommendations:

Based on the empirical findings and insights obtained from the existing system, it is possible to create a web-based recommendation system. This technology would function as an interactive platform where students may get customized suggestions based on their performance data. An automated system might classify pupils according to their aptitudes and provide tailored materials and activities to address their individual areas of weakness. For example, programmers with lower proficiency levels might be detected and provided with additional lectures, coding challenges, and mentorship opportunities to address their skill deficiencies.

Improving User Interface and User Experience:

Further research might also prioritize enhancing the user interface and overall user experience of the EDM system. A user-friendly and intuitive interface is essential to ensure efficient use of the system by both students and instructors. This encompasses the creation of visually comprehensible representations of performance indicators, user-friendly navigation across different functionalities, and interactive components that actively involve users in the process of learning.

Scalability and Generalization:

Another crucial subject for future study is to guarantee the system's scalability and its capacity to generalize across diverse educational environments. To prove its usefulness, the system should undergo testing in a range of situations, including different educational institutions and a variety of programming courses. Furthermore, while considering scalability, it is important to ensure that the system has the ability to effectively manage and process huge datasets and accommodate a substantial number of users at the same time, while maintaining optimal performance.

Integration of Feedback Mechanisms:

Integrating feedback mechanisms into the system may provide a perpetual cycle of improvement. Developers may find areas for improvement and create iterative enhancements by soliciting input from students and instructors on the recommendations

and general functionality of the system. This technique guarantees that the system adapts and develops based on real user demands and experiences.

Longitudinal studies:

It is crucial to carry out longitudinal studies in order to monitor the lasting effects of the EDM system on students' programming abilities. These studies may provide useful insights into the long-term performance of the system and aid in finding any enduring advantages or areas that need change.

Engaging with educational stakeholders:

Future endeavors should also include extensive cooperation with educational stakeholders, such as teachers, curriculum creators, and legislators. Their feedback may inform the creation of features that are in line with educational standards and pedagogical best practices. This partnership guarantees that the system is not only highly advanced in technology, but also well-grounded in pedagogy and very practical in real-life educational environments.

By prioritizing these areas, the future advancement of the Enhanced Explainable EDM System may result in more precise forecasts, enhanced assistance for students, and a more significant overall influence on programming education.

Reference

1. R Baker and George Siemens. Educational data mining and learning analytics. Cambridge Handbook of the Learning Sciences: 2014.
2. Madni, H. A., Anwar, Z., & Shah, M. A. (2017, September). Data mining techniquesand applications—A decade review. In *2017 23rd international conference on* automation and computing (ICAC) (pp. 1-7). IEEE.
3. R. Baker, "Data mining for education," in International Encyclopedia of Education, B. McGaw, P. Peterson, and E. Baker, Eds., 3rd ed. Oxford, U.K.: Elsevier, 2010
4. T. Barnes, M. Desmarais, C. Romero, and S. Ventura, presented at the 2nd Int. Conf. Educ. Data Mining, Cordoba, Spain, 2009.
5. Aldowah, H., Al-Samarraie, H., and Fauzy, W. M. (2019) Educational Data Mining andLearning Analytics for 21st Century Higher Education: A Review and Synthesis. Telematics and Informatics, 37, 13–49.
6. Livieris, I. E., Kotsilieris, T., Tampakas, V., and Pintelas, P. (2019) Improving the Evaluation Process of Students' Performance Utilizing a Decision Support Software.Neural Computing and Applications, 31, 1683–1694.
7. Romero, C., & Ventura, S. (2010). Educational data mining: a review of the state of theart. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (applications and reviews)*, *40*(6), 601-618.
8. Ashraf, A., Anwer, S., & Khan, M. G. (2018). A Comparative study of predicting student's performance by use of data mining techniques. American Scientific ResearchJournal for Engineering, Technology, and Sciences (ASRJETS), 44(1), 122-136.
9. ROMERO, C., and VENTURA, S. (2013) Data mining in education. WileyInterdisciplinary Reviews: Data Mining and Knowledge Discovery, 3(1), pp. 12-27. doi: 10.1002/widm.1075
10. A. Farissi, H. Mohamed Dahlan, and Samsuryadi, ''Genetic algorithm-based feature selection with ensemble methods for student academic performance prediction,'' *J.Phys., Conf. Ser.*, vol. 1500, no. 1, Apr. 2020, Art. no. 012110.

11. H. Turabieh, S. A. Azwari, M. Rokaya, W. Alosaimi, A. Alharbi, W. Alhakami, and M.Alnfiai, ‘‘Enhanced Harris hawks optimization as a feature selection for the prediction of student performance,’’ *Computing*, vol. 7, pp. 1417–1438, Jan. 2021.
12. A. Nabil, M. Seyam, and A. Abou-Elfetouh, ‘‘Prediction of students’ academic performance based on courses’ grades using deep neural networks,’’ IEEE Access, vol. 9, pp. 140731–140746, 2021.
13. L. Gao, Z. Zhao, C. Li, J. Zhao, and Q. Zeng, ‘‘Deep cognitive diagnosis model for predicting students’ performance,’’ Future Gener. Comput. Syst., vol. 126, pp. 252–262, Jan. 2022.
14. L. Gao, Z. Zhao, C. Li, J. Zhao, and Q. Zeng, ‘‘Deep cognitive diagnosis model for predicting students’ performance,’’ Future Gener. Comput. Syst., vol. 126, pp. 252–262, Jan. 2022. 15. 16
15. V. G. Karthikeyan, P. Thangaraj, and S. Karthik, ‘‘Towards developing hybrid educational data mining model (HEDM) for efficient and accurate student performance evaluation,’’ Soft Comput., vol. 24, no. 24, pp. 18477–18487, Dec. 2020.
16. L. M. Crivei, G. Czibula, G. Ciubotariu, and M. Dindelegan, ‘‘Unsupervised learning based mining of academic data sets for students’ performance analysis,’’ in Proc. IEEE 14th Int. Symp. Appl. Comput. Intell. Informat. (SACI), Timisoara, Romania, May 2020, pp. 11–16.
17. K. Okoye, A. Arrona-Palacios, C. Camacho-Zuæiga, J. A. G. Achem, J. Escamilla, and S. Hosseini, ‘‘Towards teaching analytics: A contextual model for analysis of students’ evaluation of teaching through text mining and machine learning classification,’’ Educ. Inf. Technol., vol. 1, pp. 1–43, Oct. 2021.
18. E. S. V. Kumar, S. A. A. Balamurugan, and S. Sasikala, ‘‘Multi-tier student performance evaluation model (MTSPEM) with integrated classification techniques for educational decision making,’’ Int. J. Comput. Intell. Syst., vol. 14, no. 1, pp. 1796–1808, Mar. 2021.
19. A. Siddiqa, S. A. Z. Naqvi, M. Ahsan, A. Ditta, H. Alquhayz, M. A. Khan, and M. A. Khan, ‘‘An improved evolutionary algorithm for data mining and knowledge discovery,’’ Comput., Mater. Continua, vol. 71, no. 1, pp. 1233–1247, 2022.
20. J. Bennedsen and M. E. Caspersen., " Failure rates in introductory programming," ACM SIGCSE Bulletin 39.2, pp. 32-36, 2007.

22. Ahmed Mueen, Bassam Zafar, Umar Manzoor, "Modeling and Predicting Students' Academic Performance Using Data Mining Techniques", International Journal of Modern Education and Computer Science(IJMECS), Vol.8, No.11, pp.36-42, 2016.

23. Neeta Sharma, Shanmuganathan Appukutti, Umang Garg, Jayati Mukherjee, Sneha Mishra, "Analysis of Student's Academic Performance based on their Time Spent on Extra-Curricular Activities using Machine Learning Techniques", International Journal of Modern Education and Computer Science(IJMECS), Vol.15, No.1, pp. 46-57, 2023.

DOI:10.5815/ijmecs.2023.01.04

24. Sudais, M., Safwan, M., Khalid, M. A., & Ahmed, S. (2022). Students' Academic Performance Prediction Model Using Machine Learning. Research Square; 2022. DOI: 10.21203/rs.3.rs-1296035/v1.

25. Masangu, Lonia, Ashwini Jadhav, and Ritesh Ajoodha. "Predicting Student Academic Performance Using Data Mining Techniques." Advances in Science, Technology and Engineering Systems Journal 6.1 (2021): 153-163.

26. Hasan, R., Palaniappan, S., Mahmood, S., Abbas, A., Sarker, K. U., & Sattar, M. U. (2020). Predicting student performance in higher educational institutions using video learning analytics and data mining techniques. Applied Sciences, 10(11), 3894.

27. K. T. Chui, D. C. L. Fung, M. D. Lytras, and T. M. Lam, Predicting at-risk university students in a virtual learning environment via a machine learning algorithm," Computers in Human Behavior, pp. 0{1, 6 2018. 25, 29, 41

28. V. L. Migu´eis, A. Freitas, V. P. J. Garcia, and A. Silva, Early segmentation of students according to their academic performance: A predictive modelling approach," Decis.

Support Syst., vol. 115, pp. 36{51, 11 2018. 27, 31, 41

29. Takamatsu, K., Murakami, K., Oshiro, T., Sugiura, A., Bannaka, K., and Nakata, Y. (2019) Predicting the Probability of Student's Academic Abilities and Progress with EMIR and Data from Current and Graduated Students. 2019 8th International Congress

on Advanced Applied Informatics (IIAI-AAI), Toyama, Japan, July, pp. 359–362. IEEE.

30. Abidi, S., Hussain, M., Xu, Y., and Zhang, W. (2018) Prediction of Confusion Attempting Algebra Homework in an Intelligent Tutoring System through Machine Learning Techniques for Educational Sustainable Development. Sustainability, 11, 105–

125.

31. Kabathova, Janka, and Martin Drlik. "Towards predicting student's dropout in university courses using different machine learning techniques." Applied Sciences 11.7 (2021):3130.

32. Livieris, I. E., Kotsilieris, T., Tampakas, V., and Pintelas, P. (2019) Improving the Evaluation Process of Students' Performance Utilizing a Decision Support Software. *Neural Computing and Applications*, **31**, 1683–1694

33. Rodrigues, R. L., Ramos, J. L. C., Silva, J. C. S., Dourado, R. A., and Gomes, A. S. (2019) Forecasting Students' Performance Through Self-Regulated Learning Behavioral Analysis: International Journal of Distance Education Technologies, 17, 52–74

34. Yihun, Meseret. "Global challenges of students dropout: A prediction model development using machine learning algorithms on higher education datasets." SHS Web of Conferences. Vol. 129.2021

35. Hashim, A. S., Awadh, W. A., & Hamoud, A. K. (2020, November). Student performance prediction model based on supervised machine learning algorithms. In *IOPConference Series: Materials Science and Engineering* (Vol. 928, No. 3, p. 032019). IOP Publishing.

36. Marbouti, F., Diefes-Dux, H. A., and Madhavan, K. (2016) Models for Early Predictionof At-Risk Students in a Course Using Standards-Based Grading. Computers & Education, 103, 1–15

37. M. Hlosta, Z. Zdrahal, and J. Zendulka, Ouroboros: early identification of at-riskstudents without models based on legacy data," *LAK17 - Seventh International Learning Analytics Knowledge Conference*, pp. 6{15, 2017. 28, 29, 33, 42

38. Pathan, A. A., Hasan, M., Ahmed, M. F., and Farid, D. M. (2014) Educational Data Mining: A Mining Model for Developing Students' Programming Skills. The 8th International Conference on Software, Knowledge, Information Management and Applications (SKIMA 2014), Dhaka, Bangladesh, December, pp. 1– 5. IEEE.

Improved Explainable Educational Data Mining System for Enhancing Programming Skills

ORIGINALITY REPORT

22%	19%	16%	11%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	academic.oup.com Internet Source	4%
2	Md Abu Marjan, Md Palash Uddin, Masud Ibn Afjal. "An Educational Data Mining System For Predicting And Enhancing Tertiary Students' Programming Skill", The Computer Journal, 2022 Publication	2%
3	dspace.daffodilvarsity.edu.bd:8080 Internet Source	2%
4	Guiyun Feng, Muwei Fan, Yu Chen. "Analysis and Prediction of Students' Academic Performance Based on Educational Data Mining", IEEE Access, 2022 Publication	2%
5	bit.kuas.edu.tw Internet Source	1%
6	mro.massey.ac.nz Internet Source	1%