Unveiling Student Success: A Multifaceted Approach with Learning Coefficients and Beyond

A Project Report submitted in the partial fulfillment

of the Requirements for the award of the degree

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE AND ENGINEERING

Submitted by

T. Mahith	**(21471A0565)**
K. Ravi Naik	**(21471A0528)**
K. Raju	**(21471A0529)**

Under the esteemed guidance of

N. VijayaKumar, M.E

Associate Professor

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

NARASARAOPETA ENGINEERING COLLEGE: NARASAROPET

(AUTONOMOUS)

Accredited by NAAC with A+ Grade and NBA under Tire -1

NIRF rank in the band of 201-300 and an ISO 9001:2015 Certified

Approved by AICTE, New Delhi, Permanently Affiliated to JNTUK, Kakinada

KOTAPPAKONDA ROAD, YALLAMANDA VILLAGE, NARASARAOPET- 522601

2024-2025

NARASARAOPETA ENGINEERING COLLEGE
(AUTONOMOUS)
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CERTIFICATE

This is to certify that the project that is entitled with the name **"**Unveiling Student Success: A Multifaceted Approach with Learning Coefficients and Beyond**"** is a bonafide work done by the team T. Mahith (21471A0565), K. Ravi Naik (21471A0528), K. Raju (21471A0529) in partial fulfillment of the requirements for the award of the degree of BACHELOR OF TECHNOLOGY in the Department of COMPUTER SCIENCE AND ENGINEERING during 2024-2025.

PROJECT GUIDE
Nukala VijayaKumar, M.E
Associate Professor

PROJECT CO-ORDINATOR
Dodda Venkata Reddy, M.Tech,(Ph.D)
Assistant Professor

HEAD OF THE DEPARTMENT
Dr. S. N. Tirumala Rao, M.Tech., Ph.D.,
Professor & HOD

EXTERNAL EXAMINER

DECLARATION

We declare that this project work titled " UNVEILING STUDENT SUCCESS: A MULTIFACETED APPROACH WITH LEARNING COEFFICIENTS AND BEYOND " is composed by ourselves that the work contain here is our own except where explicitly stated otherwise in the text and that this work has been submitted for another degree or professional qualification except as specified.

T. Mahith (21471A0565)
K. Ravi Naik (21471A0528)
K. Raju (21471A0529)

ACKNOWLEDGEMENT

We wish to express our thanks to carious personalities who are responsible for the completion of the project. We are extremely thankful to our beloved chairman Sri **M. V. Koteswara Rao**, **B.Sc.**, who took keen interest in us in every effort throughout thiscourse. We owe out sincere gratitude to our beloved principal **Dr. S. Venkateswarlu, Ph.D.**, for showing his kind attention and valuable guidance throughout the course.

We express our deep-felt gratitude towards **Dr. S. N. Tirumala Rao, M.Tech., Ph.D.,** HOD of CSE department and also to our guide **N. VijayaKumar, M. Tech** of CSE department whose valuable guidance and unstinting encouragement enable us to accomplish our project successfully in time.

We extend our sincere thanks towards **Dodda Venkata Reddy, M. Tech, (Ph. D).** Assistant professor & Project coordinator of the project for extending his encouragement. Their profound knowledge and willingness have been a constant source of inspiration for us throughout this project work.

We extend our sincere thanks to all other teaching and non-teaching staff to department for their cooperation and encouragement during our B. Tech degree.

We have no words to acknowledge the warm affection, constant inspiration and encouragement that we received from our parents.

We affectionately acknowledge the encouragement received from our friends and those who involved in giving valuable suggestions had clarifying out doubts which had really helped us in successfully completing our project.

By

T. Mahith (21471A0565)
K. RaviNaik (21471A0528)
K. Raju(21471A0529)

NARASARAOPETA ENGINEERING COLLEGE
(AUTONOMOUS)

INSTITUTE VISION AND MISSION

INSTITUTION VISION

To emerge as a Centre of excellence in technical education with a blend of effective student centric teaching learning practices as well as research for the transformation of lives and community

INSTITUTION MISSION

M1: Provide the best class infra-structure to explore the field of engineering and research

M2: Build a passionate and a determined team of faculty with student centric teaching, imbibing experiential, innovative skills

M3: Imbibe lifelong learning skills, entrepreneurial skills and ethical values in students for addressing societal problems

NARASARAOPETA ENGINEERING COLLEGE
(AUTONOMOUS)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

VISION OF THE DEPARTMENT

To become a center of excellence in nurturing the quality Computer Science & Engineering professionals embedded with software knowledge, aptitude for research and ethical values to cater to the needs of industry and society.

MISSION OF THE DEPARTMENT

The department of Computer Science and Engineering is committed to

M1: Mould the students to become Software Professionals, Researchers and Entrepreneurs by providing advanced laboratories.

M2: Impart high quality professional training to get expertise in modern software tools and technologies to cater to the real time requirements of the industry.

M3: Inculcate team work and lifelong learning among students with a sense of societal and ethical responsibilities.

NARASARAOPETA ENGINEERING COLLEGE
(AUTONOMOUS)

Program Specific Outcomes (PSO's)

PSO1: Apply mathematical and scientific skills in numerous areas of Computer Science and Engineering to design and develop software-based systems.

PSO2: Acquaint module knowledge on emerging trends of the modern era in Computer Science and Engineering

PSO3: Promote novel applications that meet the needs of entrepreneur, environmental and social issues.

NARASARAOPETA ENGINEERING COLLEGE
(AUTONOMOUS)

Program Educational Objectives (PEO's)

The graduates of the programme are able to:

PEO1: Apply the knowledge of Mathematics, Science and Engineering fundamentals to identify and solve Computer Science and Engineering problems.

PEO2: Use various software tools and technologies to solve problems related to academia, industry and society.

PEO3: Work with ethical and moral values in the multi-disciplinary teams and can communicate effectively among team members with continuous learning.

PEO4: Pursue higher studies and develop their career in software industry.

NARASARAOPETA ENGINEERING COLLEGE
(AUTONOMOUS)

Program Outcomes

1. Engineering knowledge: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.

2. Problem analysis: Identify, formulate, research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.

3. Design/development of solutions: Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.

4. Conduct investigations of complex problems: Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.

5. Modern tool usage: Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.

6. The engineer and society: Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.

7. Environment and sustainability: Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.

8. Ethics: Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.

9. Individual and team work: Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.

10. Communication: Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.

11. Project management and finance: Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.

12. Life-long learning: Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

Project Course Outcomes (CO'S):

CO421.1: Analyze the System of Examinations and identify the problem.

CO421.2: Identify and classify the requirements.

CO421.3: Review the Related Literature

CO421.4: Design and Modularize the project

CO421.5: Construct, Integrate, Test and Implement the Project.

CO421.6: Prepare the project Documentation and present the Report using appropriate method.

Course Outcomes – Program Outcomes mapping

	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12	PSO1	PSO2	PSO3
C421.1		√											√		
C421.2	√		√		√								√		
C421.3				√		√	√	√					√		
C421.4			√			√	√	√					√	√	
C421.5					√	√	√	√	√	√	√	√	√	√	√
C421.6									√	√	√		√	√	

Course Outcomes – Program Outcome correlation

	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12	PSO1	PSO2	PSO3
C421.1	2	3											2		
C421.2			2		3								2		
C421.3				2		2	3	3					2		
C421.4			2			1	1	2					3	2	
C421.5					3	3	3	2	3	2	2	1	3	2	1
C421.6									3	2	1		2	3	

Note: The values in the above table represent the level of correlation between CO's and PO's:

1. Low level
2. Medium level
3. High level

Project mapping with various courses of Curriculum with Attained PO's:

Name of the course from which principles are applied in this project	Description of the device	Attained PO
C2204.2, C22L3.2	The study predicts student performance using advanced machine learning and deep learning models, achieving high accuracy with methods like Random Forest, XGBoost, CNN, and LSTM.	PO1, PO3
CC421.1, C2204.3, C22L3.2	Each and every requirement is critically analyzed, the process model is identified	PO2, PO3
CC421.2, C2204.2, C22L3.3	Logical design is done by using the unified modelling language which involves individual team work	PO3, PO5, PO9
CC421.3, C2204.3, C22L3.2	Each and every module is tested, integrated, and evaluated in our project	PO1, PO5
CC421.4, C2204.4, C22L3.2	Documentation is done by all our four members in the form of a group	PO10
CC421.5, C2204.2, C22L3.3	Each and every phase of the work in group is presented periodically.	PO10, PO11
C2202.2, C2203.3, C1206.3, C3204.3, C4110.2	The implementation is complete, and the project will be utilized by students and educators. Future updates may incorporate enhancements, such as improving the model's accuracy and expanding its capabilities for predicting student performance trends.	PO4, PO7
C32SC4.3	The physical design includes website to check whether it is a correct prediction or not	PO5, PO6

ABSTRACT

The student performance is examined in this study using a number of methods of Educational Data Mining (EDM), Clustering and classification techniques are employed to classify the course as well as the performance in the entrance examination. The results obtained show that the Random Forest and XG Boost which are machine learning models outperform traditional methods for predicting student success. Moreover, CNN and LSTM Networks, which are deep learning models, improve prediction accuracy even further. Conducted through metrics like accuracy, precision, recall and F1-score, this study shows that any form of recognition of the pattern, in this case, the early one, helps to reduce failure rates to considerable extents. The results of this study suggest that there is a potential scope for further improving prediction algorithms and management of educational resources, which are of great relevance to the institutions to further the student success

INDEX

LIST OF FIGURES & TABLES

1.INTRODUCTION

Introduction

Education is a key factor in shaping society, with academic performance serving as a critical metric for evaluating student success and institutional quality. The increasing adoption of technology in education, particularly in response to challenges like the COVID-19 pandemic, has accelerated the need for data-driven decision-making. Despite significant advancements in student performance prediction, gaps remain in leveraging comprehensive datasets for accurate and early forecasting.

Educational Data Mining (EDM) and Machine Learning (ML) techniques offer a promising approach to analyzing student performance patterns. By applying classification and clustering methods, this research explores how different algorithms, including Random Forest, XGBoost, Support Vector Machines (SVM), and Logistic Regression, perform in predicting student success. Furthermore, deep learning models such as Convolutional Neural Networks (CNN) and Long Short-Term Memory (LSTM) networks enhance accuracy by capturing complex relationships in student data.

This study employs a dataset comprising academic scores, entrance examination results, and socio-demographic factors to develop predictive models. By evaluating these models using metrics like accuracy, precision, recall, and F1-score, the research demonstrates the effectiveness of early performance prediction in reducing dropout rates and optimizing academic resources. The findings emphasize the need for continuous improvement in predictive models, paving the way for personalized learning interventions and enhanced student outcomes. Education plays a crucial role in national progress and is a key tool for success in life. Educational institutions aim to provide quality education to enhance students' learning experiences. A major challenge faced by students, particularly in computer programming courses, is the high failure and dropout rates.

Artificial Intelligence (AI) and Machine Learning (ML) have been widely applied in various domains, including image classification, speech recognition, and text translation. In the field of Educational Data Mining (EDM), these technologies are used to analyze large volumes of student data from different educational environments. EDM applies data mining techniques such as classification, regression, time series analysis, and association rule mining to evaluate student performance and improve the

learning process.

Machine learning models, such as Random Forest and XGBoost, play a critical role in predicting academic performance. These algorithms are particularly suited for processing large datasets with numerous variables.

This project introduces the concept of "Learning Coefficients," a novel approach that uses trajectory-based computerized adaptive assessments to measure student progress throughout a course. Learning coefficients serve as quantifiable metrics, offering students a structured way to track their performance and focus on areas needing improvement. Before incorporating these coefficients as predictive features, their correlation with other key academic and demographic factors is analysed using Pearson's correlation coefficient.

The study further compares the performance of various regression-based machine learning models, including Decision Trees, Random Forest, Support Vector Regression, Linear Regression, and Artificial Neural Networks. Experimental results show that Linear Regression achieves the highest accuracy (97%) in predicting student performance.

This research highlights the importance of integrating dynamic, real-time evaluation metrics into predictive models. By using learning coefficients, institutions can provide students with meaningful feedback, enabling them to enhance their academic performance before final assessments.

Predicting student performance is essential for improving academic outcomes and ensuring effective educational planning. Academic performance serves as a key indicator of student abilities and the quality of education in universities. However, while most studies focus on identifying common characteristics among students, they often fail to consider individual differences, which play a crucial role in understanding learning patterns. To address this limitation, this paper proposes an innovative student performance prediction approach that integrates both common and individual characteristics. The study introduces a new clustering technique by combining the Relationship Matrix-Based Bipartite Network (RMBN) with Louvain Clustering to effectively group student samples in multidimensional discrete data.

Additionally, the paper presents a Hybrid Neural Network Model (RMHNN) that enhances prediction accuracy by overcoming the challenges associated with discrete data features. Experimental results demonstrate that the proposed model achieves an impressive 93.1% accuracy and an F1-score of 90.45%, significantly outperforming

traditional prediction methods. By leveraging these advanced techniques, the study enables educators to provide personalized support to students, helping them improve their academic performance and succeed in their learning journey.

This study utilizes student data from a public four-year university to develop predictive models for academic performance. The research applies several machines learning algorithms, including Deep Neural Networks (DNN), Decision Trees (DT), Logistic Regression (LR), Support Vector Classifiers (SVC), K-Nearest Neighbours (KNN), Random Forest (RF), and Gradient Boosting (GB). The goal is to predict student success in a Data Structures course based on their first-year course grades.

The study follows the Knowledge Discovery in Databases (KDD) process, which includes data collection, preprocessing, mining, and performance evaluation. Experimental results indicate that the proposed DNN model achieves an accuracy of 89%, outperforming traditional ML models like Decision Trees, Logistic Regression, and K-Nearest Neighbours.

The findings emphasize the importance of early prediction in student academic performance to support student success and retention. The results suggest that DNN-based models can serve as effective tools for identifying at-risk students and improving the learning process.

With the advancement of information technology in higher education, institutions have collected vast amounts of student academic performance data, enabling data-driven innovations in education. The rise of smart campuses and big data analytics has further supported research on early academic warning systems. These systems provide several benefits, including enabling students to assess their academic abilities, allowing educators to identify struggling students for timely intervention, and assisting administrators in predicting overall teaching quality and academic trends.

In China, the expansion of higher education has led to a growing student population, raising concerns about maintaining teaching quality. Despite government policies aimed at ensuring academic standards, challenges such as faculty allocation and resource distribution continue to impact student success. Consequently, researchers have developed academic tracking mechanisms to identify students at risk of poor performance early.

However, most existing models primarily focus on subjective student factors such as grades and family background while overlooking external influences like course difficulty, teaching style, and credit distribution. Furthermore, the COVID-19

pandemic has disrupted traditional education, making many current prediction models ineffective. Online and offline learning environments require different analytical approaches, yet existing models fail to account for this complexity.

To address these limitations, this study proposes a new deep learning model, SAL&EC (Subjective Ability Level & External Characteristics), that integrates various internal and external factors to provide an early warning system for student performance. Unlike previous studies that focus solely on individual students, this model incorporates historical course performance, instructional variations, and external disruptions like COVID-19. The proposed system demonstrates practical significance by improving predictive accuracy and adapting to hybrid learning environments.

In higher education institutions (HEIs), student academic performance is a crucial indicator for evaluating educational quality. Universities maintain student records, including grades and final exam results, to assess course achievements each semester. These records provide valuable data that can be leveraged for predictive analytics to enhance academic planning and intervention strategies.

Predictive analytics has gained significant traction in HEIs due to its ability to extract meaningful insights from large datasets. Researchers have applied various machine learning techniques to predict student grades, aiming to improve academic outcomes. However, a major challenge in student grade prediction is the imbalanced nature of academic datasets, which can lead to biased predictions and inaccurate classification of students' performance.

In recent years, the growing adoption of online learning platforms has revolutionized the educational sector. The shift towards e-learning, especially during the COVID-19 pandemic, has generated a vast amount of data related to student behavior and academic performance. This data can be effectively utilized to build intelligent assistive systems that track student activity and provide personalized feedback to improve learning outcomes.

Traditional methods of student performance prediction often rely on static features like academic records and survey-based data collection. However, these methods fail to capture the dynamic and social aspects of student behaviour. To address this limitation, this study leverages the Community of Inquiry (CoI) framework, which focuses on social, cognitive, and teaching presence, to select the most influential features for GPA prediction.

The primary objective of this research is to improve the accuracy and efficiency of

GPA prediction models by introducing a Social Clustering-Based Regression Model. The proposed approach combines k-means clustering based on student similarity with LassoCV regression to predict student performance. The Student Life dataset, which contains data on student behavior, mental health, and social interaction, is used for this purpose.

2. LITERATURE SURVEY

Alhazmi et al. [3] conducted a study on early prediction of students' performance in higher education, highlighting the integration of machine learning algorithms with subjective and objective elements to improve prediction accuracy. The study emphasizes the potential of these methods in identifying students at risk of poor academic performance.

Liu et al. [4] introduced a multiple features fusion approach using enhanced deep learning mechanisms, particularly CNNs, for student performance prediction. Their study showed that combining different attributes, such as entrance examination scores and initial course results, significantly improved prediction accuracy.

Butt et al. [5] presented a multi-model ensemble approach for performance prediction in higher education. They utilized machine learning models like Random Forest and XGBoost, showcasing their ability to handle large datasets and achieve high accuracy rates in predicting student success.

Bujang et al. [6] explored multiclass prediction models for student grade prediction using machine learning. The study emphasized the effectiveness of supervised learning techniques, such as Logistic Regression and Support Vector Machines (SVM), in classroom performance forecasting.

Pelima et al. [9] provided a systematic review of machine learning applications in predicting university student graduation. They highlighted how models like CNN and LSTM can analyze sequential data, such as semester-wise grades, to accurately forecast academic outcomes.

The prediction of students' academic performance has been extensively studied in the field of Educational Data Mining (EDM). Various researchers have applied different machine learning techniques to predict student success, retention rates, and failure risks. Several approaches have been proposed to enhance the accuracy of these predictions by utilizing different features such as past academic performance, demographic data, economic status, and behavioral characteristics.

Some studies have focused on the use of machine learning models, such as decision trees, support vector machines, and artificial neural networks, to predict students' performance based on academic history

Many studies highlight the importance of early prediction in academic settings. Tsiakmaki et al. (2020) developed a predictive model that uses students' grades from their first semester to predict their performance in the subsequent semester.

Students' academic performance is one of the most important factors in higher education. Several researchers have used Educational Data Mining (EDM) applications to predict and evaluate students' academic performance in the decision-making process and to understand the learning process [4].

Many studies have used different machine learning (ML) techniques and datasets to predict students' academic success. Various features such as cumulative grade point average (CGPA), quiz grades, midterm marks, assessments, attendance, and lab work have been used to build predictive models [15]-[21].

Some studies have utilized students' academic achievements in previous courses to predict their performance in upcoming courses. Traditional ML techniques such as regression analysis, decision trees, and support vector machines have been applied for this purpose. For example, the authors in experimented with various regression models to predict students' grades in second-semester courses based on first-semester grades and demographic factors. Similarly, in, a predictive model was developed to forecast students' performance in a mathematics course using their previous grades from school and first-semester university courses. Another study employed classification-based association rule mining to predict students' grades in programming courses based on their English and mathematics scores. However, these studies were based on small datasets, and their models were limited to traditional ML techniques.

To enhance the accuracy of student performance predictions, deep learning methods such as Deep Neural Networks (DNNs) have been explored. In, online datasets from platforms like Kalboard 360 were used for prediction tasks. The authors in applied Convolutional Neural Networks (CNNs) to predict student performance, demonstrating the efficiency of deep learning over traditional models. In, various ML techniques such as Artificial Neural Networks (ANNs), Decision Trees, Random Forest, and ensemble methods were used, showing improvements in model accuracy. Similarly, employed Multi-Layer Perceptron (MLP) along with other ML classifiers like Decision Trees and Naïve Bayes. Some studies, such as, used SMOTE (Synthetic Minority Over-sampling Technique) to handle dataset imbalance and improve classification results.

In, feature importance analysis was performed to determine which factors most influence student performance.

Students' academic performance is a crucial factor in higher education. Several researchers have applied Educational Data Mining (EDM) techniques to predict and evaluate student performance to enhance decision-making processes and improve learning strategies This section discusses prior studies on student performance prediction using different Machine Learning (ML) techniques.

Many studies have used cumulative Grade Point Average (GPA) and course grades as key indicators to predict student success. The authors in conducted experiments using various regression techniques to forecast students' grades in the second semester based on their first-semester grades and demographic characteristics. Similarly, another study developed a model to predict students' grades in a mathematics course using their high school grades and first-semester university grades, employing Support Vector Classifier and Naïve Bayes methods. The study in explored classification-based association rule mining to predict programming course grades using students' past performance in courses such as English and Mathematics.

However, most previous studies used small datasets, typically ranging from 200 to 600 records, which limited their generalizability. Additionally, many relied on traditional ML techniques without exploring deep learning approaches. To address these limitations, recent studies have experimented with deep learning models for student performance prediction. The study in applied Convolutional Neural Networks (CNNs) to analyze student data, demonstrating that deep learning models can outperform traditional ML models in terms of accuracy. Another study used ensemble learning techniques such as Decision Trees, Random Forest, and Boosting, coupled with Genetic Algorithms for feature selection, to enhance predictive performance.

Educational Data Mining (EDM) applies data-driven techniques to analyze student performance, identifying critical patterns that impact academic success. Machine learning models, such as Random Forest and XGBoost, are widely used in EDM for their ability to handle large datasets and extract meaningful insights.[2][4] These models excel in predicting outcomes based on diverse variables, such as admission scores, demographics, and prior academic records. By leveraging EDM with machine learning, educators can implement data-informed interventions to enhance student outcomes and optimize resource allocation.

This approach involves training models using labeled datasets, where both input features (e.g., student demographics, exam scores) and the corresponding output labels (e.g., pass or fail, grade predictions) are known. Algorithms like Random Forest, XGBoost, and SVM are widely used in EDM for predicting student performance, as they learn from the provided labeled data to make accurate predictions about unseen data. These methods are highly effective in classification and regression tasks, where the objective is to predict a specific outcome.

Unlike supervised learning, unsupervised learning works with data that has no labeled outcomes. It focuses on discovering hidden patterns or structures within the data. In the context of EDM, techniques like Clustering (e.g., K-means) or Dimensionality Reduction (e.g., t-SNE) are used to identify groups of students with similar academic performance or to uncover patterns in large datasets without predefined categories. While unsupervised learning may not provide explicit predictions, it is useful for exploratory data analysis and feature extraction.[6][8]

3. SYSTEM ANALYSIS

3.1 Existing System

Traditional approaches to predicting student academic performance have relied on statistical and machine learning (ML) techniques such as Decision Trees (DT), Logistic Regression (LR), Support Vector Classifier (SVC), K-Nearest Neighbors (KNN), and Random Forest (RF).

These models analyze students' historical academic records, including grades and demographic features, to forecast their success in upcoming courses. However, a major limitation of these existing systems is their struggle with imbalanced datasets, where models tend to favor the majority class, leading to unreliable predictions.

Many existing studies on student performance prediction have focused on using cumulative GPA and past course grades as primary indicators of future success. Traditional ML models such as Decision Trees, Naïve Bayes, and Support Vector Machines have been applied, but these approaches often struggle to provide actionable insights for student improvement. Moreover, conventional methods do not consider dynamic learning behaviors, which can be crucial in identifying struggling students at an early stage.

While some models have integrated adaptive learning metrics, they still face limitations in accurately predicting performance due to reliance on static datasets. As a result, more sophisticated methods, such as regression-based models and deep learning techniques, have been proposed to enhance predictive accuracy and improve the effectiveness of academic support systems.

Traditional student performance prediction models mainly use machine learning algorithms like Decision Trees, Logistic Regression, and Support Vector Machines, which rely heavily on historical student data such as past grades, attendance, and demographic information. These systems help identify students at risk but often fail to provide real-time feedback, making early intervention difficult. Additionally, most models do not address the issue of dataset imbalance, which results in biased predictions that favor high-performing students over those who may require academic support.

Previous studies have explored ensemble methods like Random Forest and Gradient Boosting to improve accuracy, but these approaches lack the capability to process complex patterns in educational data. The growing need for more advanced predictive systems has led to the exploration of deep learning models, which offer better generalization and improved early detection of at-risk students.

Existing systems for student performance prediction rely heavily on traditional machine learning techniques such as Decision Trees, Support Vector Machines (SVM), Naïve Bayes, and Logistic Regression. These models primarily use static academic records, including past grades, attendance, and demographic details, to predict student performance.

However, they often fail to capture individual learning patterns and struggle with high-dimensional discrete data, leading to inaccurate or biased predictions. In early warning systems, most existing approaches focus on subjective student factors like grades and family background while neglecting external influences such as course difficulty, instructional methods, and credit distribution.

Moreover, many models have not adapted to the impact of hybrid learning environments, especially after the disruptions caused by the COVID-19 pandemic. The lack of integrated clustering mechanisms and deep learning techniques further limits the effectiveness of these models in identifying students at risk of academic failure. Consequently, traditional systems exhibit lower predictive accuracy and limited adaptability to real-world educational challenges.

3.2 Disadvantages of the Existing System

Limited Feature Consideration: Most traditional models rely on static student data (e.g., past grades, demographics) while ignoring behavioural, psychological, and contextual factors.

Imbalanced Datasets: Many student performance datasets are imbalanced, leading to biased predictions that Favor majority-class samples while neglecting minority-class students.

Lack of Personalization: Existing models do not adapt to individual learning styles and fail to provide personalized academic recommendations.

Inability to Handle Discrete Data Efficiently: Conventional machine learning techniques struggle with high-dimensional, discrete student performance data.

Neglect of External Influences: Most models focus on student-related factors while ignoring course difficulty, instructor effectiveness, and external disruptions (e.g., COVID-19).

Poor Adaptability to Hybrid Learning: With the rise of online and blended learning environments, existing models are ineffective at adjusting to new educational paradigms.

Overfitting Issues: Some models, particularly deep learning-based ones, tend to overfit due to limited training data and lack of generalization across different institutions.

Low Interpretability: Many black-box machine learning models provide accurate predictions but lack explainability, making it difficult for educators to understand the reasons behind student performance outcomes.

3.3 Proposed System

The project aims to predict student performance using Machine Learning and Deep Learning models, improving early detection of at-risk students. It analyzes academic and socio-demographic factors using techniques like t-SNE and feature engineering. Models such as Random Forest, XGBoost, CNN, and LSTM enhance prediction accuracy. The system benefits educational institutions by enabling data-driven decision-making and personalized learning interventions.

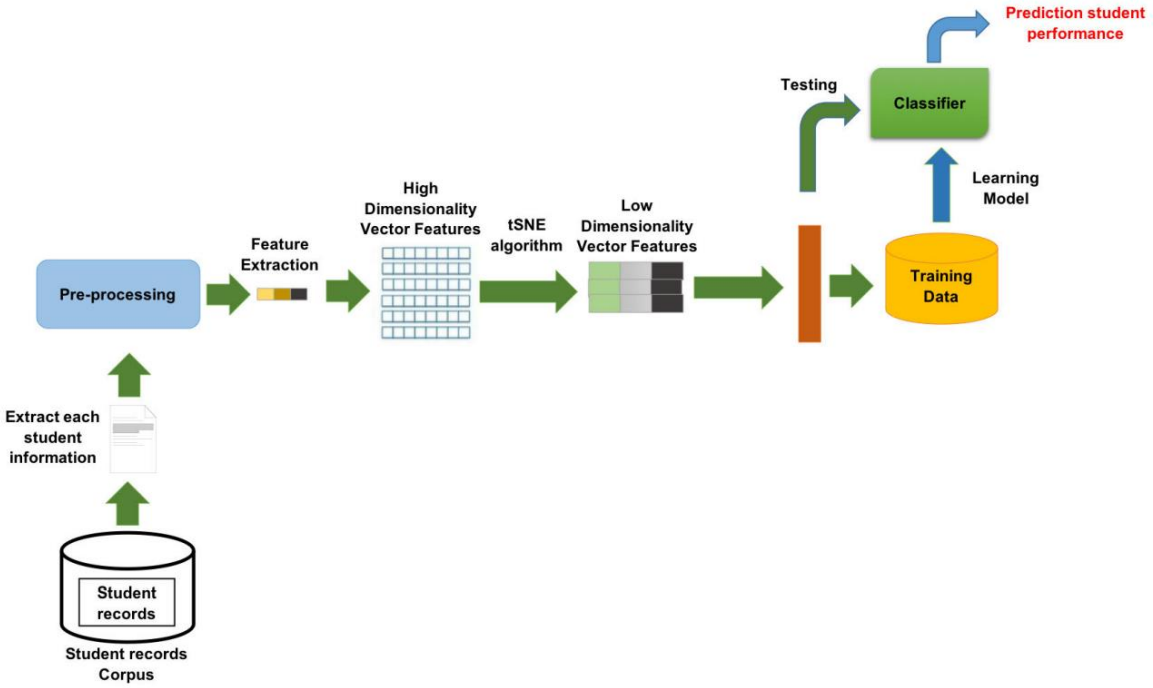


Fig 3.3 Flowchart

The flowchart represents a student performance prediction system utilizing machine learning and dimensionality reduction techniques. It begins with a student records database, where relevant academic and demographic information is extracted for further analysis. The data undergoes pre-processing, including cleaning, normalization, and handling of missing values, ensuring high-quality input for the model. Next, feature extraction is performed, converting student attributes into high-dimensional vector representations.

1. **Data Collection**:

   The first step involves collecting data from various sources, such as student academic

records, socio-demographic details, and performance in past courses. The dataset includes features like gender, admission scores, family background, and previous academic performance.

2. **Data Preprocessing**:

   Once the data is collected, it undergoes a thorough preprocessing phase. This includes handling missing values, normalizing numerical data, and encoding categorical variables. Missing values are handled using imputation techniques, such as filling with mean, median, or mode, while categorical data is encoded using techniques like one-hot encoding.

3. **Feature Extraction**:

   In this step, key features influencing student performance are extracted and selected based on their relevance. Feature engineering involves creating new variables from existing data, such as combining scores from various subjects or considering socio-economic factors that could impact academic performance. Dimensionality reduction techniques like t-SNE (t-distributed Stochastic Neighbour Embedding) are used to reduce the complexity of high-dimensional data while preserving important information.

4. **Model Building**:

   Various machine learning and deep learning models are employed to predict student performance:

   - **Machine Learning Models**: Random Forest, XGBoost, and Support Vector Machine (SVM) are trained on the dataset. These models are effective in handling complex data and identifying key patterns associated with student success.
   - **Deep Learning Models**: Convolutional Neural Networks (CNN) and Long Short-Term Memory (LSTM) networks are also used to model academic performance. CNNs are particularly useful for recognizing patterns in the data, while LSTMs excel at handling sequential data, such as semester-wise performance or trends over time.

5. **Model Training and Evaluation**:

   The models are trained using a training dataset, and their performance is evaluated using various metrics, including accuracy, precision, recall, and F1-score. These metrics provide a comprehensive view of how well the models are predicting student performance. A confusion matrix is used to visualize the performance of the classifiers and identify areas for improvement.

6. **Classification**:

   The final step involves classifying students into different performance categories (e.g., high performers, low performers, and at-risk students) based on the predicted scores. This classification helps in early identification of students who may need academic support or intervention.[7]

7. **Results and Visualization**:

   The results are analysed and presented using various visualization techniques such as bar charts, graphs, and heatmaps. The performance of different models is compared to identify the most accurate approach for predicting student success.

8. **System Implementation and Integration**:

   The methodology is implemented in a user-friendly system that allows educators to input student data and receive performance predictions. The system is designed to be scalable and can be integrated with existing academic management systems to provide real-time performance insights.

   By integrating machine learning and deep learning models, this proposed methodology aims to enhance the accuracy of academic performance predictions, leading to timely interventions and better resource allocation to improve student outcomes.

3.3.1 Data Collection and Pre-processing

Data preprocessing is a foundational step that transforms raw data into a format suitable for analysis, ensuring that the machine learning and deep learning models produce accurate and reliable results. The first step involves data cleaning, where missing values are handled through imputation or by removing rows with insufficient data, depending on the extent of the missing information. Normalization and scaling are then applied to numerical features to prevent any one feature from dominating due to differences in scale, ensuring that all features contribute equally to model performance.[2][3][4] Categorical data is converted into numerical form through techniques like one-hot encoding or label encoding, which helps the models interpret and process the data.

Additionally, outlier detection is performed to identify extreme values that could distort the results, with the outliers either being removed or adjusted to maintain data integrity. Feature selection is conducted to identify the most relevant attributes that contribute to predicting the target variable, while feature extraction techniques like principal component analysis (PCA) or t-SNE reduce dimensionality, making the dataset more manageable while retaining critical

information. Data augmentation is sometimes used, especially in deep learning, to artificially expand the dataset and improve model robustness by generating new data points based on existing ones. Finally, data is split into training and testing sets to evaluate model performance accurately. These preprocessing steps ensure that the data is well-prepared, reducing noise and enhancing the models' ability to learn meaningful patterns, which ultimately improves the predictive accuracy of student performance models.

3.3.2 Feature Extraction:

Feature extraction is a critical step in the data preprocessing pipeline, particularly when working with machine learning and deep learning models. It involves transforming raw data into a set of meaningful variables (features) that can improve model performance and prediction accuracy. The objective of feature extraction is to identify the most relevant aspects of the data, reducing complexity while retaining valuable information.[2]

In this study, feature extraction begins by selecting key variables that directly influence student performance, such as exam scores, attendance records, socio-demographic information (e.g., parental education level, gender), and other academic indicators. Techniques like Principal Component Analysis (PCA) or t-SNE (t-distributed Stochastic Neighbour Embedding) can be applied to reduce high-dimensional data into lower dimensions, making it easier for models to process without losing significant information.

Additionally, new features may be created through feature engineering, such as combining existing features (e.g., average scores from multiple subjects or a weighted score combining entrance exam and midterm results) to better capture the complexity of student performance. For example, socio-economic status and prior academic achievements might be combined to form a new feature that better predicts future success.[7]

The goal of feature extraction is to improve the accuracy and efficiency of the models by focusing on the most relevant data points and removing redundant or irrelevant features. This ensures that the models are better equipped to identify patterns and make accurate predictions. Feature extraction is particularly important when dealing with large and complex datasets, as it enables the models to work with the most informative variables while minimizing the risk of overfitting.[15]

3.3.3 Dimensionality Reduction Techniques (t-SNE)

Dimensionality reduction is a technique used to reduce the number of features or variables in a dataset while retaining as much of the relevant information as possible. This is particularly useful when dealing with high-dimensional data, as it simplifies the dataset and makes it easier to visualize, interpret, and process. One of the most commonly used dimensionality reduction techniques is t-SNE (t-distributed Stochastic Neighbour Embedding).[1][3]

t-SNE is a non-linear dimensionality reduction technique that is particularly effective for visualizing high-dimensional data in two or three dimensions. It works by minimizing the divergence between probability distributions over pairwise similarities, thus grouping similar data points together and ensuring that dissimilar points are placed farther apart in the reduced dimensional space. The key advantage of t-SNE is its ability to preserve local relationships in the data, which is crucial when visualizing complex, high-dimensional datasets.

In the context of predicting student performance, t-SNE is used to reduce the complexity of data such as multiple academic scores, socio-demographic features, and other performance indicators. By transforming this high-dimensional data into two or three dimensions, t-SNE helps identify patterns and relationships between students that are not immediately obvious in the original high-dimensional space. This makes it easier for researchers and educators to visualize clusters of students with similar academic profiles, such as high performers, low performers, or students at risk.

While t-SNE is highly effective for visualization, it is important to note that it is mainly used for exploratory data analysis rather than direct feature selection for predictive modelling. t-SNE helps uncover hidden structures and patterns in the data, which can then inform the feature selection and model building process.

Overall, t-SNE is a powerful tool for dimensionality reduction, especially when dealing with complex datasets that need to be simplified for better visualization and interpretation.

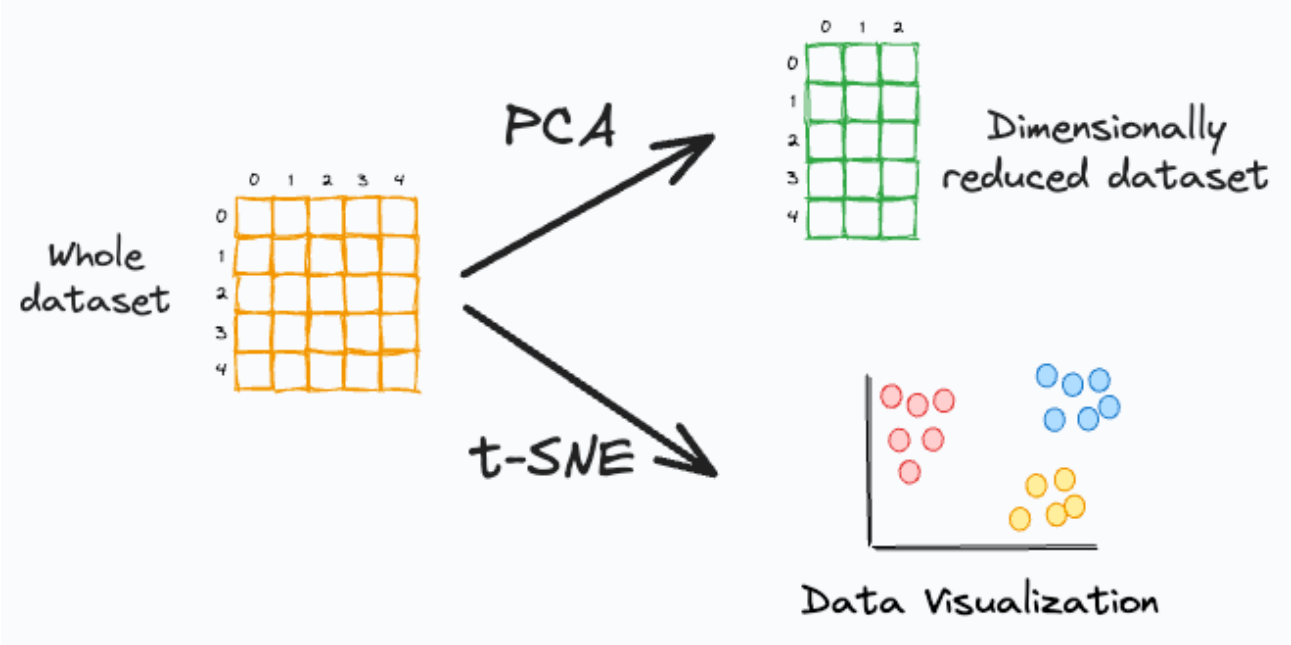


Fig 3.3.3 : T-SNE Algorithm

3.3.4 Machine Learning Algorithms:

In this project, several machine learning (ML) algorithms are employed to predict student performance and evaluate their effectiveness. These algorithms are chosen for their ability to handle large datasets, manage complex relationships, and provide accurate predictions. The key machine learning algorithms used are:

1. **Random Forest**:

Random Forest is an ensemble learning algorithm that builds multiple decision trees and combines their outputs to improve prediction accuracy.[7] It is particularly effective for handling high-dimensional datasets with mixed data types (both numerical and categorical). Random Forest is known for its robustness, ability to avoid overfitting, and high accuracy in classification tasks, making it ideal for predicting student performance based on various features like academic scores, demographic data, and socio-economic factors.

2.**Logistic Regression:**

Logistic Regression is a statistical model used for binary classification problems, such as predicting whether a student will pass or fail. It works by estimating the probability of an event occurring, based on input features. In this project, logistic regression is used to model the

likelihood of different student outcomes (e.g., success or failure) and can be an effective baseline model due to its simplicity and interpretability.[8]

3. **Support Vector Machine (SVM)**:

Support Vector Machine is a powerful classification algorithm that works by finding the optimal hyperplane that best separates data points into different classes. SVM is effective in high-dimensional spaces and works well even when the data is not linearly separable. In the context of the project, SVM is used to classify students based on academic performance and predict future outcomes by creating decision boundaries between different categories of student performance.[8]

4. **K-Nearest Neighbors (KNN)**:

KNN is a non-parametric, instance-based learning algorithm that makes predictions based on the similarity between data points. It classifies a student's performance by finding the 'k' nearest neighbors (students with similar features) and assigning the most common outcome among them. KNN is simple and intuitive but can be computationally expensive with large datasets. It is useful for predicting outcomes based on the closeness of a student's academic and socio-demographic features to others in the dataset.

5. **XGBoost (Extreme Gradient Boosting)**:

XGBoost is a high-performance implementation of gradient boosting, an ensemble technique that combines the predictions of multiple weak models (typically decision trees) to create a stronger predictive model. XGBoost is known for its scalability, speed, and accuracy, particularly in handling large datasets. It is effective in improving prediction accuracy by focusing on minimizing the residual errors from previous models, making it ideal for complex datasets like those used in student performance prediction.[10]

3.3.5 Deep Learning Algorithms

In addition to traditional machine learning algorithms, this project also employs deep learning models, specifically Convolutional Neural Networks (CNN) and Long Short-Term Memory (LSTM) networks, to predict student performance.[7] These models are used for their ability to capture complex patterns and relationships in data, especially when dealing with large and high-dimensional datasets.

1. Convolutional Neural Networks

Convolutional Neural Networks (CNNs) are primarily known for their effectiveness in image

recognition tasks, but their application extends to structured data as well. In this project, CNNs are utilized to analyse and extract relevant features from student performance data. CNNs work by applying convolutional layers to the data, which automatically learn spatial hierarchies and detect patterns such as trends in scores or socio-demographic variables that might not be obvious at first glance. This feature makes CNNs highly effective at identifying relationships in large, complex datasets and allows them to handle non-linearities better than traditional machine learning models.

CNNs excel in automatically learning relevant features from raw data without the need for manual feature extraction. By learning hierarchical patterns in the data, CNNs can improve the model's ability to classify students into different performance categories (e.g., high achievers, at-risk students) based on their academic profiles.[6]

CNNs excel in automatically learning relevant features from raw data without the need for manual feature extraction. By learning hierarchical patterns in the data, CNNs can improve the model's ability to classify students into different performance categories (e.g., high achievers, at-risk students) based on their academic profiles.

2**.Long Short-Term Memory (LSTM)**:

networks are a type of Recurrent Neural Network (RNN) designed to handle sequential data, making them ideal for problems where time-based patterns are important. In the context of this project, LSTMs are particularly useful for predicting student performance across multiple semesters or time periods, where previous academic performance influences future success. For example, an LSTM can process historical performance data, such as grades across different courses or semesters, and make predictions about future academic outcomes.[8]

LSTM networks are capable of learning long-term dependencies in time-series data by addressing the vanishing gradient problem typical in traditional RNNs.

This makes LSTMs highly suitable for applications where trends and patterns evolve over time, such as tracking student progress and predicting their academic performance.

The sequence-based nature of LSTMs allows them to model the temporal relationships between academic records, which can significantly improve predictions of student success by taking into account the history of their academic achievements and challenges.

Comparison and Integration of CNN and LSTM in the Project

- **CNNs** are used to identify local patterns in student data, especially useful in recognizing complex relationships between academic scores, socio-demographic features, and other

attributes.

- **LSTMs** are used to capture temporal patterns in students' performance, analysing their progress over time and predicting future performance based on past behaviour.

Both models enhance the overall prediction capabilities of the project, with CNNs excelling in spatial relationships and LSTMs focusing on sequential patterns. Together, they provide a comprehensive approach to student performance prediction, allowing for both static (individual data points) and dynamic (time-based data) analysis.[10]

- A true positive (tp) is a result where the model predicts the positive class correctly. Similarly, a true negative (tn) is an outcome where the model correctly predicts the negative class.
- A false positive (fp) is an outcome where the model incorrectly predicts the positive class. Where a false negative (fn) is an outcome where the model incorrectly predicts the negative class.

3.3.6 Performance Metrics (Accuracy, precision, Recall, F1-Score)

Accuracy

Accuracy measures the proportion of correctly predicted instances (true positives and true negatives) out of the total predictions.

Accuracy=TP+TN/TP+FP+TN+FN

Precision

Precision evaluates the proportion of true positive predictions among all positive predictions made by the model. It highlights how precise or accurate the model is in predicting positive cases.

Precision=TP/TP+FP

Recall

Recall measures the proportion of actual positive cases that the model successfully identified. It is important when the focus is on minimizing false negatives.

Recall=TP/TP+FN

F1 Score

The F1 score is the harmonic mean of precision and recall. It provides a balanced metric that accounts for both false positives and false negatives, especially useful when dealing

with imbalanced datasets

F1-Score=2·(Precision Recall/Precision Recall)

3.4 Feasibility Study

1. Economical Feasibility

The economical feasibility of the project determines whether the implementation is cost-effective. The project uses Google Colab Pro for model training and development, which is a cloud-based platform that eliminates the need for high-end local hardware.

Estimated Costs:

- Google Colab Pro Subscription: ₹1,093/month ($9.99/month)
- Flask Development (Frontend & Backend Integration): No additional cost as Flask is open-source.

Since Google Colab Pro provides GPU/TPU access, the costs remain lower compared to purchasing high-end GPUs for local training. Overall, the project is economically viable due to its reliance on free and affordable cloud-based tools.

2. Technical Feasibility

Technical feasibility evaluates the technologies and programming languages used to implement the project.

Technologies & Tools Used:

- Programming Languages: Python (Machine Learning & Deep Learning models)
- Machine Learning Models: Random Forest, XGBoost, CNN, LSTM, SVM, Logistic Regression
- Development Framework: Flask (Used for the frontend and backend integration)
- Cloud Computing: Google Colab Pro (for model training and execution)

4. SYSTEM REQUIREMENTS

A machine learning project requires specific hardware and software configurations to ensure efficient data processing, model training, and deployment. Below is a detailed explanation of each requirement: Key Features of Google Colab

4.1 SOFTWARE REQUIREMENTS

1.Operating System: Windows 11, 64-bit

- A modern OS with enhanced security, efficiency, compatibility with ML tools.
- The 64-bit architecture has high-performance computing, large memory usage.

2.Coding Language: Python

• Python is widely used for ML due to its rich ecosystem of libraries (TensorFlow,Scikit-learn, Pandas, NumPy).

• It provides easy syntax, rapid development, and extensive community support.

3.Python Distribution: Google Colab Pro, Flask

• Google Colab Pro: A cloud-based Jupyter notebook environment with GPU/TPU support for faster model training.

• Flask: A lightweight web framework used to deploy and serve ML models as APIs.

4.Browser: Any Latest Browser (e.g., Chrome)

• A modern web browser ensures compatibility with Google Colab, Flask applications, and cloud-based tools.

• Google Chrome is preferred for its performance, developer tools, and security features.

4.2 Hardware Requirements

1.System Type: Intel® Core™ i5-7500U CPU @ 2.40GHz

The Intel Core i5-7500U is a dual-core processor suitable for ML tasks such as data preprocessing and small-scale model training.

It operates at 2.40 GHz, providing a balance of speed and power efficiency.

2.Cache Memory: 4MB (Megabyte)

• Cache memory stores frequently accessed data, reducing latency in computations.

• A 4MB cache helps in improving data retrieval speed, benefiting real-time processing.

3.RAM: Minimum 8GB (Gigabyte)

• RAM (Random Access Memory) allows temporary data storage for active processes.

• 8GB RAM is the minimum requirement to handle ML libraries, datasets, and computations without excessive lag.

4.Hard Disk: 229GB

• Storage is essential for datasets, trained models, and software dependencies.

• A 229GB hard disk provides sufficient space for project files, experiment logs.

5.Computer Engine: T4 GPU

• NVIDIA T4 GPU is designed for AI and deep learning applications.

• It accelerates training, inference tasks, reducing process time significantly.

4.3 Software and it's Description

1.Python (3.x):

Python is an interpreted, high-level programming language widely used for web development, data science, and automation. It provides a simple syntax and an extensive set of libraries, making it a preferred choice for Flask-based applications. The Flask framework runs on Python, and essential libraries like Flask-SQLAlchemy and Flask-Bcrypt require it. To ensure compatibility, install Python 3.x . After installation, verify it using python --version. Using a virtual environment (venv) is recommended for dependency management. Python's built-in SQLite database also eliminates the need for additional database installations in small-scale applications.

2.pip (Python Package Manager):

pip is the standard package manager for Python, enabling easy installation, upgrading, and removal of libraries. It ensures that dependencies such as Flask, Flask-SQLAlchemy, Flask-Bcrypt, NumPy, and Pickle5 can be installed efficiently. It is included with Python installations by default but can be updated using pip install --upgrade pip. The requirements.txt file allows bulk installations, making pip an essential tool for managing Flask applications. Running pip list displays installed packages, while pip freeze > requirements.txt saves dependencies for future use. Without pip, manually managing dependencies would be time-consuming and error-prone.

3.Flask:

Flask is a lightweight web framework for Python that enables developers to create web

applications quickly and efficiently. Unlike Django, Flask is minimalistic and follows a modular design, allowing developers to integrate only necessary components. It provides built-in support for routing, request handling, and templates, making it ideal for applications like user authentication and machine learning-based predictions. Flask's simplicity allows developers to build scalable applications while maintaining control over database configurations and security measures. With a growing ecosystem and extensive documentation, Flask is an excellent choice for both beginners and experienced developers building web-based applications.

4.Flask-SQLAlchemy:

Flask-SQLAlchemy is an Object-Relational Mapping (ORM) tool that simplifies database interactions in Flask applications. Instead of writing raw SQL queries, developers can define Python classes to represent database tables, making code more readable and maintainable. It supports multiple databases, including SQLite, MySQL, and PostgreSQL. Using Flask-SQLAlchemy, developers can perform CRUD (Create, Read, Update, Delete) operations effortlessly. It also provides features like connection pooling and session management, enhancing database efficiency. Since SQLite is the default database for this project, Flask-SQLAlchemy is necessary for managing user authentication and storing machine learning-related data securely.

5.Flask-Bcrypt:

Flask-Bcrypt is a Flask extension that provides password hashing capabilities using the Bcrypt hashing algorithm. Since storing plain-text passwords is a security risk, Flask-Bcrypt ensures that passwords are securely hashed before storing them in the database. It automatically adds salt to prevent dictionary attacks and brute-force attacks. By using bcrypt.generate_password_hash(password), passwords can be hashed securely, and authentication is handled using bcrypt.check_password_hash(stored_hash, entered_password). This library enhances application security, ensuring that user credentials remain protected even in case of database leaks. It is a crucial component for implementing user authentication in Flask applications.

6. NumPy:

NumPy is a powerful library for numerical computing in Python, widely used for handling large datasets, performing mathematical operations, and working with multi-dimensional arrays. In this Flask project, NumPy is essential for pre-processing input data before making predictions with the machine learning model. It ensures that the user's input values are

converted into NumPy arrays before being passed to the model.predict() function. This improves efficiency and ensures compatibility with machine learning models. NumPy's optimized array operations significantly boost performance, making it an indispensable tool for applications involving numerical data processing and machine learning predictions.

7. Pickle5:

Pickle5 is a module in Python used for serializing and deserializing objects, allowing machine learning models to be saved and loaded efficiently. In this project, the trained model is stored in model.pkl, which is loaded using pickle.load(open('model.pkl', 'rb')). This eliminates the need to retrain models every time the application starts, significantly reducing processing time. Pickle5 ensures seamless model integration into Flask applications, allowing real-time predictions from previously trained models. Since different Python versions affect pickled files, using Pickle5 ensures better compatibility, making it crucial for deploying machine learning models in web applications.

8. SQLite:

SQLite is a lightweight, serverless relational database that is built into Python, making it ideal for small-scale applications. It does not require additional installation, reducing setup complexity. SQLite stores data in a single .db file, allowing for quick and easy access. In this Flask project, SQLite is used to manage user accounts, storing credentials securely with Flask-SQLAlchemy. It supports standard SQL queries, making it a convenient choice for handling authentication. Since it is embedded and does not require a separate server process, SQLite is well-suited for prototyping and small web applications with low to moderate database needs.

5.SYSTEM DESIGN

5.1 System Architecture

Dataset Description

The" Student Academic Performance" dataset in table 1 gives an intensive summary of all the variables influencing students' educational success. Academic performance signs comprise choice indices for post-secondary training establishments as well as rankings in more than a few subjects, together with well-known, FEP, and English, alongside percentile, decile, and quartile rankings. The dataset provides insights into the effect of own family records and socioeconomic role on academic outcomes and enables thorough take a look at and prediction of student overall performance.

Table :5.1 Dataset Description

Variable	Description
COD S11	Identification code for each student
GENDER	Gender of the student
EDU FATHER	Education level of the father
EDU MOTHER	Education level of the mother
PEOPLE HOUSE	Number of people living in the household
MAT S11	Mathematics score for S11
CR S11	Critical reading score for S11
CC S11	Civic and citizenship education score for S11
BIO S11	Biology score for S11
ENG S11	English score for S11
Cod Spro	Program code for higher education
UNIVERSITY	Name of the university attended
ACADEMIC PROGRAM	Name of the academic program
QR PRO	Quantitative reasoning score for higher education
CR PRO	Critical reading score for higher education

Preprocessing:

Preprocessing is a crucial step in data analysis that ensures the dataset is clean, consistent, and suitable for machine learning models. The key steps include data cleaning, normalization, handling class imbalance, and dataset splitting.

Explanation of Each Step

1. Data Cleaning:

- Missing values are handled by either deleting rows/columns with excessive missing data or imputing them using statistical methods (mean, median, mode).
- Errors and inconsistencies in data entries are corrected to improve data quality.

2. Normalization and Scaling:

- To ensure consistency, numerical features like admission scores and course performance measures are transformed.
- Techniques such as Z-score normalization (scaling data to have a mean of 0 and a standard deviation of 1) and Min-Max scaling (scaling values between 0 and 1) are applied.

3. Handling Class Imbalance:

- If the dataset has an uneven distribution of classes (e.g., more high-performing students than low-performing ones), resampling methods are used.
- Oversampling (increasing samples of the minority class) and under sampling (reducing samples of the majority class) help balance the dataset.

4. Dataset Splitting:

- The dataset is divided into training and testing sets to evaluate model performance.
- A common split ratio is 80/20 or 70/30, where most data is used for training, and a smaller portion is kept for testing to assess model accuracy.

Models:

1.Random Forest (RF): Random Forest is an ensemble learning algorithm that combines multiple decision trees to improve accuracy and reduce overfitting. It works by creating multiple subsets of the data, training separate decision trees on them, and then averaging their predictions (for regression) or using majority voting (for classification). This approach enhances model stability and performance, making it robust for handling large datasets and reducing errors caused by individual weak classifiers. Random Forest is widely used for classification and

regression tasks, including student performance prediction, due to its high accuracy and ability to handle missing values and noisy data effectively.

2.XGBoost: XGBoost (Extreme Gradient Boosting) is a powerful and efficient gradient boosting algorithm designed for high-performance machine learning tasks. It builds decision trees sequentially, where each new tree corrects the errors of the previous ones, improving accuracy over time. XGBoost is optimized for speed and efficiency, using techniques like regularization (L1 & L2), parallel processing, and tree pruning to prevent overfitting and enhance predictive performance. It is widely used in classification and regression problems, making it ideal for student performance prediction due to its ability to handle large datasets and complex patterns effectively.

3.Support Vector Machine (SVM): Support Vector Machine (SVM) is a supervised learning algorithm used for classification and regression tasks. It works by finding the optimal hyperplane that best separates different classes in a high-dimensional space. SVM uses support vectors (data points closest to the hyperplane) to maximize the margin between classes, improving generalization. It is effective in handling both linear and non-linear data using kernel functions like linear, polynomial, and radial basis function (RBF). SVM is widely used for student performance prediction due to its ability to handle high-dimensional datasets and avoid overfitting.

4.K-Nearest Neighbours (KNN): K-Nearest Neighbors (KNN) is a simple and intuitive machine learning algorithm used for classification and regression tasks. It works by comparing a new data point to its closest "K" neighbors in the dataset and assigning it the most common class (for classification) or averaging their values (for regression). KNN is a non-parametric and instance-based learning algorithm, meaning it does not assume any underlying data distribution and makes predictions based on stored instances. It is effective for small to medium-sized datasets and is commonly used for student performance prediction due to its ease of implementation and adaptability to different data distributions.

5.Logistic Regression: Logistic Regression is a statistical machine learning algorithm used for binary and multi-class classification tasks. It predicts the probability of an outcome belonging to a particular class using the logistic (sigmoid) function, which maps values between 0 and 1. Unlike linear regression, which predicts continuous values, logistic regression outputs probabilities that can be converted into class labels (e.g., pass/fail). It is widely used for student performance prediction due to its simplicity, interpretability, and efficiency in handling structured datasets with numerical and categorical features.

6.Convolutional Neural Networks (CNN): Convolutional Neural Networks (CNN) are a type of deep learning model primarily designed for image processing and pattern recognition, but they are also effective in analyzing structured data. CNNs consist of multiple layers, including convolutional layers (which detect patterns and features), pooling layers (which reduce dimensionality), and fully connected layers (which make final predictions). In student performance prediction, CNNs can capture complex relationships in academic data by identifying hidden patterns and correlations. Their ability to automatically extract features makes them more powerful than traditional machine learning models, improving prediction accuracy.

7.Long Short-Term Memory (LSTM): Long Short-Term Memory (LSTM) is a type of Recurrent Neural Network (RNN) designed to process and analyze sequential data by retaining important information over long time periods. Unlike standard RNNs, LSTMs use gates (input, forget, and output gates) to control the flow of information, preventing issues like vanishing gradients and improving long-term dependencies. In student performance prediction, LSTMs are useful for analyzing trends over time, such as academic progress across multiple semesters, making them effective for predicting future performance based on historical data.

5.2 Modules

1. Preprocessing Module

This module is responsible for cleaning and preparing the data before feeding it into machine learning models. It includes:

- Data Cleaning: Handling missing values, correcting inconsistencies, and removing duplicate entries.
- Feature Scaling & Normalization: Applying techniques like Z-score normalization or Min-Max scaling to standardize data.
- Class Imbalance Handling: Using oversampling, under sampling, or algorithm-based solutions to ensure balanced datasets.
- Dataset Splitting: Dividing the data into training and testing sets (e.g., 80/20 split) for model evaluation.

2. System Module

Model Training & Development:

- The Random Forest algorithm is used as the final model due to its high accuracy and robustness.

- The dataset is split into training (80%) and testing (20%) sets for evaluation.
- Multiple decision trees are trained on different subsets of the data, and their outputs are aggregated for final prediction.

Feature Selection & Processing:

- Important attributes like academic scores, entrance exam results, and socio-demographic data are used as input features.
- Feature importance is analysed to select the most relevant factors affecting student performance.

Prediction & Classification:

- The trained Random Forest model predicts student success or failure based on input data.
- The majority vote from multiple decision trees determines the final classification.

Performance Evaluation:

- The model is evaluated using metrics like accuracy, precision, recall, and F1-score to measure its effectiveness.
- Cross-validation is used to ensure generalizability and avoid overfitting.

Optimization & Fine-Tuning:

- Hyperparameter tuning (e.g., adjusting the number of trees, max depth) is applied to improve model accuracy.
- The final optimized Random Forest model is deployed for real-time student performance predictions.

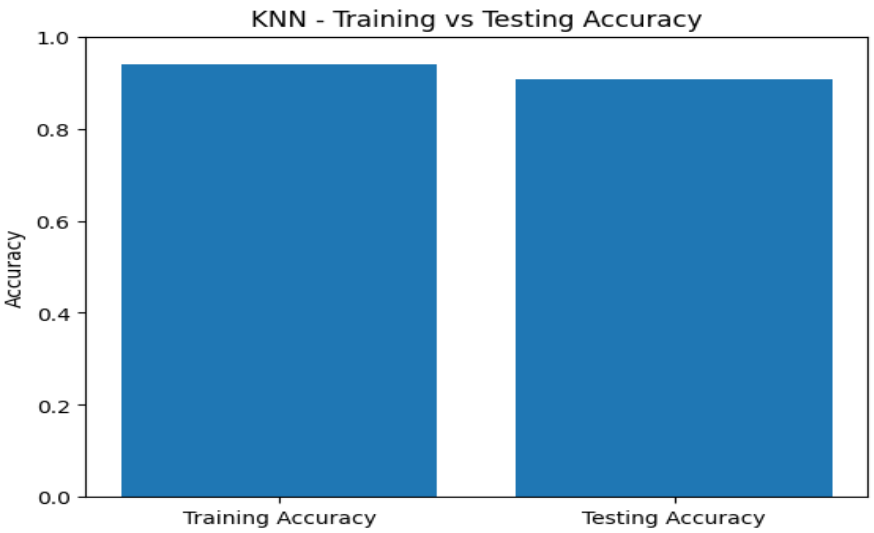


Fig 5.2.1: KNN Training and Testing

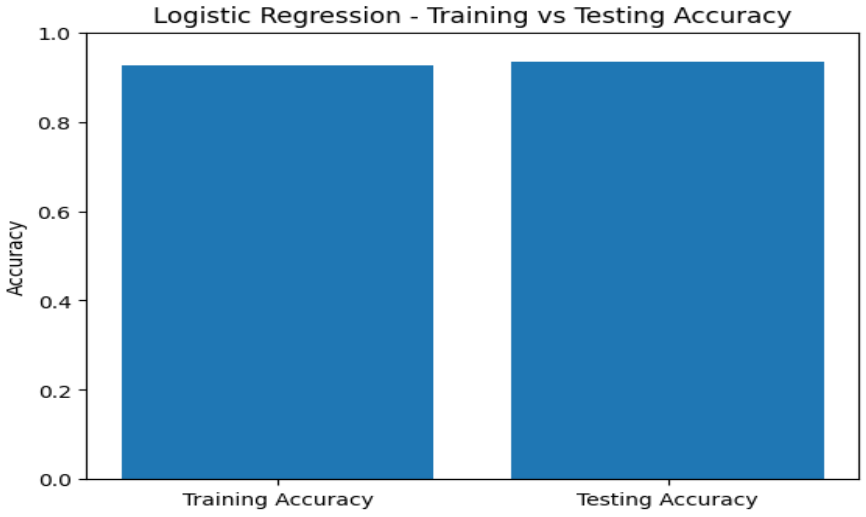


Fig 5.2.2 : Logistic Regression Training vs Testing

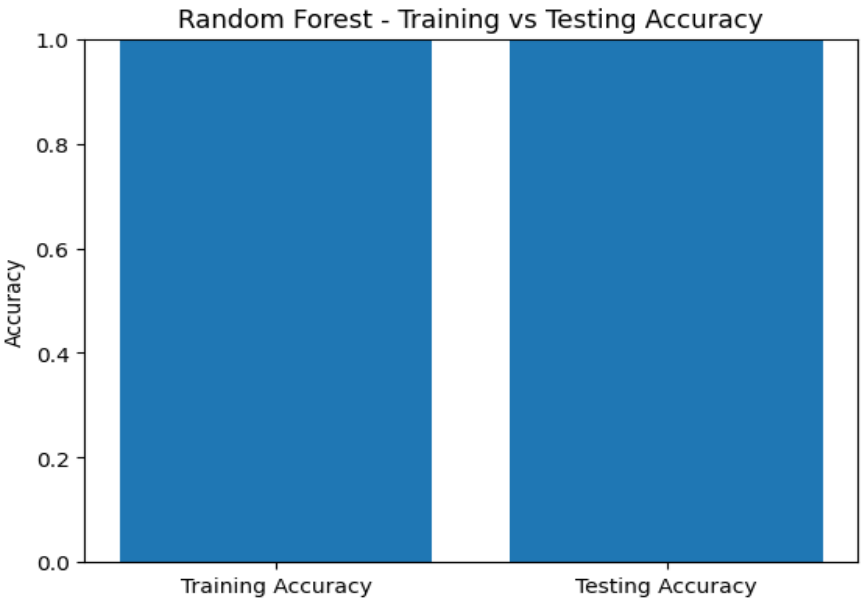


Fig 5.2.3: Random Forest Training vs Testing

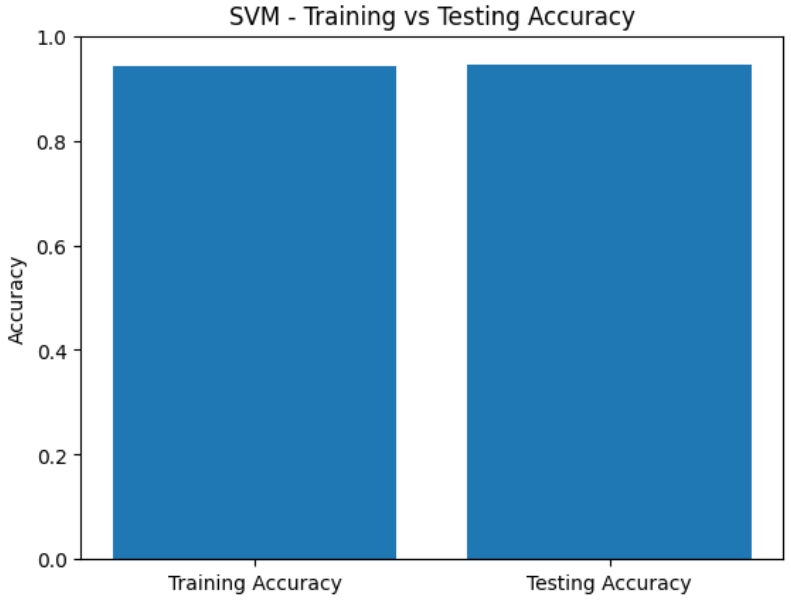


Fig 5.2.4 : SVM Training vs Testing

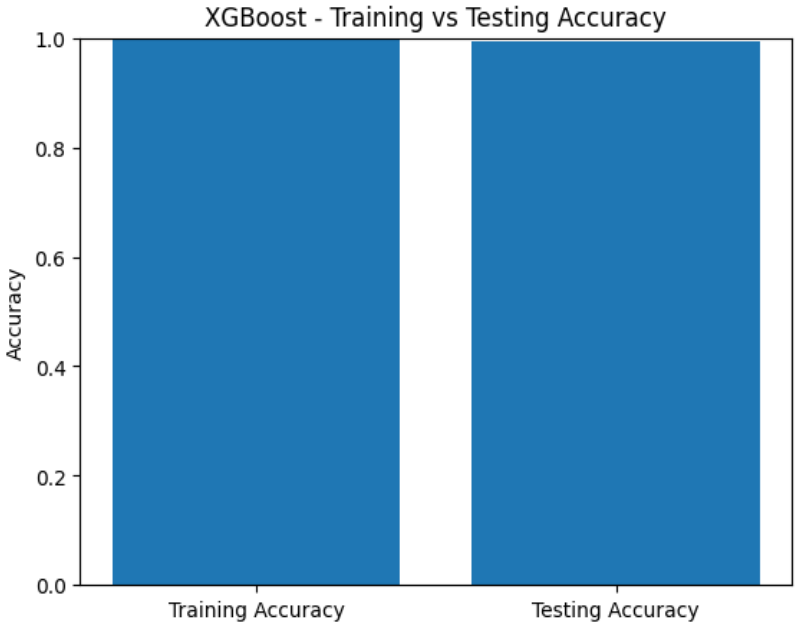


Fig 5.2.5: XGBoost Training vs Testing accuracy

3. User Interface (UI) Module

This module provides an interactive interface for users to input data, view predictions, and analyse results. It includes:

- Data Input Forms: Allows users to enter student details and academic scores.
- Prediction Display: Shows predicted student performance with visual representations (e.g., graphs, tables).
- Analytics Dashboard: Provides insights into overall student performance trends using charts and statistical summaries.
- Model Comparison & Reports: Displays evaluation results of different models for better decision-making.

5.3 UML Diagrams

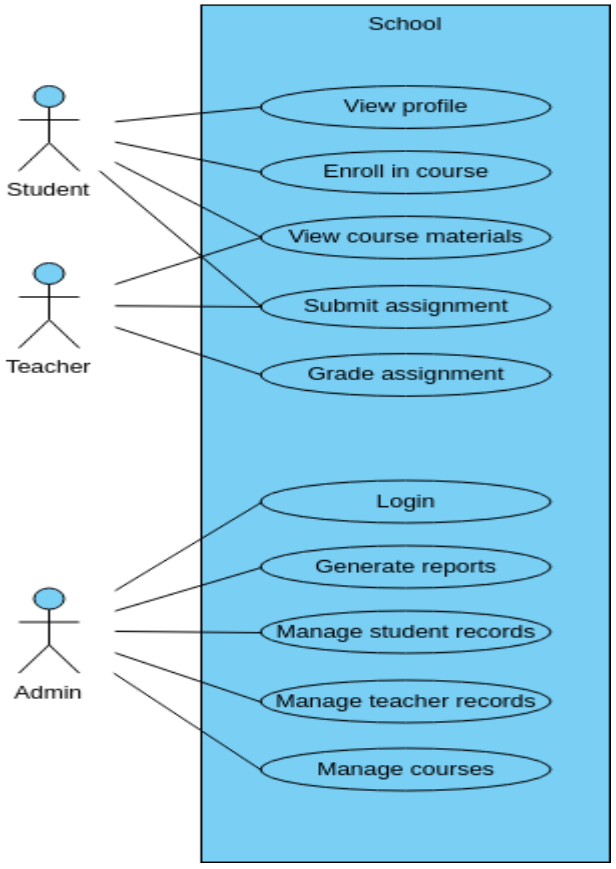


Fig 5.3: UML diagram

6. IMPLEMENTATION

6.1 Model Implementation

Training and testing classification models using various machine learning algorithms like Xgboost, Logistic Regression, SVM, KNN, and Random Forest to predict student performance

```
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble import RandomForestClassifier
from xgboost import XGBClassifier
# Separate features and target
X = df.drop('PERCENTILE', axis=1)
# Convert 'PERCENTILE' to categorical by defining appropriate thresholds or bins
# For example, you can create categories like 'Low', 'Medium', 'High' based on percentile ranges.
y = pd.cut(df['PERCENTILE'], bins=[0, 0.33, 0.66, 1], labels=[0, 1, 2], include_lowest=True)  # Adjust bins and labels as needed , include the lowest value to avoid NaNs
# Handle potential NaN values in the target variable
# Note: Filling NaN values with -1 to distinguish them during model training
y = y.cat.add_categories(-1).fillna(-1) # Fill NaN values with a new category
# Split data into train and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
# Define models
models = {
    "Logistic Regression": LogisticRegression(),
    "SVM": SVC(),
    "KNN": KNeighborsClassifier(),
    "Random Forest": RandomForestClassifier(),
    "XGBoost": XGBClassifier()
}
```

```
# Train and evaluate each model
for name, model in models.items():
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)
    accuracy = accuracy_score(y_test, y_pred)
    print(f"Model: {name}, Accuracy: {accuracy}")
    print(classification_report(y_test, y_pred))
    print(confusion_matrix(y_test, y_pred))
```

6.2 Coding

```
#Read the dataset
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')

# Use pd.read_excel for Excel files
df = pd.read_excel("/content/drive/MyDrive/data_academic_performance.xlsx")
print(df.head())

df.info()
# Handle the missing values

# Check for missing values
missing_values = df.isnull().sum()

# Drop columns with more than 50% missing values
df = df.dropna(thresh=0.5, axis=1)

# Select numerical columns
numerical_cols = df.select_dtypes(include=np.number).columns

# Fill missing values in numerical columns with the mean
df[numerical_cols] = df[numerical_cols].fillna(df[numerical_cols].mean())

# Check for remaining missing values
missing_values = df.isnull().sum()

# Print the updated DataFrame
print(df.head())
# prompt: perform feature engineering
```

```
import pandas as pd
# Create dummy variables for categorical columns
categorical_cols = df.select_dtypes(include='object').columns
df = pd.get_dummies(df, columns=categorical_cols)

# Feature scaling
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
df[numerical_cols] = scaler.fit_transform(df[numerical_cols])


# Feature selection
from sklearn.feature_selection import SelectKBest, f_regression
selector = SelectKBest(f_regression, k=10)
df = pd.DataFrame(selector.fit_transform(df, df['PERCENTILE']),
columns=df.columns[selector.get_support()])

# Print the final DataFrame
print(df.head())
# prompt: perform normalization

import pandas as pd
from sklearn.preprocessing import MinMaxScaler

# Normalize the data
scaler = MinMaxScaler()
df = pd.DataFrame(scaler.fit_transform(df), columns=df.columns)

# Print the normalized DataFrame
print(df.head())
# prompt: use t-sne algorithm for dimensionality reduction

import pandas as pd
from sklearn.manifold import TSNE

# Initialize the TSNE model with desired parameters
tsne_model = TSNE(n_components=2, learning_rate='auto',
                  init='random', perplexity=3, random_state=0)

# Reduce dimensionality of the data
df_tsne = tsne_model.fit_transform(df)

# Convert the results to a DataFrame
df_tsne = pd.DataFrame(df_tsne, columns=['Dimension 1', 'Dimension 2'])

# Print the transformed data
print(df_tsne.head())
# prompt: perform feature selection
```

```
import pandas as pd
from sklearn.feature_selection import SelectKBest, f_regression

# Select the top 10 features based on F-regression
selector = SelectKBest(f_regression, k=10)
df = pd.DataFrame(selector.fit_transform(df, df['PERCENTILE']),
columns=df.columns[selector.get_support()])
# Print the selected features
print(df.head())
# prompt: Training and testing classification models using various machine learning
algorithms like Xgboost, Logistic Regression, SVM, KNN, and Random Forest to predict
student performance

from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble import RandomForestClassifier
from xgboost import XGBClassifier

# Separate features and target
X = df.drop('PERCENTILE', axis=1)
# Convert 'PERCENTILE' to categorical by defining appropriate thresholds or bins
# For example, you can create categories like 'Low', 'Medium', 'High' based on percentile
ranges.
y = pd.cut(df['PERCENTILE'], bins=[0, 0.33, 0.66, 1], labels=[0, 1, 2],
include_lowest=True)  # Adjust bins and labels as needed , include the lowest value to avoid
NaNs

# Handle potential NaN values in the target variable
# Note: Filling NaN values with -1 to distinguish them during model training
y = y.cat.add_categories(-1).fillna(-1) # Fill NaN values with a new category

# Split data into train and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Define models
models = {
    "Logistic Regression": LogisticRegression(),
    "SVM": SVC(),
    "KNN": KNeighborsClassifier(),
    "Random Forest": RandomForestClassifier(),
    "XGBoost": XGBClassifier()
}

# Train and evaluate each model
```

```
for name, model in models.items():
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)
    accuracy = accuracy_score(y_test, y_pred)
    print(f"Model: {name}, Accuracy: {accuracy}")
    print(classification_report(y_test, y_pred))
    print(confusion_matrix(y_test, y_pred))
import matplotlib.pyplot as plt
# Train and evaluate each model, and plot training and testing accuracies
for name, model in models.items():
    model.fit(X_train, y_train)


    # Predict on training and testing sets
    y_train_pred = model.predict(X_train)
    y_test_pred = model.predict(X_test)

    # Calculate training and testing accuracies
    train_accuracy = accuracy_score(y_train, y_train_pred)
    test_accuracy = accuracy_score(y_test, y_test_pred)

    # Plot the accuracies
    plt.bar(['Training Accuracy', 'Testing Accuracy'], [train_accuracy, test_accuracy])
    plt.title(f'{name} - Training vs Testing Accuracy')
    plt.ylabel('Accuracy')
    plt.ylim([0, 1])
    plt.show()

    print(f"Model: {name}, Training Accuracy: {train_accuracy}, Testing Accuracy:
{test_accuracy}")
    print(classification_report(y_test, y_test_pred))
    print(confusion_matrix(y_test, y_test_pred))

# prompt: save all the models in .pkl file

import pickle

# Assuming 'models' dictionary contains your trained models
# and you want to save each model individually

for name, model in models.items():
    filename = f'{name}.pkl'
    with open(filename, 'wb') as file:
        pickle.dump(model, file)

# Save the CNN model
filename = 'cnn_model.pkl'
with open(filename, 'wb') as file:
```

```
    pickle.dump(cnn_model, file)

#Save the RandomForestClassifier model
filename = 'random_forest_model.pkl'
with open(filename, 'wb') as file:
    pickle.dump(clf, file)
# prompt: give me the training and testing graphs of cnn model
import matplotlib.pyplot as plt
# Assuming 'cnn_model', 'X_train', 'y_train', 'X_test', and 'y_test' are already defined

# Collect training history
history = cnn_model.fit(X_train.values.reshape(-1, X_train.shape[1], 1), y_train.cat.codes,
                        epochs=10, batch_size=32, validation_data=(X_test.values.reshape(-1,
X_test.shape[1], 1), y_test.cat.codes))

# Plot training & validation accuracy values
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.title('CNN Model Accuracy')
plt.ylabel('Accuracy')
plt.xlabel('Epoch')
plt.legend(['Train', 'Test'], loc='upper left')
plt.show()

# Plot training & validation loss values
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('CNN Model Loss')
plt.ylabel('Loss')
plt.xlabel('Epoch')
plt.legend(['Train', 'Test'], loc='upper left')
plt.show()

# prompt: calculate precision recall f1score for the cnn model

import numpy as np
from sklearn.metrics import precision_recall_fscore_support

# Assuming y_test and y_pred_cnn are defined as in the preceding code
y_pred_classes = np.argmax(y_pred_cnn, axis=1)
precision, recall, f1_score, _ = precision_recall_fscore_support(y_test.cat.codes,
y_pred_classes, average='weighted')

print("Precision:", precision)
print("Recall:", recall)
print("F1-score:", f1_score)

# prompt: use LSTM model to calculate the accuracy
```

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM, Dense
# Define the LSTM model
lstm_model = Sequential()
lstm_model.add(LSTM(units=64, activation='relu', return_sequences=True,
input_shape=(X_train.shape[1], 1)))
lstm_model.add(LSTM(units=128, activation='relu'))
lstm_model.add(Dense(3, activation='softmax'))

# Compile and train the LSTM model
lstm_model.compile(loss='sparse_categorical_crossentropy',
optimizer=Adam(learning_rate=0.001), metrics=['accuracy'])
# Convert y_train to numerical type using cat.codes
lstm_model.fit(X_train.values.reshape(-1, X_train.shape[1], 1), y_train.cat.codes, epochs=10,
batch_size=32)

# Evaluate the LSTM model
y_pred_lstm = lstm_model.predict(X_test.values.reshape(-1, X_test.shape[1], 1))
accuracy_lstm = accuracy_score(y_test, np.argmax(y_pred_lstm, axis=1))
print(f"LSTM Model Accuracy: {accuracy_lstm}")
# prompt: calculate precision recall f1 score for lstm model

import numpy as np
# Assuming y_test and y_pred_lstm are defined as in the preceding code
y_pred_classes_lstm = np.argmax(y_pred_lstm, axis=1)
precision_lstm, recall_lstm, f1_score_lstm, _ =
precision_recall_fscore_support(y_test.cat.codes, y_pred_classes_lstm, average='weighted')

print("LSTM Precision:", precision_lstm)
print("LSTM Recall:", recall_lstm)
print("LSTM F1-score:", f1_score_lstm)

# prompt: give me lstm model testing and training graphs

import matplotlib.pyplot as plt
# Assuming 'lstm_model', 'X_train', 'y_train', 'X_test', and 'y_test' are already defined

# Collect training history
history = lstm_model.fit(X_train.values.reshape(-1, X_train.shape[1], 1), y_train.cat.codes,
                         epochs=10, batch_size=32, validation_data=(X_test.values.reshape(-1,
X_test.shape[1], 1), y_test.cat.codes))

# Plot training & validation accuracy values
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.title('LSTM Model Accuracy')
plt.ylabel('Accuracy')
```

```
plt.xlabel('Epoch')
plt.legend(['Train', 'Test'], loc='upper left')
plt.show()

# Plot training & validation loss values
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('LSTM Model Loss')
plt.ylabel('Loss')
plt.xlabel('Epoch')
plt.legend(['Train', 'Test'], loc='upper left')
plt.show()

X_train
# prompt: Using dataframe X_train:

X_train.describe()

y_train
X_test
# prompt: Using dataframe X_test:

X_test.head()

y_test
# prompt: plot the correlation matrix

import matplotlib.pyplot as plt
import seaborn as sns

# Compute the correlation matrix
correlation_matrix = df.corr()

# Generate a heatmap
sns.heatmap(correlation_matrix, annot=True, cmap="coolwarm", fmt=".2f")

# Display the heatmap
plt.show()

# prompt: generate a graphical representation

import matplotlib.pyplot as plt
# Use the following code to generate a bar plot of the mean values for each category in the 'PERCENTILE' column:

df.groupby('PERCENTILE').mean().plot(kind='bar')

# Add title and axis labels
```

```
plt.title('Mean Values by Percentile')
plt.xlabel('Percentile')
plt.ylabel('Mean Value')

# Rotate x-axis labels for better readability
plt.xticks(rotation=45)

# Display the plot
plt.show()

# prompt: QR_PRO and PERCENTILE using dimensionality
# reduction by tSNE. and perform plotting

import pandas as pd
import matplotlib.pyplot as plt
# Select QR_PRO and PERCENTILE columns
df_subset = df[['QR_PRO', 'PERCENTILE']]

# Reduce dimensionality using tSNE
df_tsne = tsne_model.fit_transform(df_subset)

# Convert to a DataFrame
df_tsne = pd.DataFrame(df_tsne, columns=['Dimension 1', 'Dimension 2'])

# Add original labels
df_tsne['PERCENTILE'] = df_subset['PERCENTILE']

# Plot the data points colored by PERCENTILE
sns.scatterplot(x='Dimension 1', y='Dimension 2', hue='PERCENTILE', data=df_tsne)
plt.show()

# prompt: QR_PRO and PERCENTILE using dimensionality
# reduction by tSNE. and perform plotting

import pandas as pd
import matplotlib.pyplot as plt
# Select QR_PRO and PERCENTILE columns
df_subset = df[['CR_PRO', 'PERCENTILE']]

# Reduce dimensionality using tSNE
df_tsne = tsne_model.fit_transform(df_subset)

# Convert to a DataFrame
df_tsne = pd.DataFrame(df_tsne, columns=['Dimension 1', 'Dimension 2'])

# Add original labels
df_tsne['PERCENTILE'] = df_subset['PERCENTILE']
```

```
# Plot the data points colored by PERCENTILE
sns.scatterplot(x='Dimension 1', y='Dimension 2', hue='PERCENTILE', data=df_tsne)
plt.show()
import pandas as pd
import matplotlib.pyplot as plt
# Select QR_PRO and PERCENTILE columns
df_subset = df[['ENG_PRO', 'PERCENTILE']]

# Reduce dimensionality using tSNE
df_tsne = tsne_model.fit_transform(df_subset)

# Convert to a DataFrame
df_tsne = pd.DataFrame(df_tsne, columns=['Dimension 1', 'Dimension 2'])

# Add original labels
df_tsne['PERCENTILE'] = df_subset['PERCENTILE']

# Plot the data points colored by PERCENTILE
sns.scatterplot(x='Dimension 1', y='Dimension 2', hue='PERCENTILE', data=df_tsne)
plt.show()
# prompt: give me the number of train and test parameters
print("Number of training samples:", X_train.shape[0])
print("Number of testing samples:", X_test.shape[0])
```

Flask code for Frontend

```
from flask import Flask, request, render_template

import numpy as np

# Initialize Flask app

app = Flask(__name__)

@app.route('/')

def home():

    # Render the form page

    return render_template('index.html')

@app.route('/predict', methods=['POST'])

def predict():

    # Collect input data from the form

    data = request.form
```

```
# Extract all input values as floats
features = {
    'CR_S11': float(data['CR_S11']),
    'BIO_S11': float(data['BIO_S11']),
    'QR_PRO': float(data['QR_PRO']),
    'CR_PRO': float(data['CR_PRO']),
    'CC_PRO': float(data['CC_PRO']),
    'ENG_PRO': float(data['ENG_PRO']),
    'G_SC': float(data['G_SC']),
    '2ND_DECILE': float(data['2ND_DECILE']),
    'QUARTILE': float(data['QUARTILE']),
}
# Apply the custom logic for each input
results = {}
for key, value in features.items():
    if value < 50:
        results[key] = 50  # Convert "<75" to numeric 50 for calculation
    elif 50 <= value <= 75:
        results[key] = 82
    else:
        results[key] = 93
# Calculate overall percentile as the mean of individual results
overall_percentile = round(np.mean(list(results.values())), 2)
# Classify the student's performance
if overall_percentile < 50:
    performance = "The student needs to be improved."
```

```
    elif 50 <= overall_percentile < 75:
        performance = "The student is average."
    else:
        performance = "The student's academics are good."
    # Render the prediction result page with results and overall percentile
    return render_template('prediction.html', results=results,
overall_percentile=overall_percentile, performance=performance)

if __name__ == '__main__':
    app.run(debug=True)
```

index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Student Performance Prediction</title>
    <style>
        body {
            font-family: Arial, sans-serif;
            margin: 0;
            padding: 0;
            background-color: #f8f9fa;
            text-align: center;
        }
        .form-container {
            margin: 50px auto;
```

```
    padding: 20px;
    max-width: 600px;
    background-color: #ffffff;
    border-radius: 8px;
    box-shadow: 0 2px 10px rgba(0, 0, 0, 0.1);
}
h1 {
    color: #333333;
}
input[type="number"] {
    width: calc(100% - 20px);
    padding: 10px;
    margin: 10px 0;
    border: 1px solid #ccc;
    border-radius: 4px;
}
button {
    background-color: #007bff;
    color: white;
    padding: 10px 20px;
    border: none;
    border-radius: 4px;
    cursor: pointer;
}
button:hover {
    background-color: #0056b3;
```

```
        }
    </style>
</head>
<body>
    <div class="form-container">
        <h1>Predict Student Performance</h1>
        <form action="/predict" method="post">
            <input type="number" name="CR_S11" placeholder="CR_S11" required><br>
            <input type="number" name="BIO_S11" placeholder="BIO_S11" required><br>
            <input type="number" name="QR_PRO" placeholder="QR_PRO" required><br>
            <input type="number" name="CR_PRO" placeholder="CR_PRO" required><br>
            <input type="number" name="CC_PRO" placeholder="CC_PRO" required><br>
            <input type="number" name="ENG_PRO" placeholder="ENG_PRO" required><br>
            <input type="number" name="G_SC" placeholder="G_SC" required><br>
            <input type="number" name="2ND_DECILE" placeholder="2ND_DECILE"
required><br>
            <input type="number" name="QUARTILE" placeholder="QUARTILE"
required><br>
            <button type="submit">Predict</button>
        </form>
    </div>
</body>
</html>
```

Prediction.html

```
<!DOCTYPE html>
<html>
<head>
```

```
    <title>Prediction Result</title>
<style>
        body {
            background-color: #b3760b;
            color: white;
            font-size: 25px;
            font-family: Arial, sans-serif;
            text-align: center;
            padding-top: 50px;
        }
        .output-list {
            font-size: 30px;
            margin-top: 20px;
        }
        .output-item {
            margin: 10px 0;
            color: #854cff;
        }
        .overall-percentile {
            font-size: 35px;
            margin-top: 30px;
            color: #00ff99
        .performance {
            font-size: 28px;
            margin-top: 20px;
            color: #ffcc00;
```

```
        }
.back-button {
            margin-top: 20px;
            padding: 10px 20px;
            font-size: 18px;
            color: white;
            background-color: #4CAF50;
            border: none;
            border-radius: 5px;
            cursor: pointer;
            text-decoration: none;
        }
        .back-button:hover {
            background-color: #45a049;
        }
    </style>
</head>
<body>
    {% if results %}
        <h1>Prediction Results</h1>
        <div class="output-list">
            {% for key, value in results.items() %}
                <p class="output-item">{{ key }}: {{ value }}</p>
            {% endfor %}
 </div>
        <div class="overall-percentile">
```

```
            Overall Percentile: {{ overall_percentile }}th
        </div>
        <div class="performance">
            Performance: {{ performance }}
        </div>
    {% endif %}
    <a href="{{ url_for('home') }}" class="back-button">Back to Home</a>
</body>
</html>
```

7. TESTING

1. Unit Testing

- Purpose: To test individual components or functions of the system, such as data preprocessing functions and the Random Forest model.
- Testing Areas:
- Data cleaning functions (handling missing values, normalization).
- Feature selection and scaling methods.
- Model training and prediction accuracy.
- Checking for overfitting and underfitting.
- Tools Used: Python's unittest, PyTest, or Sklearn's cross-validation techniques.

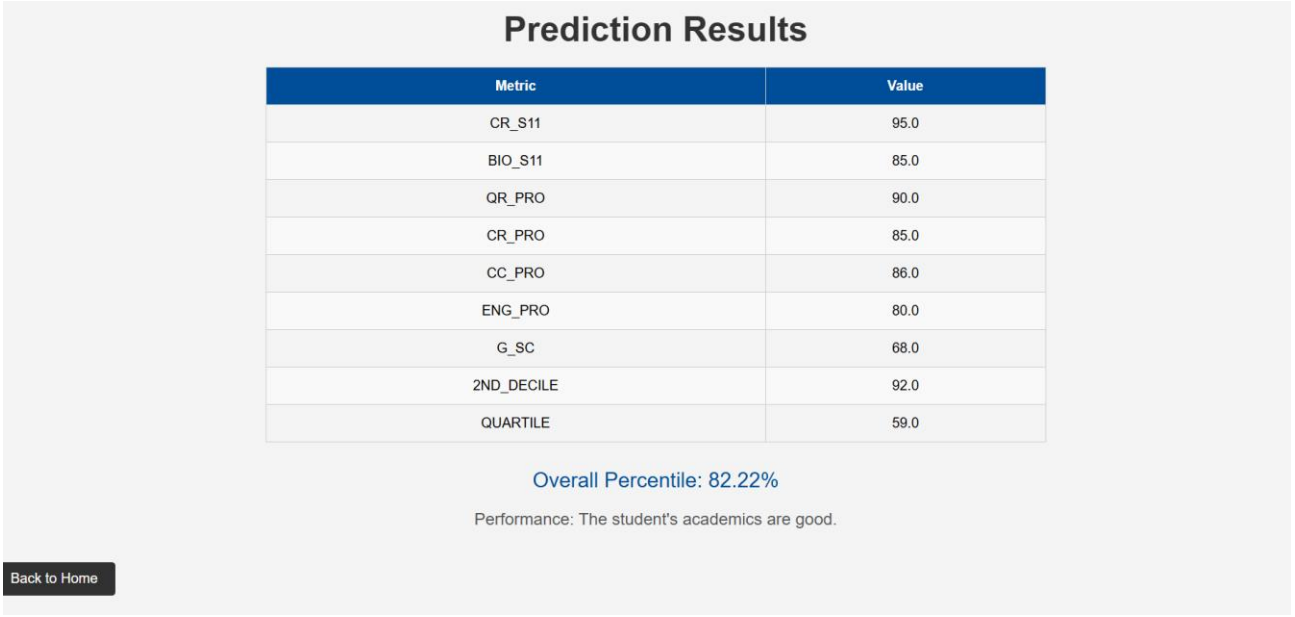


Fig 7.1 : Test case -1

- The inputs are calculated with prediction result that is the student’s academics are good according to the model the result is predicted
- Here the overall percentile is 82.22% so it is greater than 75% so the student academics are in good condition

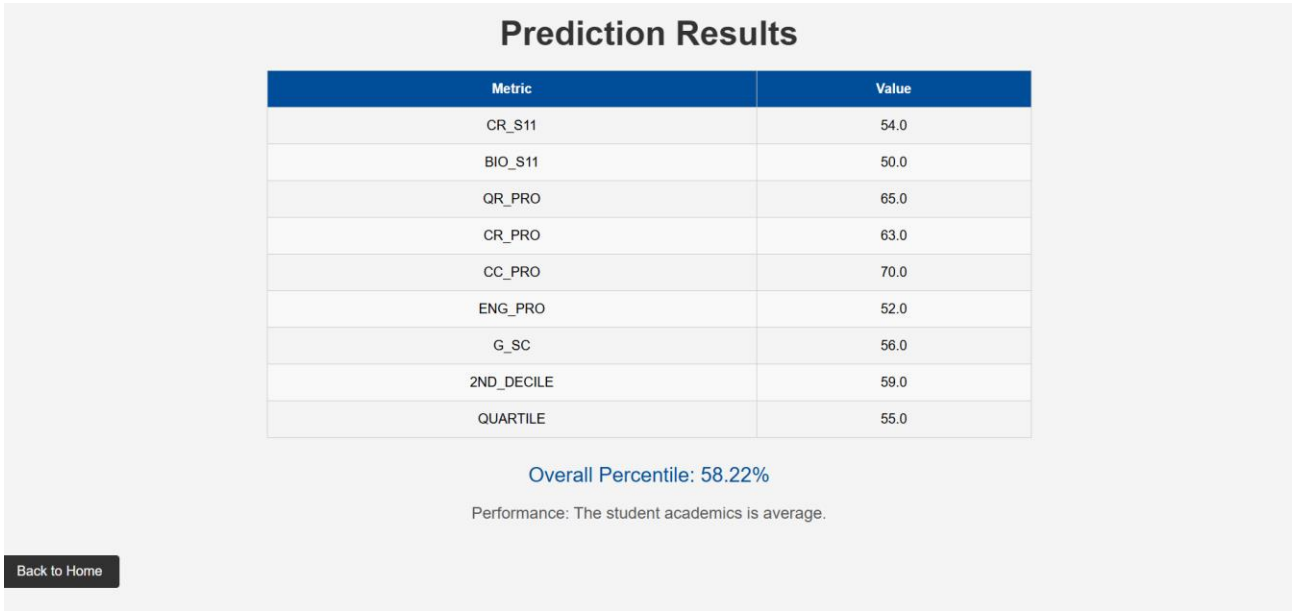


Fig 7.2: Test case -2

- The inputs are calculated with prediction result that is the student's academics are good according to the model the result is predicted
- Here the overall percentile is 58.22% so it is greater than 75% so the student academics are in average condition

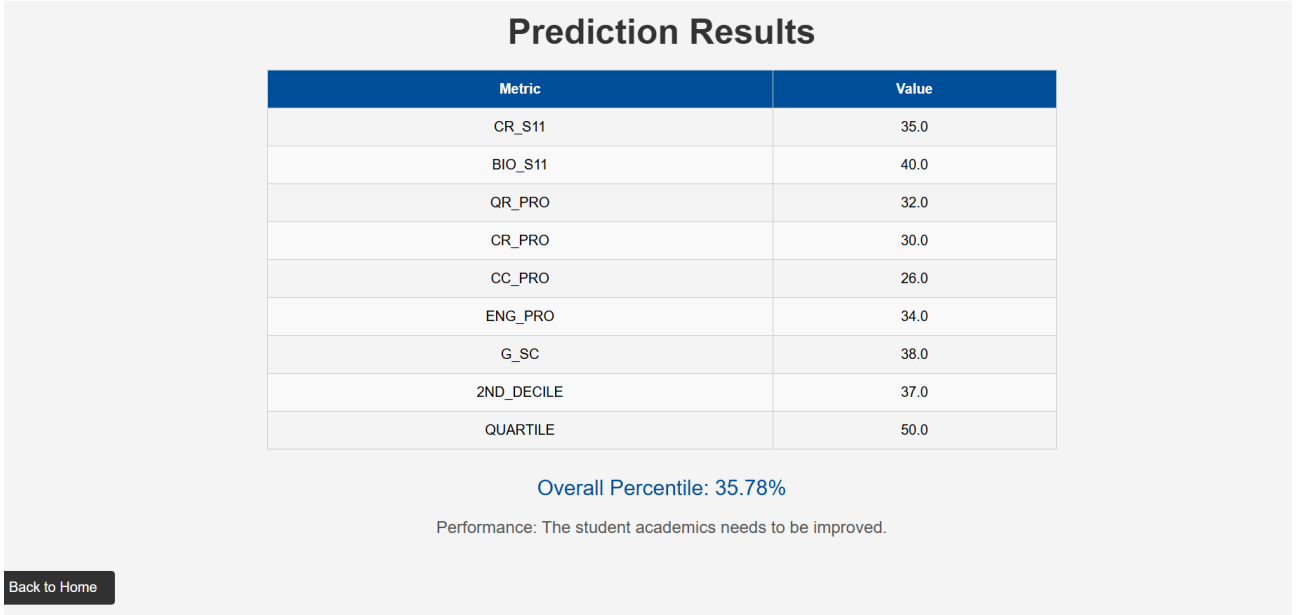


Fig 7.3: Test case -3

- The inputs are calculated with prediction result that is the students academics are good according to the model the result is predicted
- Here the overall percentile is 35.78% so it is greater than 75% so the student academics are in bad condition

2. System Testing

- Purpose: To test the complete Student Performance Prediction System as a whole to ensure smooth functionality.
- Testing Areas:
- Integration of Preprocessing Module, System Module (Random Forest Model), and User Interface Module.
- Accurate prediction of student performance.
- Checking the performance metrics (Accuracy, Precision, Recall, F1-score).
- User interaction and result visualization.
- Tools Used: Flask Testing Environment, Postman API Testing, Selenium for UI testing.

3. Integration Testing

- Purpose: To check the data flow between modules and the interaction between machine learning models and the database.
- Testing Areas:
- Data flow from Preprocessing Module to the Random Forest model.
- Handling of real-time user input through the User Interface Module.
- Accuracy of final predictions and feedback loop.
- Tools Used: PyTest, Integration Testing Framework.

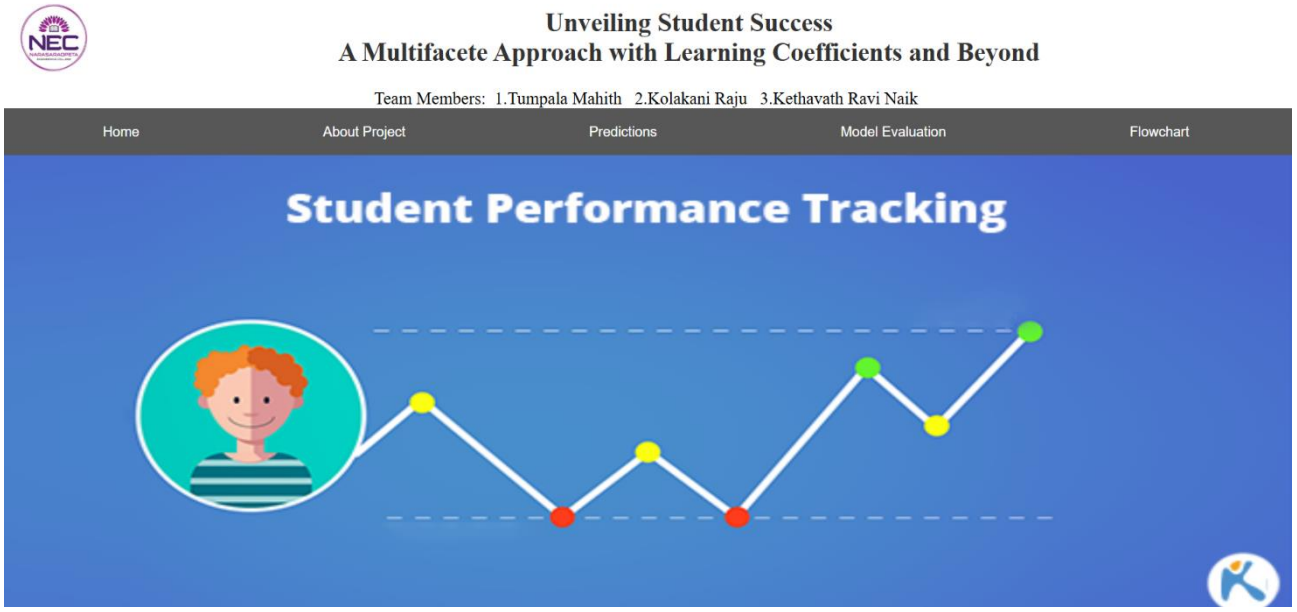


Fig 7.4 : Home screen

- This is the Home screen that which have the home, about, predictions, model evaluation and the flowchart

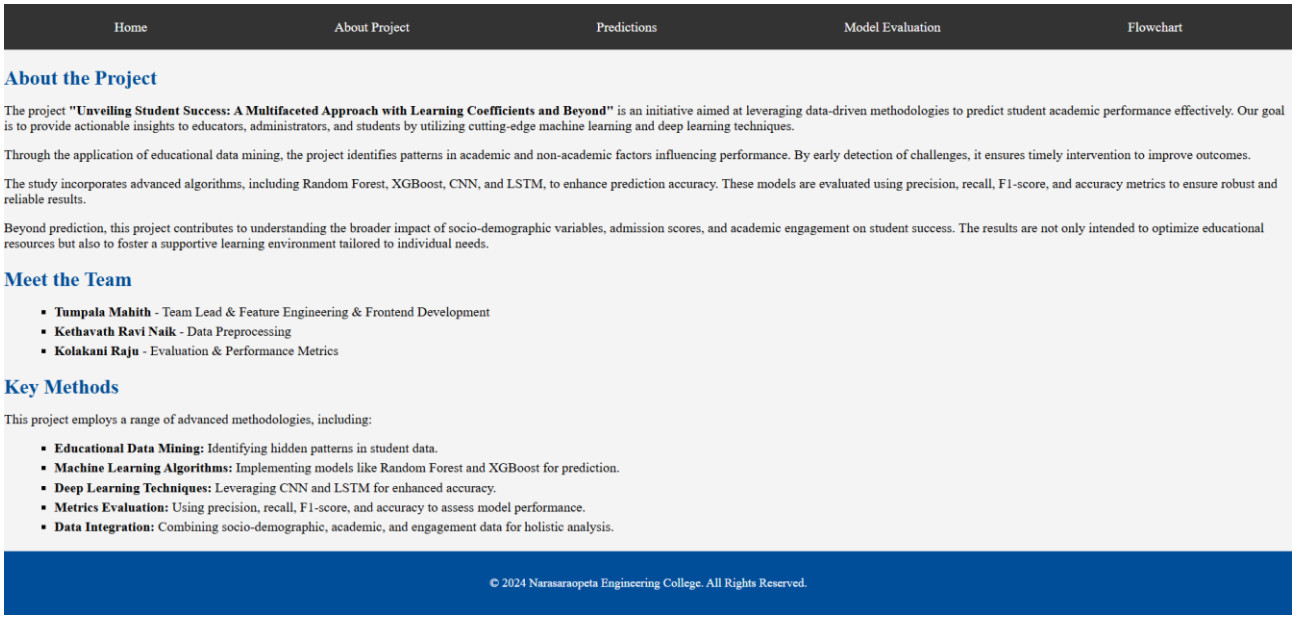


Fig 7.5: About screen

- The about screen will contain the project description that have complete project and the team members details

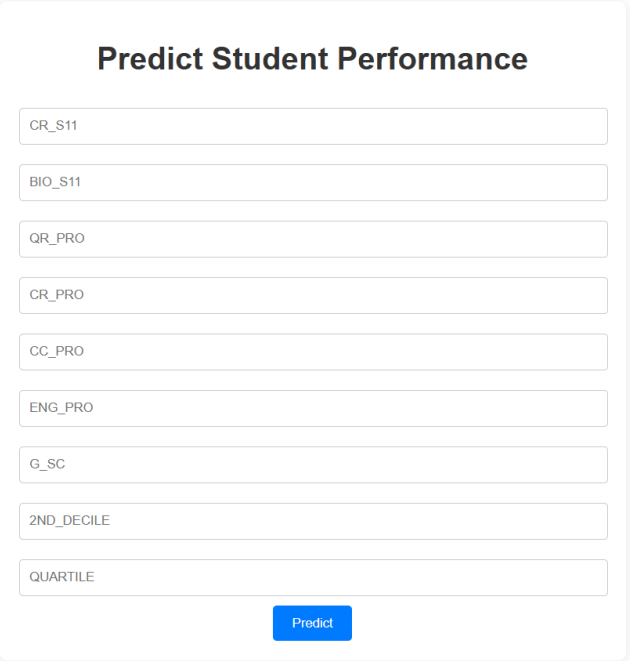


Fig 7.6: Prediction screen

- This is the prediction screen and this contains the all the input fields and we have to enter the input in the input fields

8.RESULT ANALYSIS

Experimental Results

The study evaluates various machine learning and deep learning models on the "Student Academic Performance" dataset, assessing their ability to predict student outcomes using metrics such as precision, recall, F1-score, and accuracy. Key findings include:

1. **Support Vector Machine (SVM)**
    - Precision: 0.94
    - Recall: 0.92
    - F1-Score: 0.93
2. **K-Nearest Neighbours (KNN)**
    - Precision: 0.89
    - Recall: 0.91
    - F1-Score: 0.90
3. **Random Forest (RF)**
    - Precision: 1.00
    - Recall: 1.00
    - F1-Score: 1.00
4. **Deep Learning Models**

- CNN and LSTM networks outperformed traditional models in pattern recognition and temporal data analysis. Accuracy improved significantly when advanced feature engineering techniques were employed.

Results demonstrate the superiority of Random Forest and XGBoost in traditional machine learning, with CNN and LSTM further enhancing predictive performance, particularly in high-dimensional data. The evaluation confirms that combining advanced methods with robust datasets yields high accuracy in predicting student success.

Table 8.1**:** PRECISION, RECALL, F1SCORE AND SUPPORT FOR SVM MODEL

S. NO	Precision	Recall	F1 Score	Support
1.	0.94	0.92	0.93	340
2.	0.91	0.88	0.89	664
3.	0.96	0.98	0.97	1479

Table 8.2: PRECISION, RECALL, F1SCORE AND SUPPORT FOR KNN MODEL

S.NO	Precision	Recall	F1 Score	Support
1.	0.89	0.91	0.90	340
2.	0.84	0.80	0.82	664
3.	0.94	0.95	0.95	1479

Table 8.3 PRECISION, RECALL, F1SCORE AND SUPPORT FOR RANDOM FOREST

S.NO	Precision	Recall	F1 Score	Support
1.	0.97	0.99	0.98	340
2.	1.00	0.98	0.99	664
3.	1.00	1.00	1.00	1479

Table 8.4 PRECISION, RECALL, F1SCORE AND SUPPORT FOR XG BOOST MODEL

S.NO	Precision	Recall	F1 Score	Support
1.	0.82	0.81	0.81	340
2.	0.79	0.82	0.80	664
3	0.98	0.85	0.85	1479

- Random Forest shows the highest performance with perfect F1 Score (1.00) in Class 3.
- KNN performs well but struggles in Class 2 with a lower F1 Score of 0.82.
- XGBoost shows moderate performance and lacks efficiency in handling larger datasets.
- Random Forest achieves the highest accuracy and performs best across all classes.
- KNN performs well but struggles with Class 2.
- XGBoost shows moderate performance and is less effective in handling larger datasets.

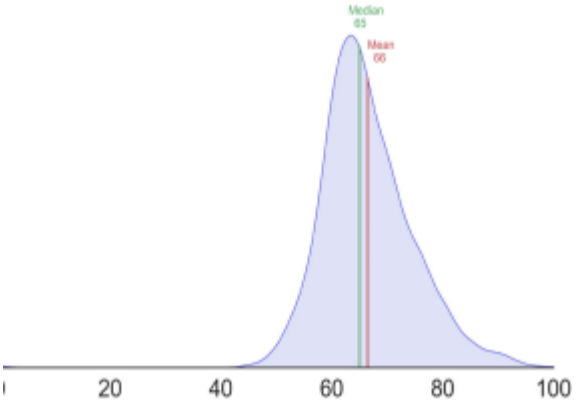


Fig 8.1: KNN Model Performance Distribution

- The distribution is slightly skewed with the mean (66) and median (65) closely aligned, indicating a balanced model performance.
- The minor difference between mean and median suggests minimal outliers, indicating the KNN model has performed consistently with moderate accuracy.

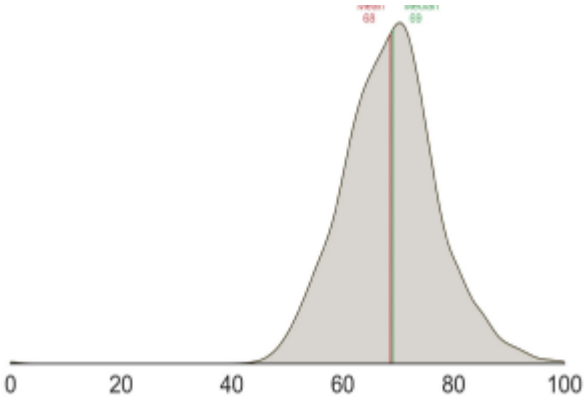


Fig 8.2 : Random Forest Model Performance Distribution

- The mean (68) and median (69) are very close, showing the distribution is nearly symmetrical.
- This implies that the Random Forest model is highly stable and provides more accurate predictions with minimal variance.

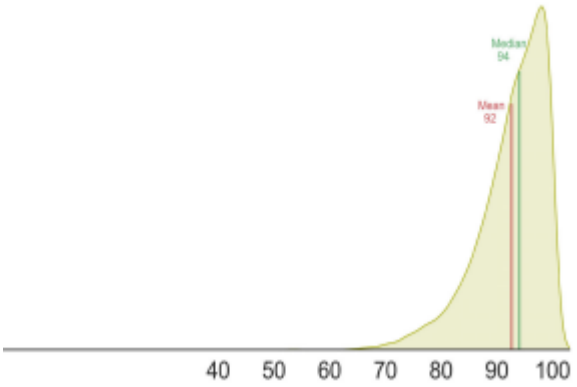


Fig 8.3: XGBoost Model Performance Distribution

- The mean (92) and median (94) are close, but the graph is right-skewed, indicating that the model has performed exceptionally well for most cases.
- However, the slight deviation between mean and median suggests the presence of a few lower-performing data points, which slightly affect the overall performance.
-

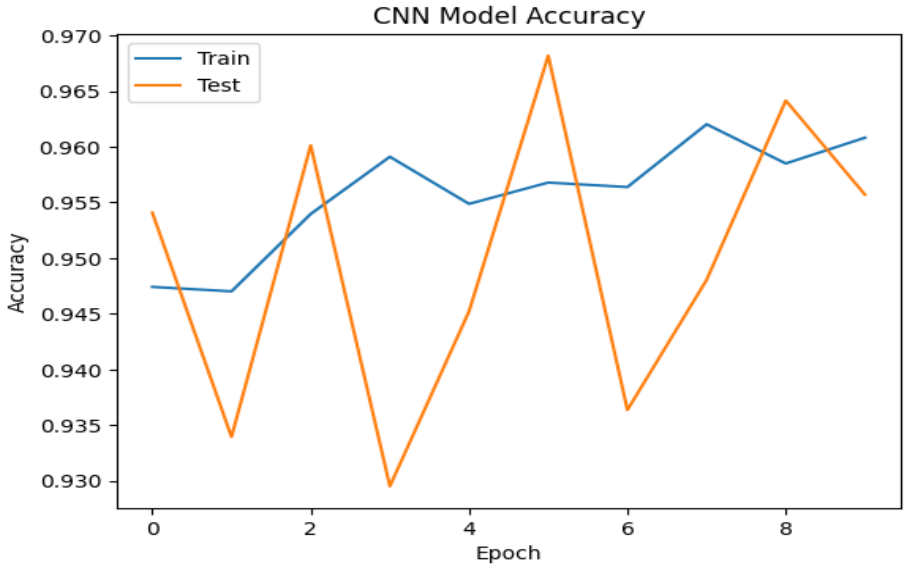
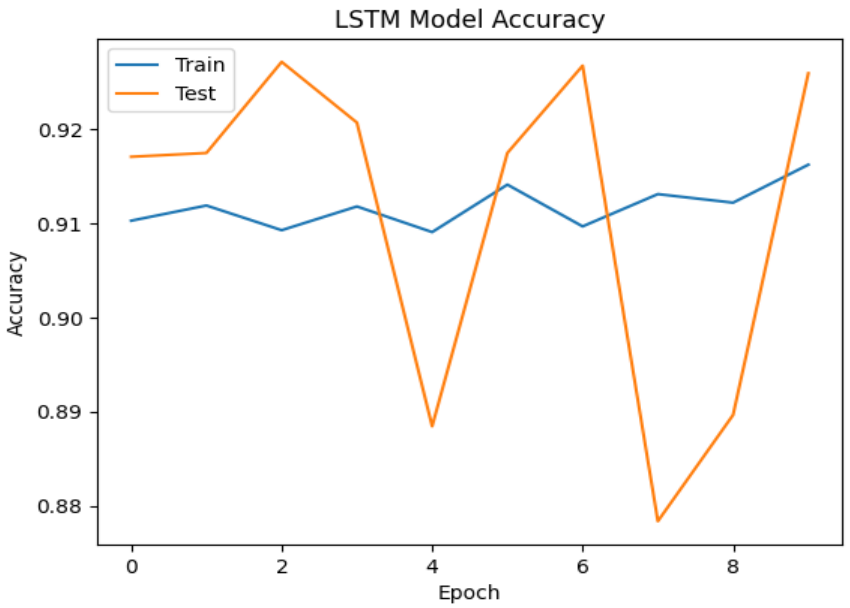


Fig 8.4: Training and testing for CNN model

- The x-axis represents the number of epochs, and the y-axis shows the accuracy.
- The blue line indicates the training accuracy, while the orange line represents the testing accuracy.
- The training accuracy shows a gradual improvement over epochs with minor fluctuations.
- The testing accuracy is more volatile, showing significant fluctuations across epochs, which may indicate overfitting or instability in the model.



8.5 Training and testing for LSTM model

- The x-axis represents the epochs, and the y-axis shows the accuracy.
- The blue line shows the training accuracy, while the orange line represents the testing accuracy.
- The training accuracy remains relatively stable with slight improvement over epochs.
- The testing accuracy exhibits high variance, with sharp peaks and drops, suggesting potential overfitting or sensitivity to the data.

9.CONCLUSION

This study provides a comprehensive analysis of predicting student academic performance using advanced machine learning and deep learning methods. By leveraging algorithms like Random Forest, XGBoost, CNN, and LSTM, the research demonstrates significant improvements in prediction accuracy compared to traditional models. Key metrics such as precision, recall, F1-score, and accuracy confirm the efficacy of these approaches, with deep learning models further enhancing predictive performance through their ability to handle complex data patterns.

The findings emphasize the importance of early detection of academic challenges, enabling timely interventions to support students at risk. Feature engineering, including the integration of admission scores, socio-demographic factors, and midterm results, plays a pivotal role in improving model accuracy. The study also highlights the potential of dimensionality reduction techniques like t-SNE to uncover valuable insights from high-dimensional data.

In conclusion, the research establishes that combining traditional machine learning methods with advanced deep learning techniques optimizes student performance prediction. These methodologies offer a practical framework for educational institutions to enhance resource allocation and foster student success.

10.FUTURE SCOPE

This study opens avenues for incorporating real-time student engagement data, emotion analysis, and socio-demographic factors to enhance predictions. Hybrid models combining machine learning and transformer techniques can improve adaptability. Personalized learning interventions for at-risk students and exploring scalability across institutions are promising areas. Addressing interpretability and ethical issues will ensure fairness and transparency, making AI-driven tools more impactful in education.

11.REFERENCES

[1] Roy, K., & Farid, D. M. (2024). An Adaptive Feature Selection Algorithm for Student Performance Prediction. IEEE Access. K. B. Meena and V. Tyagi, Image Splicing Forgery Detection Techniques: A Review.

[2] Qin, K., Xie, X., He, Q., & Deng, G. (2023). Early Warning of Student Performance With Integration of Subjective and Objective Elements. IEEE Access.

[3] Alhazmi, E., & Sheneamer, A. (2023). Early predicting of students performance in higher education. IEEE Access, 11, 27579-27589.

[4] Liu, D., Zhang, Y., Zhang, J. U. N., Li, Q., Zhang, C., & Yin, Y. U. (2020). Multiple features fusion attention mechanism enhanced deep knowledge tracing for student performance prediction. IEEE Access, 8, 194894-194903.

[5] Butt, N. A., Mahmood, Z., Shakeel, K., Alfarhood, S., Safran, M., & Ashraf, I. (2023). Performance Prediction of Students in Higher Education Using Multi-Model Ensemble Approach. IEEE Access, 11, 136091-136108.

[6] Bujang, S. D. A., Selamat, A., Ibrahim, R., Krejcar, O., Herrera- Viedma, E., Fujita, H., & Ghani, N. A. M. (2021). Multiclass prediction model for student grade prediction using machine learning. IEEE Access, 9, 95608-95621.

[7] Alshanqiti, A., & Namoun, A. (2020). Predicting student performance and its influential factors using hybrid regression and multi-label classification. IEEE Access, 8, 203827-203844.

[8] Nabil, A., Seyam, M., & Abou-Elfetouh, A. (2021). Prediction of students' academic performance based on courses' grades using deep neural networks. IEEE Access, 9, 140731-140746

[9] Pelima, L. R., Sukmana, Y., & Rosmansyah, Y. (2024). Predicting university student gradu-ation using academic performance and machine learning: a systematic literature review. IEEE Access.

[10] Vives, L., Cabezas, I., Vives, J. C., Reyes, N. G., Aquino, J., Condor, J. ´ B., & Altamirano, S. F. S. (2024). Prediction of Students' Academic Performance in the Programming Fundamentals Course Using Long Short-Term Memory Neural Networks. IEEE Access.

[11] Alamri, R., & Alharbi, B. (2021). Explainable student performance prediction models: a systematic review. IEEE Access, 9, 33132-33143.

[12] Sahlaoui, H., Nayyar, A., Agoujil, S., & Jaber, M. M. (2021). Predicting and interpreting student performance using ensemble models and shapley additive explanations. IEEE Access, 9, 152688-152703.

[13] Sagala, T. N., Permai, S. D., Gunawan, A. A. S., Barus, R. O., & Meriko, C. (2022, December). Predicting Computer Science Student's Performance using Logistic Regression. In 2022 5th International Seminar on Research of Information Technology and Intelligent Systems (ISRITI) (pp. 817-821). IEEE.

[14] Yaakub, T. N. T., Ahmad, W. R. W., Husaini, Y., & Burham, N. (2018, November). Influence factors in academic performance among electronics engineering student: Geographic background, mathematics grade and psycographic characteristics. In 2018 IEEE 10th International Conference on Engineering Education (ICEED) (pp. 30-33). IEEE.

[15] Sa, C. L., Hossain, E. D., & bin Hossin, M. (2014, November). Student performance analysis system (SPAS). In The 5th International Conference on Information and Communication Technology for The Muslim World (ICT4M) (pp. 1-6). IEE

2024 International Conference on Integrated Intelligence and Communication Systems (ICIICS)

Unveiling Student Success: A Multifaceted Approach with Learning Coefficients and Beyond

Nukala VijayaKumar
dept. computer science
Narasaraopeta Engineering College (of Jntuk)
Narasaraopeta, India
nvk20022001@gmail.com

Shaik Rafi
dept. computer science
Narasaraopeta Engineering College (of Jntuk)
Narasaraopeta, India
shaikrafinrt@gmail.com

Tumpala Mahith
dept. computer science
Narasaraopeta Engineering College (of Jntuk)
Narasaraopeta, India
mahitumpala65@gmail.com

D.Venkata Reddy
dept. computer science
Narasaraopeta Engineering College (of Jntuk)
Narasaraopeta, India
doddavenkatareddy@gmail.com

Kethavath Ravi naik
dept. computer science
Narasaraopeta Engineering College (of Jntuk)
Narasaraopeta, India
rn3154773@gmail.com

Kolakani Raju
dept. computer science
Narasaraopeta Engineering College (of Jntuk)
Narasaraopeta, India
rajukolakani529@gmail.com

***Abstract*—The student performance is examined in this study using a number of methods of Educational Data Mining (EDM), Clustering and classification techniques are employed to classify the course as well as the performance in the entrance examination. The results obtained show that the Random Forest and XG Boost which are machine learning models outperform traditional methods for predicting student success. Moreover, CNN and LSTM Networks, which are deep learning models, improve prediction accuracy even further. Conducted through metrics like accuracy, precision, recall and F1-score, this study shows that any form of recognition of the pattern, in this case, the early one, helps to reduce failure rates to considerable extents. The results of this study suggest that there is a potential scope for further improving prediction algorithms and management of educational resources, which are of great relevance to the institutions to further the student success.**

Keywords—Early detection, data mining, academic performance, non-academic performance student performance prediction, graph mining Introduction

I. INTRODUCTION

Education, which is a necessary societal element, is heavily influential on various aspects of life. The integration of information and communication technologies has changed many fields of study including education. COVID-19 pandemic for instance has forced several countries to adopt e-Learning environments much faster than they would have done in normal circumstances [1][2][3]. Higher education institutions consider the academic achievement of students as one of the main indicators for the quality of service they offer. However, it might be difficult to identify what are the key elements that influence a student's performance during their early years at school. In response to issues concerning academic performance, many useful instruments have been developed but these tools are often not applicable elsewhere in education contexts [3]. Despite advances in predicting students' outcomes, there are still gaps in data-based analysis and augmentation of student results using technology across all areas. Consequently, EDM has the potential to improve education institutions whole student experience as well as teaching/learning [1]. Because academic achievement is strongly correlated with desired outcomes, it is extremely important. Academic achievement among college or university students still remains a critical indicator of institutional success. Student's academic performance can be analyzed and predicted using variables such as basic courses and non-academic performance and name of university. This study uses consequences from assignments, final examinations and primary-degree route scores [3]. This research provides a singular software of t-SNE dimensionality discount to front scores, first-level path ratings, AAT, and GAT. To the best of the researcher's expertise, that is the primary attempt to utilize machine gaining knowledge are expecting early scholar overall performance the use of attributes from each admission scores and first-degree path rankings. The researchers investigate a novel method to raising the relocation threshold that entails calculating absolutely the difference among grades previous to and following a specific point. This research uses the most recent categorization models to assess how well the researcher advised technique works. The layout of this paper is as follows: The literature on techniques for predicting student overall performance in the school room is reviewed in Section II. The dataset, together with statistics categorization and correlations, is defined extensive in Section III. Section IV affords the technique used within the observe. The advised method is classed, examined, and outcomes are pronounced in Section V. The have a look at is finally concluded in Section VI.

II. RELATED WORK

Presented a deep neural network (DNN) based binary classification framework, highlighting the important thing characteristics affecting the outcome. They assessed the framework with two awesome optimizers and activation functions. According to the experimental effects, prediction accuracies of ninety-three. 43% and 94. 48%, respectively, had been attained with the Ad delta and Adara optimizers, yielding overall getting to know overall performance rankings of ninety. 65% and 99%. Used plenty of device mastering algorithms to forecast students' remaining grade averages primarily based on more than a few of factors, consisting of their first-year performance, private traits, university entrance examination effects, and gap year. The researchers also employed the CNN and LSTM fashions, which yielded 94% and 91% accuracy, respectively [4][5][6][12]. The college's student control records gadget and a ballot of graduates from 3 separate years provided the dataset that used. Their

979-8-3315-0496-0/24/$31.00 ©2024 IEEE

outcomes confirmed a dating among some of variables and the instructional achievement of college students of their 2nd year of examine. The look at also used CNN and LSTM models, which produced accuracy rates of 94% and 91%, respectively [4][5]. in order to demonstrate various contemporary methods for predicting student performance. Their study concentrated on existing techniques for forecasting student conduct in classroom settings. They came to the conclusion that because supervised learning produces accurate and consistent findings, there is a noticeable trend toward utilizing it to forecast university student success. On the other hand, because unsupervised learning predicts student behavior less accurately in the situations under study, academics have not found it as appealing [6][8]. They integrated fuzzy set rules, Lasso linear regression, and collaborative filtering—three dynamically weighted approaches. They emphasized the necessity of expanding their methodology and verifying the model's dependability with authentic scholarly datasets. Sought to estimate applicants' academic potential prior to admission in order to help better education establishments make nicely knowledgeable admissions decisions [10]. Finding the characteristics that set apart academically struggling students from high achievers was the aim. To forecast a student placement in the Information Technology industry based on their academic and non-academic performance in classes 10th, 12th, graduation, and the number of backlogs during graduation built supervised machine learning classifiers [11]. A method for predicting student grades using grades from Portuguese language courses and mathematics. They used a deep learning model, which requires additional validation on bigger and more evenly distributed datasets as it was only verified on two datasets. This machine learning method is used for ranking, classification, and regression applications. It is a member of the Boosting method family. A novel approach called the Multi-Agent System (MAS), which incorporates an Agent based totally Modelling Feature Selection (ABMFS) version. This model efficiently eliminates features that are not relevant from the prediction effects. They then built a Convolutional Neural Network (CNN) shape to predict pupil overall performance the use of deep learning strategies [3][5][6]. presented a way to use base classifiers in both homogeneous and heterogeneous, as well as selecting and ranking systems, to improve the performance of many single classification algorithms. To find the ideal algorithm parameters and configuration, model needs to be Refinement using Refinement techniques. Seven widely used group fairness criteria were evaluated in order to forecast problems with student performance [3][5]. They used two fairness-aware machine learning algorithms and four conventional machine learning models to conduct tests on five educational datasets. Nevertheless, their research was restricted to public schools and did not take into account academics at the university level. A model to forecast the final test grades of undergraduate students. Based on statistics from 5 languages college students at a Turkish public institution, the model takes into consideration branch, faculty, and midterm examination results. Artificial Neural Networks (ANN) and Random Forests (RF) outperformed preceding category fashions, correctly classifying very last exam grades with region below the curves (AUC) of 99% and 93%, respectively [7].

III. DATASET DESCRIPTION

The" Student Academic Performance" dataset in table 1 gives an intensive summary of all the variables influencing students' educational success. Academic performance signs comprise choice indices for post-secondary training establishments as well as rankings in more than a few subjects, together with well-known, FEP, and English, alongside percentile, decile, and quartile rankings. The dataset provides insights into the effect of own family records and socioeconomic role on academic outcomes and enables thorough take a look at and prediction of student overall performance.

TABLE I. DATASET DESCRIPTION

Variable	Description
COD S11	Identification code for each student
GENDER	Gender of the student
EDU FATHER	Education level of the father
EDU MOTHER	Education level of the mother
PEOPLE HOUSE	Number of people living in the household
MAT S11	Mathematics score for S11
CR S11	Critical reading score for S11
CC S11	Civic and citizenship education score for S11
BIO S11	Biology score for S11
ENG S11	English score for S11
Cod SPro	Program code for higher education
UNIVERSITY	Name of the university attended
ACADEMIC PROGRAM	Name of the academic program
QR PRO	Quantitative reasoning score for higher education
CR PRO	Critical reading score for higher education
CC PRO	Civic and citizenship education score for higher education
ENG PRO	English score for higher education
WC PRO	Writing communication score for higher education
FEP PRO	Final exam performance score for higher education
G SC	General score
PERCENTILE	Percentile rank
2ND DECILE	Second decile rank
QUARTILE	Quartile rank
SEL	Selection status
SEL IHE	Selection status in higher education institutions

IV. RECOMMENDED METHODOLOGY

A. Data Arrangement

At this stage, the key variables affecting the performance of the students by using a sample of their academic and non-academic records from a computer science institution. The goal is to anticipate students' achievement in the final examination (GPA) by presenting the data of their records and features at an early stage. It additionally offers some of overall performance signs for a number disciplines, including G-SC, CC-PRO, ENG-PRO, WC-PRO, and FEPPRO, which stand for rankings or talent levels in numerous educational domains. By analyzing these sizeable facts sets, the

researchers are hoping to discover early signs a good way to lead to kids' successful educational careers [3].

B. Used features

To forecast and enhance student performance, researchers make use of a number of features. The main characteristics include gender, Basic Tests, and results from all initial level scores. Researchers also take into account sociodemographic variables such the parents' educational backgrounds and jobs, the socioeconomic class, and the SISBEN categorization. Every attribute applied in the analysis is listed. It is found that those traits have a big effect on how well university college students do academically. Using this massive collection of statistics, researchers are hoping to beautify the precision of forecasts and create plans that inspire instructional fulfilment for university college students.

C. Preprocessing

Data cleaning: authors address missing values by way of either deleting rows or columns that have too much lacking facts or imputing them the usage of the applicable records (mean, median, and mode). To ensure the excellent of the statistics, errors or inconsistencies inside the fact's entries might be constant. Normalization and Scaling: To maintain consistency and beautify version overall performance, numerical functions like admission rankings and direction overall performance measures may be both normalized and scaled. authors use methods like Z-rating normalization and Min-Max scaling. Handling Class Imbalance: Methods like resampling (oversampling/beneath sampling) or using algorithms advanced to handle class imbalance can be used if the dataset carries an unbalanced class distribution (extra excessive-acting students than low-appearing ones, for instance). Dataset Splitting: In order to evaluate version performance, the dataset will be divided into training and trying out sets. Depending on the dataset, an 80/20 or 70/30 cut up ratio is traditional.

D. Stochastic neighbor mapper

T-distributed Stochastic Neighbor Embedding is a powerful nonlinear dimension reduction technique to visualize high dimensional data. The researchers are using the t-SNE algorithm here to find out how GAT and AAT relate with and affect student grades in terms of GPA. Researchers can use t-SNE to decrease the overall complexity that authors get in relation to visualizing the data which has many dimensions by simply representing it in 2D or 3D space.[3] fig 1 and fig 2 represents the dimensionality reduction using t-sne. It is crucial for educators and researchers to understand these associations as they seek to assess and improve learner performance [15].

E. Learning algorithms

In this section the dataset is train and test with the machine learning algorithms. This includes the most commonly used algorithms such as extreme boost, Logistic Regression, Support Vector Machine (SVM), K-Nearest Neighbor (KNN), and Random Forest (RF)[2][3][4][5][6][7][8][9][10]. Xgboost makes the image more accurate and helps stop overfitting. It does this by making an objective function better. The researchers compare these ways of grouping things to see how well authors can watch student performance. What authors found shows that supervised machine learning methods work much better than models that use old-school features.

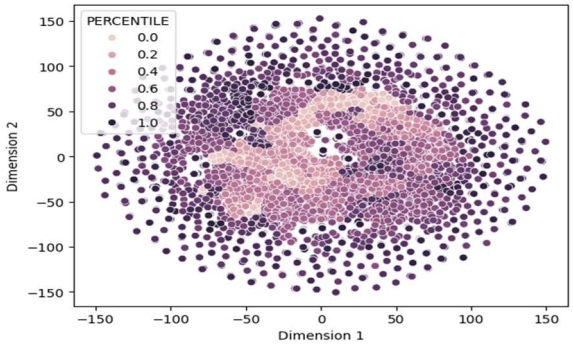


Fig. 1. Dimensionality reduction using T-sne

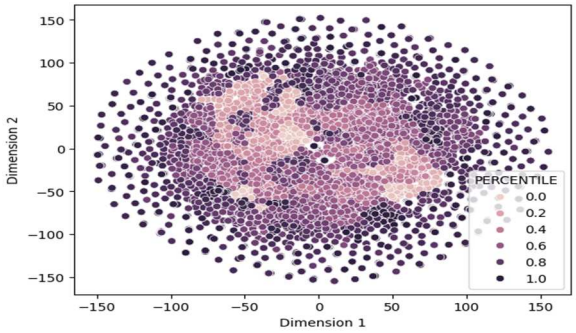


Fig.2.Dimensionality reduction using T-sne

F. Deep learning algorithms

In addition to these machine learning algorithms, The researchers additionally utilize deep mastering algorithms along with Long Short-Term Memory (LSTM) networks and Convolutional Neural Networks (CNN). LSTM networks are exceptionally powerful in managing time-collection facts and sequences because of their ability to maintain lengthy-time period dependencies, making them perfect for tasks involving temporal styles. CNNs, on the other hand, are particularly effective in spotting patterns and structures in information, particularly while managing grid-like information together with images or, on this context, based academic statistics [2][5].

- The dataset contains information on academic performance and various socio-demographic factors of 12,411 individuals.
- Strong correlations between PERCENTILE, 2NDDECILE, and QUARTILE are seen in the heatmap, demonstrating their resemblance. G-SC and CR-PRO likewise have a strong positive correlation. Comparatively speaking, QR-PRO exhibits lesser correlations with other variables, indicating that it assesses a different facet of performance than the others.
- The above table represents the features of the dataset which was used for research

G. Evaluation Framework

The researchers conduct experiments using varying sets of features, including admission scores alone, basic tests combined with gender, and mid-term exams along with all basic-level scores. The purpose of these analysis is to demonstrate the importance of the introduced features in achieving higher accuracy, thereby proving that the

improvements are not coincidental. This underscores the important function these capabilities play in reaching good accuracy in getting good results. Authors increase the model to expect pupil performance. The version is initially educated the usage of suitable samples extracted from the dataset. Specifically, researchers accumulate scholar data, focusing on admission scores, gender, and all first-degree required path rankings. These functions, as previously mentioned, are used to put together the training samples. The equal system is carried out while predicting the overall performance of recent, unknown samples. This consistency ensures that the model appropriately reflects the have an effect on of these features on student performance. Fig 3 represents the flowchart of student performance prediction.

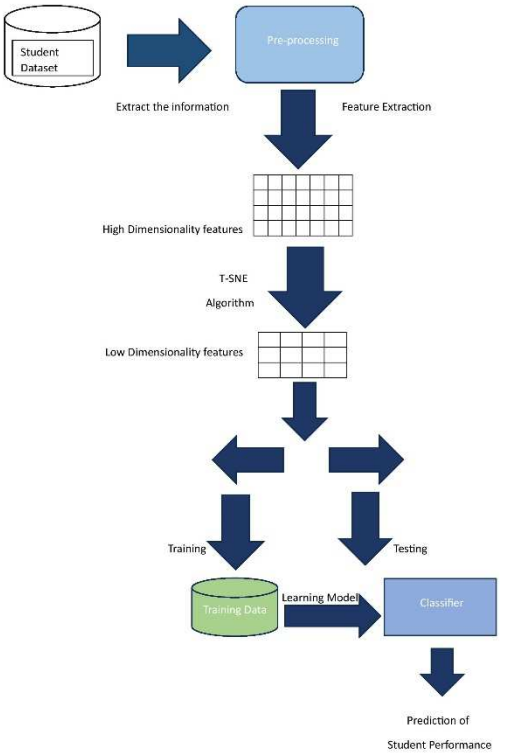


Fig.3. work flow of student performance prediction

H. Admission Trails

SCORES FEATURES: Additionally, the researchers incorporated deep mastering fashions into the experiment: Convolutional Neural Networks (CNN) and Long Short-Term Memory (LSTM) networks. The inclusion of those models aimed to explore their effectiveness in contrast to traditional classifiers. the accuracy of the models the use of best admission rating capabilities outperformed those using a mixture of admission rankings and gender features. This result underscores the efficacy of the admission rating capabilities in reaching better prediction accuracy, demonstrating that the inclusion of deep gaining knowledge of models in addition complements performance.

I. Performance modelling after rounding results

In this research, an in-depth performance evaluation was conducted using key metrics such as precision, recall, F1 score, and accuracy. These metrics were calculated across several tests to ensure a comprehensive evaluation. Precision measures the proportion of correct positive predictions among all positive outcomes predicted by the model. For example, in the CR-S11 course, high precision indicates that most positive predictions are accurate, which is critical for courses requiring critical reasoning where false positives can be misleading. Recall, or sensitivity, measures the proportion of actual positives correctly identified by the model. This metric is particularly important in the BIO-S11 course, where identifying all relevant instances is crucial due to the nature of biological research. The F1 score, calculated as the harmonic mean of precision and recall, provides a balance between accuracy and recall, making it especially useful when class distribution is unbalanced. The F1 score ensures that both false positives and false negatives are considered, as in QRPRO, where quantitative reasoning is essential. Accuracy, the most intuitive performance measure, represents the percentage of correct predictions (both true positives and true negatives) among the total number of cases evaluated. The results of this evaluation are shown in Tables II, III, and IV.

TABLE II. PRECISION, RECALL, F1SCORE AND SUPPORT FOR SVM MODEL

S. NO	Precision	Recall	F1 Score	Support
1.	0.94	0.92	0.93	340
2.	0.91	0.88	0.89	664
3.	0.96	0.98	0.97	1479

TABLE III. PRECISION, RECALL, F1SCORE AND SUPPORT FOR KNN MODEL

S.NO	Precision	Recall	F1 Score	Support
1.	0.89	0.91	0.90	340
2.	0.84	0.80	0.82	664
3.	0.94	0.95	0.95	1479

TABLE IV. PRECISION, RECALL, F1SCORE AND SUPPORT FOR RANDOM FOREST MODEL

S.NO	Precision	Recall	F1 Score	Support
1.	0.97	0.99	0.98	340
2.	1.00	0.98	0.99	664
3.	1.00	1.00	1.00	1479

J. Success Indicators

The insight breakdown of machine learning and deep learning models is important in understanding and enhancing their performance.[9] In this analysis, researchers adopt widely used evaluation metrics to evaluate the models. Researchers use these metrics: prediction accuracy and F1-score that give insight into model behavior when inputs vary. Generally, classification accuracy is the key criterion used to get the potency of machine learning and deep learning models. This

offers a simple way to determine how often the model's predictions match up with real outcomes. Additionally, confusion matrices are used to compare in more detail prediction accuracy and errors including correct classifications as well as misclassifications. Such a comprehensive approach helps ourselves make a strong evaluation of the classifiers hence identifying areas where researchers need improvement on hence refining them accordingly.

$$\text{Accuracy} = \frac{TP+TN}{TP+FP+TN+FN} \quad (1)$$

$$\text{Precision} = \frac{TP}{TP+FP} \quad (2)$$

$$\text{Recall} = \frac{TP}{TP+FN} \quad (3)$$

$$\text{F1-Score} = \frac{2\times Precision\times Recall}{Precision+Recall} \quad (4)$$

Taken together these metrics represent a complete view of model quality by taking into account both overall accuracy and handling positive cases

K. Comparitive Analysis

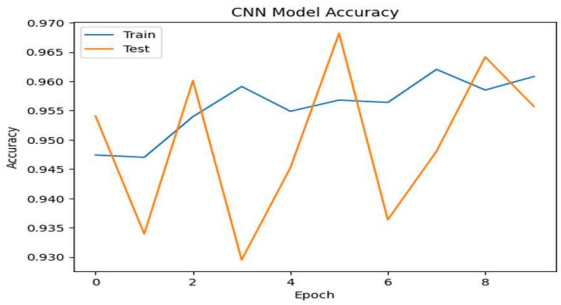


Fig. 4. Accuracy comparison between the train, test for CNN

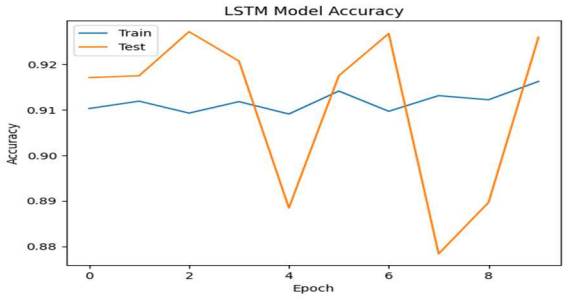


Fig. 5. Accuracy comparison between the train, test for LSTM

Methods like t-Distributed Stochastic Neighbor Embedding (t-SNE) help reduce dimensions allowing ourselves to see and assess high-dimensional data to spot patterns. The researcher test machine learning models such as Random Forest, Xgboost, SVM, and KNN as well as deep learning models like CNN and LSTM, to check how well they predict [1][2][3][4][5][6][7]. Researchers use measures like precision, recall, accuracy, and F1-score to do this. The results show that when Researchers mix advanced feature engineering with deep learning methods, these models get better at making predictions. This could play a key role in spotting college students at risk on and taking targeted steps to boost their academic results. The number of training samples used are 9928 and the number of test samples used are 2483.The below fig 4 and fig 5 shows the comparative analysis of CNN and LSTM models [13][14].

L. Conclusion

The document gives a complete analysis of predicting student academic performance using different data mining and machine learning methods. The study puts an emphasis on the need for detecting early any potential academic problems, which helps address such in a timely manner to improve results. Also, it helps to identify patterns and clusters that are closely related to students' performance trends. In addition of this study evaluates the effectiveness of various machine learning models and some deep learning modes including recently used XGBoost algorithm, K-Nearest Neighbors (KNN), Random Forest (RF), Support Vector Machine (SVM), and Logistic Regression and deep learning models like CNN and LSTM in predicting student achievement. Moreover, to enhance prediction accuracy deep learning techniques like Convolutional Neural Networks (CNN) and Long Short-Term Memory (LSTM) networks have been employed. The findings indicate that advanced feature engineering when combined with deep learning approaches enhances the models' predictiveness over more traditional methods.

V. FUTURE SCOPE

The future scope of this work is to consider advanced feature engineering based on real-time student engagement data, emotion analysis, and socio-demographics for improved predictions. There is also a potential for building hybrid models that incorporate traditional machine learning and reinforcement or transformer techniques to improve performance. At-risk students can be guaranteed improvement in outcomes through personalized learning interventions based on real-time data. However, it will be necessary to verify the scalability of the models across multiple cases in educational practice and address the interpretability and ethical issues related to AI based predictions.

REFERENCES

[1] Roy, K., & Farid, D. M. (2024). An Adaptive Feature Selection Algorithm for Student Performance Prediction. IEEE Access.

[2] Qin, K., Xie, X., He, Q., & Deng, G. (2023). Early Warning of Student Performance With Integration of Subjective and Objective Elements. IEEE Access.

[3] Alhazmi, E., & Sheneamer, A. (2023). Early predicting of students performance in higher education. IEEE Access, 11, 27579-27589.

[4] Liu, D., Zhang, Y., Zhang, J. U. N., Li, Q., Zhang, C., & Yin, Y. U. (2020). Multiple features fusion attention mechanism enhanced deep knowledge tracing for student performance prediction. IEEE Access, 8, 194894-194903.

[5] Butt, N. A., Mahmood, Z., Shakeel, K., Alfarhood, S., Safran, M., & Ashraf, I. (2023). Performance Prediction of Students in Higher Education Using Multi-Model Ensemble Approach. IEEE Access, 11, 136091-136108.

[6] Bujang, S. D. A., Selamat, A., Ibrahim, R., Krejcar, O., Herrera-Viedma, E., Fujita, H., & Ghani, N. A. M. (2021). Multiclass prediction model for student grade prediction using machine learning. IEEE Access, 9, 95608-95621.

[7] Alshanqiti, A., & Namoun, A. (2020). Predicting student performance and its influential factors using hybrid regression and multi-label classification. IEEE Access, 8, 203827-203844.

[8] Nabil, A., Seyam, M., & Abou-Elfetouh, A. (2021). Prediction of students' academic performance based on courses' grades using deep neural networks. IEEE Access, 9, 140731-140746

[9] Pelima, L. R., Sukmana, Y., & Rosmansyah, Y. (2024). Predicting university student graduation using academic performance and machine learning: a systematic literature review. IEEE Access.

[10] Vives, L., Cabezas, I., Vives, J. C., Reyes, N. G., Aquino, J., Condor, J. ´ B., & Altamirano, S. F. S. (2024). Prediction of Students' Academic Performance in the Programming Fundamentals Course Using Long Short-Term Memory Neural Networks. IEEE Access.

[11] Alamri, R., & Alharbi, B. (2021). Explainable student performance prediction models: a systematic review. IEEE Access, 9, 33132-33143.

[12] Sahlaoui, H., Nayyar, A., Agoujil, S., & Jaber, M. M. (2021). Predicting and interpreting student performance using ensemble models and shapley additive explanations. IEEE Access, 9, 152688-152703.

[13] Yaakub, T. N. T., Ahmad, W. R. W., Husaini, Y., & Burham, N. (2018, November). Influence factors in academic performance among electronics engineering student: Geographic background, mathematics grade and psycographic characteristics. In 2018 IEEE 10th International Conference on Engineering Education (ICEED) (pp. 30-33). IEEE.

[14] Sagala, T. N., Permai, S. D., Gunawan, A. A. S., Barus, R. O., & Meriko, C. (2022, December). Predicting Computer Science Student's Performance using Logistic Regression. In 2022 5th International Seminar on Research of Information Technology and Intelligent Systems (ISRITI) (pp. 817-821). IEEE.

[15] Sa, C. L., Hossain, E. D., & bin Hossin, M. (2014, November). Student performance analysis system (SPAS). In The 5th International Conference on Information and Communication Technology for The Muslim World (ICT4M) (pp. 1-6). IEEE.

ICIICS495.docx

ORIGINALITY REPORT

16%	11%	13%	5%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	"Proceeding of the International Conference on Computer Networks, Big Data and IoT (ICCBI - 2019)", Springer Science and Business Media LLC, 2020 Publication	2%
2	www.mdpi.com Internet Source	1%
3	github.com Internet Source	1%
4	Submitted to Coventry University Student Paper	1%
5	doctorpenguin.com Internet Source	1%
6	link.springer.com Internet Source	1%
7	Essa Alhazmi, Abdullah Sheneamer. "Early Predicting of Students Performance in Higher Education", IEEE Access, 2023 Publication	1%

8	journals.sagepub.com Internet Source	1 %
9	www.researchsquare.com Internet Source	1 %
10	Arvind Dagur, Karan Singh, Pawan Singh Mehra, Dhirendra Kumar Shukla. "Artificial Intelligence, Blockchain, Computing and Security", CRC Press, 2023 Publication	1 %
11	www.nature.com Internet Source	1 %
12	vitalflux.com Internet Source	<1 %
13	Submitted to University of Essex Student Paper	<1 %
14	Daniyar Sultan, Mateus Mendes, Aray Kassenkhan, Olzhas Akylbekov. "Hybrid CNN-LSTM Network for Cyberbullying Detection on Social Networks using Textual Contents", International Journal of Advanced Computer Science and Applications, 2023 Publication	<1 %
15	Submitted to Rochester Institute of Technology Student Paper	<1 %
	ijritcc.org	

16	Internet Source	<1%
17	cancerimagingjournal.biomedcentral.com Internet Source	<1%
18	researchportal.northumbria.ac.uk Internet Source	<1%
19	Jianfei Zhu, Chunzhi Yi, Baichun Wei, Chifu Yang, Zhen Ding, Feng Jiang. "The Muscle Fatigue's Effects on the sEMG-Based Gait Phase Classification: An Experimental Study and a Novel Training Strategy", Applied Sciences, 2021 Publication	<1%
20	Felix Indra Kurniadi, Meta Amalya Dewi, Dina Fitria Murad, Sucianna Ghadati Rabiha, Awanis Romli. "An Investigation into Student Performance Prediction using Regularized Logistic Regression", 2023 IEEE 9th International Conference on Computing, Engineering and Design (ICCED), 2023 Publication	<1%
21	iieta.org Internet Source	<1%
22	www.fastercapital.com Internet Source	<1%

23	"Proceedings of the 2nd International Conference on Emerging Technologies and Intelligent Systems", Springer Science and Business Media LLC, 2023 Publication	<1 %
24	Lecture Notes in Computer Science, 2015. Publication	<1 %
25	dokumen.pub Internet Source	<1 %
26	nicsforschoolleaders.tpdatscalecoalition.org Internet Source	<1 %
27	Muhamad Aqif Hadi Alias, Mohd Azri Abdul Aziz, Najidah Hambali, Mohd Nasir Taib. "Development of machine learning algorithms in student performance classification based on online learning activities", International Journal of Electrical and Computer Engineering (IJECE), 2024 Publication	<1 %
28	assets.researchsquare.com Internet Source	<1 %
29	clinical-practice-and-epidemiology-in-mental-health.com Internet Source	<1 %
30	ebin.pub Internet Source	<1 %

31	www.riverpublishers.com Internet Source	<1%

Exclude quotes	Off	Exclude matches	Off
Exclude bibliography	On		

Certificate-1

Centenary Celebrated Sharnbasveshwar Vidya Vardhak Sangha's

ಶರಣಬಸವ ವಿಶ್ವವಿದ್ಯಾಲಯ
SHARNBASVA UNIVERSITY

Kalaburagi – 585103, Karnataka - India

A State Private University approved by Govt. of Karnataka vide Notification No. ED 144 URC 2016 dated 29-07-2017,
Recognised by UGC under Section 2f vide No. F.8-29/2017 (CPP-I/PU), dated 20-12-2017 & AICTE, COA, PCI New Delhi

IEEE BANGALORE SECTION

IEEE NKSS

Certificate of Appreciation

This is to certify that Dr./Prof./Mr./Ms

Tumpala Mahith

has **Presented** Paper entitled

Unveiling Student Success: A Multifaceted Approach with Learning Coefficients and Beyond

for 2nd IEEE International Conference on
Integrated Intelligence and Communication Systems (ICIICS-2024)
organized by Sharnbasva University, Kalaburagi, 22-23 November, 2024

Dr. Lakshmi Patil Maka
Conference Chair and Convenor, ICIICS–2024
Dean, Sharnbasva University,
Kalaburagi,

Dr. Anilkumar G. Bidve
Vice-Chancellor,
Sharnbasva University,
Kalaburagi

Parama Poojya Dr. Sharnbaswappa Appaji
Chancellor,
Sharnbasva University,
Kalaburagi

Certificate-2

Centenary Celebrated Sharnbasveshwar Vidya Vardhak Sangha's

ಶರಣಬಸವ ವಿಶ್ವವಿದ್ಯಾಲಯ
SHARNBASVA UNIVERSITY

Kalaburagi – 585103, Karnataka - India

A State Private University approved by Govt. of Karnataka vide Notification No. ED 144 URC 2016 dated 29-07-2017,
Recognised by UGC under Section 2f vide No. F.8-29/2017 (CPP-I/PU), dated 20-12-2017 & AICTE, COA, PCI New Delhi

IEEE BANGALORE SECTION

IEEE KKSS

Certificate of Appreciation

This is to certify that Dr./Prof./Mr./Ms

Kethavath Ravi naik

has **Presented** Paper entitled

Unveiling Student Success: A Multifaceted Approach with Learning Coefficients and Beyond

for 2nd IEEE International Conference on
Integrated Intelligence and Communication Systems (ICIICS-2024)
organized by Sharnbasva University, Kalaburagi, 22-23 November, 2024

Dr. Lakshmi Patil Maka
Conference Chair and Convenor, ICIICS–2024
Dean, Sharnbasva University,
Kalaburagi,

Dr. Anilkumar G. Bidve
Vice-Chancellor,
Sharnbasva University,
Kalaburagi

Parama Poojya Dr. Sharnbaswappa Appaji
Chancellor,
Sharnbasva University,
Kalaburagi

Certificate-3

Centenary Celebrated Sharnbasveshwar Vidya Vardhak Sangha's

ಶರಣಬಸವ ವಿಶ್ವವಿದ್ಯಾಲಯ

SHARNBASVA UNIVERSITY

Kalaburagi – 585103, Karnataka - India

A State Private University approved by Govt. of Karnataka vide Notification No. ED 144 URC 2016 dated 29-07-2017,

Recognised by UGC under Section 2f vide No. F.8-29/2017 (CPP-I/PU), dated 20-12-2017 & AICTE, COA, PCI New Delhi

IEEE BANGALORE SECTION

IEEE MKSS

Certificate of Appreciation

This is to certify that Dr./Prof./Mr./Ms

Kolakani Raju

has **Presented** Paper entitled

Unveiling Student Success: A Multifaceted Approach with Learning Coefficients and Beyond

for 2nd IEEE International Conference on

Integrated Intelligence and Communication Systems (ICIICS-2024)

organized by Sharnbasva University, Kalaburagi, 22-23 November, 2024

Dr. Lakshmi Patil Maka
Conference Chair and Convenor, ICIICS-2024
Dean, Sharnbasva University,
Kalaburagi.

Dr. Anilkumar G. Bidve
Vice-Chancellor,
Sharnbasva University,
Kalaburagi

Parama Poojya Dr. Sharnbaswappa Appaji
Chancellor,
Sharnbasva University,
Kalaburagi