Ai-Based Student Dropout Prediction and Programming Skill Assessment System Using Machine Learning

R. Praveen[1], Singareddy Mahesh[2], S. Sai Kowshik[3], Mr. Arun[4], Mr. Senthilvela G[5]
[1,2,3]*Students, Department of Computer Science and Engineering, Dr. M.G.R. Educational and Research Institute, Chennai 600095, India*
[4]*Assistant Professor, Department of Computer Science and Engineering, Dr. M.G.R. Educational and Research Institute, Chennai 600095, India*
[5]*Professor, Department of Computer Science and Engineering, Dr. M.G.R. Educational and Research Institute, Chennai 600095, India*

Abstract*—Student dropout is a major challenge for educational institutions, affecting both academic outcomes and institutional performance. Early identification of students who are at risk can help educators provide timely support and improve retention rates. This research proposes a machine learning–based system that predicts student dropout risk while also evaluating programming skills through an online mock test platform. The system analyzes academic performance data, attendance records, and student engagement metrics to detect patterns related to academic difficulties. Machine learning algorithms such as Random Forest, Decision Tree, and Support Vector Machine are used to classify students based on dropout risk. In addition, programming mock tests are used to assess coding proficiency and categorize students into different skill levels. Based on these results, the system provides personalized learning recommendations. The proposed approach helps educational institutions monitor student performance more effectively and supports improved learning outcomes.

Index Terms*—Educational Data Mining, Machine Learning, Student Performance Prediction, Programming Skill Assessment, Learning Analytics.

I. INTRODUCTION

Student dropout is a major challenge for educational institutions because it affects both academic performance and student career development. Traditional monitoring systems mainly rely on manual analysis of grades, attendance, and assignment records, which may not identify at-risk students early enough for effective intervention. With the increasing availability of educational data, machine learning techniques can be used to analyze student performance and detect patterns related to academic difficulties.

At the same time, programming skills have become essential for students in technology-related fields, but many learners struggle to assess their current knowledge level and choose suitable learning resources. This research proposes a system that combines student dropout prediction with programming skill assessment.

By analyzing academic records and programming test results, the system can identify students at risk and provide personalized learning recommendations to improve academic success and skill development.

II. RELATED WORK

In recent years, the push to bridge the gap between virtual screens and genuine human connection has led to a surge in AI and computer vision research focused on student engagement. Innovative systems, such as those by Vargas et al. and the AttenQ tool, have introduced non-intrusive ways for instructors to "read the room" digitally, allowing them to adjust their teaching in real-time based on live engagement data. Technically, the field has moved toward sophisticated deep learning models—using everything from hybrid CNNs to Vision Transformers—to more accurately interpret subtle cues like facial orientation and head pose. While landmark- based monitoring and transfer learning have made real-time alerts possible, and researchers like Boca and Khan have mapped these tools to the broader shift of Education 4.0, a significant gap remains. Most current solutions are either too

narrow in focus, difficult to scale, or struggle to balance deep behavioral analysis with the vital need for student privacy. This highlights a clear need for a more thoughtful, integrated framework that doesn't just track data, but provides educators with the meaningful, multi-layered insights required to truly support their students—which is the core mission of this proposed system.

III. PROBLEM DEFINITION AND RESEARCH OBJECTIVES

Educational institutions often face difficulties in identifying students who may be at risk of dropping out. Traditional academic systems mainly store data such as grades, attendance, and assignment scores but do not analyze patterns that indicate potential academic problems. As a result, students who need support are often identified only after their performance has declined significantly. Additionally, many students struggle to evaluate their programming abilities and select appropriate learning resources. The objective of this research is to develop a system that uses machine learning techniques to predict student dropout risk and assess programming skills through mock tests. The system analyzes academic data and engagement indicators to classify students according to risk levels and programming proficiency. It also provides personalized learning recommendations to help students improve their academic performance and technical skills.

Research Objectives

The primary objective of this research is to develop a system that can analyze student academic performance and predict potential dropout risks using machine learning techniques. The study also aims to evaluate students' programming abilities through structured mock tests. By integrating academic data analysis with programming skill assessment, the system seeks to classify students according to their risk level and technical proficiency. Additionally, the proposed approach provides personalized learning recommendations to help students improve their programming skills and overall academic performance while supporting educational institutions in making informed decisions regarding student progress.

IV. PROPOSED SYSTEM ARCHITECTURE

The proposed system architecture is designed to analyze student academic performance and programming skills to identify potential dropout risks. The system follows a modular structure consisting of several interconnected components It begins with a user interface that allows students and administrators to interact with the platform, participate in programming tests, and view results. Academic and engagement data are collected through the data collection module and then processed in the preprocessing stage to ensure data quality. Machine learning algorithms are applied in the prediction module to analyze the processed data and classify students based on their dropout risk levels. At the same time, the programming assessment module evaluates students' coding abilities through mock tests. Finally, the recommendation module generates personalized learning suggestions based on the analysis results to support student improvement and academic success.

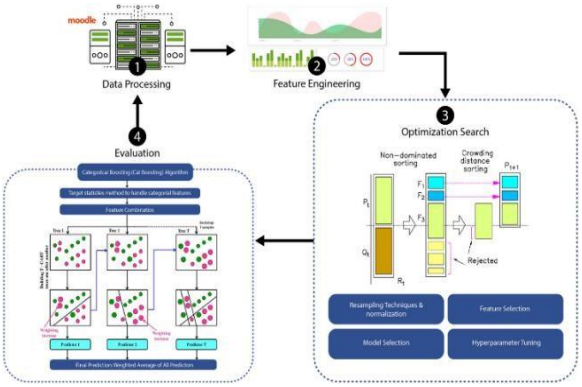

Fig 1: Architecture

V. METHODOLOGY

The methodology of the proposed system focuses on analyzing student academic performance and programming skills using a structured machine learning approach. The process is organized into several stages that allow the system to collect, process, and analyze data in order to generate meaningful predictions about student performance.

The first stage involves data collection, where information related to student academic activities is gathered from institutional records and learning platforms. This data typically includes attendance percentage, assignment scores, examination results, and course completion status. In addition to academic

information, the system also records programming mock test results, which are used to evaluate students' coding knowledge.

After collecting the data, the next step is data preprocessing. In this stage, the dataset is cleaned and prepared for analysis. Missing values are handled, inconsistent data entries are corrected, and categorical variables are converted into numerical form. Feature normalization is also performed to ensure that the dataset is balanced and suitable for machine learning models.

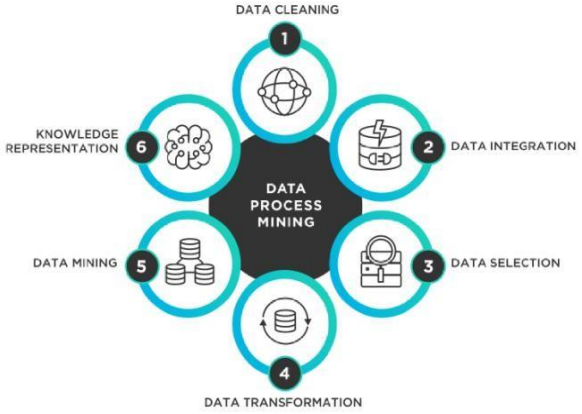

Fig 2: Methodology

Once the dataset has been prepared, feature extraction and selection are performed. Important attributes that influence student performance are identified and selected for model training. These features may include attendance rate, assignment completion percentage, engagement level, and programming test scores. Selecting relevant features helps improve the accuracy and efficiency of the prediction model.

The processed dataset is then used in the model training stage, where machine learning algorithms are applied to learn patterns from the data. Algorithms such as Random Forest, Decision Tree, and Support Vector Machine are used to classify students based on their academic performance and engagement indicators. These algorithms analyze the relationships between different features and predict whether a student is at risk of dropping out.

Alongside dropout prediction, the system also performs programming skill assessment. Students participate in mock programming tests that evaluate their understanding of programming concepts and problem-solving abilities. Based on their scores, students are categorized into skill levels such as beginner, intermediate, or advanced.

Finally, the results generated by the prediction models and skill assessment module are used to provide personalized recommendations. These recommendations guide students toward suitable learning resources and help educators identify students who may require additional academic support.

Overall, the proposed methodology provides a systematic approach for analyzing student data, predicting potential dropout risks, and improving programming skill development through targeted recommendations.

VI. DATASET DESCRIPTION

The dataset used in this study is designed to represent different aspects of student academic performance and learning behavior. It contains information collected from institutional academic records as well as programming mock test results conducted within the proposed system. By combining these different sources of information, the dataset provides a comprehensive view of student progress and engagement. The dataset mainly includes three categories of data: academic performance data, student engagement data, and programming assessment data. Academic performance data reflects a student's overall progress in their courses and includes attributes such as attendance percentage, assignment scores, and examination results. These indicators help measure the consistency of a student's academic performance throughout the semester. Student engagement data represents how actively a learner participates in academic activities and interacts with learning platforms. This type of data may include the frequency of platform logins, time spent on learning materials, and the completion of assignments or activities. Engagement indicators are important because students who actively participate in learning activities generally show better academic outcomes. The dataset also includes programming assessment data, which is obtained from mock tests conducted through the system. These tests evaluate a student's understanding of programming concepts, logical reasoning, and problem-solving ability in different programming languages. The score obtained in these tests is used to determine the student's programming proficiency level.

Before using the dataset for machine learning analysis, several preprocessing steps are performed. These

include removing incomplete records, handling missing values, converting categorical attributes into numerical values, and normalizing feature values. These steps ensure that the dataset is clean, consistent, and suitable for training predictive models. Overall, the dataset provides a structured representation of student academic performance and programming ability. By analyzing this dataset using machine learning techniques, the system can identify patterns related to student success and generate predictions regarding potential dropout risks.

VII. ALGORITHMS AND WORKFLOW

The proposed system uses machine learning algorithms to analyze student academic data and programming test results in order to predict dropout risk and evaluate programming skills. The workflow of the system follows a structured sequence of steps, beginning with data collection and ending with the generation of predictions and learning recommendations.

Algorithms Used

To perform the prediction task, the system applies several widely used machine learning algorithms that are effective for classification problems.

Decision Tree

The Decision Tree algorithm is used to classify students based on different attributes such as attendance, assignment scores, and programming test results. It works by dividing the dataset into smaller subsets according to decision rules derived from the input features. This method is easy to interpret and helps visualize how predictions are made.

Random Forest

Random Forest is an ensemble learning algorithm that combines multiple decision trees to improve prediction accuracy. Each tree is trained on a different subset of the dataset, and the final prediction is obtained through majority voting. This approach reduces overfitting and provides more reliable results when analyzing complex educational datasets.

Support Vector Machine (SVM)

Support Vector Machine is another classification algorithm used to separate students into different categories based on their performance indicators. It identifies an optimal boundary that distinguishes between different classes, such as students with low, medium, or high dropout risk.

These algorithms analyze patterns within the dataset and help identify relationships between academic performance, engagement behavior, and programming test results.

System Workflow

The overall workflow of the system is designed to process student data step by step in order to produce meaningful insights. The process begins with the collection of student academic and engagement data. This information includes attendance records, assignment scores, and course completion details. Once the data is collected, it is passed through a preprocessing stage where the dataset is cleaned and prepared for analysis. In this stage, missing values are handled, inconsistent records are corrected, and features are normalized. After preprocessing, important features that influence student performance are selected for model training. These features are then used by machine learning algorithms to learn patterns from the dataset.

VIII. IMPLEMENTATION DETAILS

The proposed system is implemented using a combination of programming tools, machine learning libraries, and web based technologies. The goal of the implementation is to create a platform that can collect student data, analyze it using machine learning techniques, and provide meaningful insights about student performance and programming skills.

The development of the system is primarily carried out using Python, which is widely used for machine learning and data analysis. Python provides a rich set of libraries that simplify the process of data processing, model development, and system integration. Libraries such as NumPy and Pandas are used for handling and manipulating datasets, while Scikit-learn is used to implement machine learning algorithms including Decision Tree, Random Forest, and Support Vector Machine.

For processing and analyzing student data, the system uses data preprocessing techniques such as data cleaning, normalization, and feature selection. These steps ensure that the dataset is suitable for training

machine learning models and improves the accuracy of predictions.

The user interface of the system is implemented using web technologies, allowing students and administrators to interact with the platform through a browser. Students can log into the system, take programming mock tests, and view their results. Administrators can access analytical dashboards that display student performance and dropout prediction results.

The programming assessment module is integrated into the platform to evaluate students' coding knowledge. The module generates mock test questions in various programming languages and records student responses. The results of these tests are automatically evaluated and stored in the system database.

A central server component manages communication between different modules of the system. It processes incoming data from users, performs machine learning analysis, and stores results in the database. The server also generates reports and visualizations that help instructors understand student engagement and performance trends.

Overall, the implementation integrates machine learning algorithms, data processing techniques, and web-based interfaces into a unified system. This design ensures that the platform is efficient, scalable, and capable of providing valuable insights for improving student learning outcomes.

IX. EXPERIMENTAL SETUP

The experimental setup was designed to evaluate the performance and effectiveness of the proposed system in predicting student dropout risk and assessing programming skills. The experiments were conducted using a structured dataset containing student academic records and programming mock test results. The objective of the experiment was to analyze how accurately the machine learning models could identify students who may be at risk and evaluate their programming proficiency.

The system was implemented in a standard computing environment using a computer equipped with a modern processor and sufficient memory to handle data processing and model training tasks. The experiments were performed using Python along with commonly used machine learning libraries for data analysis and model development.

To conduct the experiment, student data was first collected and organized into a dataset containing attributes such as attendance percentage, assignment scores, course completion information, and programming test results. Before training the models, the dataset underwent preprocessing steps including cleaning the data, handling missing values, and normalizing numerical features. These steps ensured that the dataset was consistent and suitable for machine learning analysis.

After preprocessing, the dataset was divided into two parts: a training dataset and a testing dataset. The training dataset was used to train the machine learning models, while the testing dataset was used to evaluate the performance of the trained models. Algorithms such as Decision Tree, Random Forest, and Support Vector Machine were applied to classify students based on their dropout risk levels.

The performance of the models was evaluated using common classification metrics such as accuracy, precision, recall, and F1score. These metrics helped determine how effectively the models could identify students who may face academic difficulties.

In addition to dropout prediction, the experimental setup also included programming mock tests that evaluated students' coding knowledge. The results of these tests were used to classify students into different programming skill levels. By combining academic data analysis with programming skill evaluation, the system was able to provide a more comprehensive assessment of student performance.

Overall, the experimental setup allowed the researchers to assess the accuracy, reliability, and practical applicability of the proposed system in real educational environments.

X. RESULTS AND PERFORMANCE ANALYSIS

The performance of the proposed system was evaluated by analyzing its ability to predict student dropout risk and assess programming skills based on the collected dataset. The experiments focused on measuring how accurately the machine learning models could classify students according to their academic performance and engagement indicators.

The system uses machine learning algorithms such as Random Forest, Decision Tree, and Support Vector Machine to analyze patterns in student data. These algorithms examine factors including attendance,

assignment completion, and engagement levels to determine whether a student may be at risk of dropping out. The experimental evaluation showed that the prediction models achieved an overall accuracy ranging between 85% and 90%, depending on the quality of the dataset and selected features.

To measure the effectiveness of the prediction models, several evaluation metrics were used:

- Accuracy – measures the overall percentage of correct predictions made by the model.
- Precision – indicates how many of the predicted positive cases are actually correct.
- Recall – measures the ability of the model to correctly identify students who are truly at risk.
- F1-score – provides a balanced measure that combines precision and recall.

The accuracy metric is calculated using the following formula:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

Where:

- TP (True Positive) – correctly predicted at-risk students
- TN (True Negative) – correctly predicted non-risk students
- FP (False Positive) – students incorrectly predicted as at risk
- FN (False Negative) – students incorrectly predicted as not at risk

In addition to dropout prediction, the system evaluates students' programming skills through mock tests. These tests measure knowledge of programming concepts, syntax, and problem-solving ability. Based on the results, students are categorized into Beginner, Intermediate, or Advanced skill levels.

The combination of dropout prediction and programming skill assessment allows the system to provide a more complete understanding of student performance. Educators can use the generated insights to identify students who may require additional academic support, while students can receive personalized recommendations to improve their programming skills and overall learning outcomes.

Overall, the experimental results demonstrate that the proposed system can effectively analyze student data, predict academic risks, and support better decision-making in educational environments.

XI. DISCUSSION

The results obtained from the experimental evaluation indicate that the proposed system is capable of effectively analyzing student academic performance and predicting potential dropout risks. By applying machine learning algorithms to educational data, the system can identify patterns that may not be easily visible through traditional manual monitoring methods. This enables educational institutions to detect students who may require additional academic support at an earlier stage.

The integration of programming skill assessment with dropout prediction also provides a more comprehensive understanding of student learning behavior. Programming mock tests allow the system to evaluate students' problem-solving abilities and technical knowledge, which are important factors for students in technology-related programs. The classification of students into different skill levels helps instructors understand the strengths and weaknesses of individual learners.

The experimental results demonstrate that machine learning models such as Random Forest, Decision Tree, and Support Vector Machine can achieve reliable prediction accuracy when applied to student performance data. The ability of these models to analyze multiple features simultaneously allows the system to provide more accurate insights compared to traditional academic monitoring approaches.

However, certain limitations should also be considered. The accuracy of prediction models depends largely on the quality and completeness of the dataset used for training. If the dataset contains incomplete or inconsistent information, it may affect the performance of the model. Additionally, student behavior and learning patterns can vary across different institutions and academic environments, which may require adjustments to the model for broader applications.

Overall, the proposed system demonstrates the potential of machine learning techniques in improving student monitoring and supporting data-driven decision making in educational institutions. By combining academic data analysis with programming skill evaluation, the system provides a practical

approach for enhancing student engagement and academic success.

XII. CONCLUSION

This research presented a system designed to predict student dropout risk while also evaluating programming skills using machine learning techniques. The proposed approach analyzes multiple factors related to student performance, including academic records, engagement indicators, and programming mock test results. By combining these different sources of information, the system provides a more comprehensive understanding of student learning behavior.

Machine learning algorithms such as Decision Tree, Random Forest, and Support Vector Machine were applied to analyze the dataset and classify students based on their potential dropout risk. The experimental results demonstrate that these models can effectively identify patterns in student performance and achieve reliable prediction accuracy. In addition to dropout prediction, the programming assessment module helps evaluate students' coding abilities and classify them into different skill levels.

The integration of dropout prediction and programming skill assessment allows educational institutions to identify students who may require additional academic support while also helping learners understand their programming proficiency. The system also provides personalized learning recommendations that guide students toward improving their technical skills and academic progress.

Overall, the proposed system demonstrates how machine learning and educational data analysis can be used to improve student monitoring and support informed decision making in educational environments. By providing early insights into student performance and skill development, the system contributes to enhancing learning outcomes and reducing the risk of student dropout.

XIII. FUTURE WORK

Although the proposed system provides an effective approach for predicting student dropout risk and evaluating programming skills, several improvements can be explored in future research to enhance its capabilities. One possible improvement is the use of advanced machine learning or deep learning models that can analyze larger and more complex educational datasets. These models may help improve prediction accuracy and identify deeper patterns in student behavior.

Another potential enhancement is the integration of a real time coding environment within the platform. Instead of relying only on multiple-choice programming questions, students could write and execute code directly in the system. This would allow a more accurate evaluation of their practical programming abilities.

The system can also be expanded to include additional programming languages and learning resources so that students from different technical backgrounds can benefit from the platform. Incorporating more datasets from multiple institutions could further improve the reliability and generalization of the prediction models.

In addition, future versions of the system could include mobile application support, enabling students to access programming tests, learning materials, and performance feedback through smartphones or tablets. Finally, integrating explainable machine learning techniques would help educators understand how predictions are generated, improving transparency and trust in the system.

These improvements could make the system more scalable, accurate, and useful for supporting personalized learning and student success in educational environments.

REFERENCES

[1] C. Romero and S. Ventura, "Educational data mining: A review of the state of the art," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 40, no. 6, pp. 601–618, 2010.

[2] S. Kotsiantis, C. Pierrakeas, and P. Pintelas, "Predicting students' performance in distance learning using machine learning techniques," *Applied Artificial Intelligence*, vol. 18, no. 5, pp. 411–426, 2004.

[3] M. Dekker, M. Pechenizkiy, and J. Vleeshouwers, "Predicting students drop out: A case study," in *Proc. Int. Conf. Educational Data Mining*, 2009.

[4] S. Lakkaraju, E. Aguiar, C. Shan, D. Miller, N. Bhanpuri, and R. Ghani, "A machine learning framework to identify students at risk of adverse

academic outcomes," in *Proc. 21st ACM SIGKDD Int. Conf.*, 2015.

[5] J. Han, M. Kamber, and J. Pei, *Data Mining: Concepts and Techniques*, 3rd ed. Burlington, MA, USA: Morgan Kaufmann, 2011.

[6] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning*. New York, NY, USA: Springer, 2009.

[7] Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. Cambridge, MA, USA: MIT Press, 2016.

[8] K. Murphy, *Machine Learning: A Probabilistic Perspective*. Cambridge, MA, USA: MIT Press, 2012.

[9] C. Romero, S. Ventura, and M. Pechenizkiy, *Handbook of Educational Data Mining*. Boca Raton, FL, USA: CRC Press, 2010.

[10] J. Bergstra and Y. Bengio, "Random search for hyper-parameter optimization," *Journal of Machine Learning Research*, vol. 13, pp. 281–305, 2012.

[11] L. Breiman, "Random forests," *Machine Learning*, vol. 45, no. 1, pp. 5–32, 2001.

[12] V. Vapnik, *Statistical Learning Theory*. New York, NY, USA: Wiley, 1998.

[13] T. Mitchell, *Machine Learning*. New York, NY, USA: McGraw-Hill, 1997.

[14] R. S. Baker and K. Yacef, "The state of educational data mining in 2009: A review and future visions," *Journal of Educational Data Mining*, vol. 1, no. 1, pp. 3–17, 2009.

[15] P. Cortez and A. Silva, "Using data mining to predict secondary school student performance," in *Proc. Int. Conf. Educational Data Mining*, 2008.