

A Stacked Classifier Model for Enhanced Student Performance Prediction in E-Learning Environments

Sajithunisa Hussain¹, Jayachandran Jeyachidra²

^{1,2}Department of Computer Applications, Periyar Maniammai Institute of Science & Technology, Vallam-613403, Thanjavur, Tamil Nadu, India

Article Info

Article history:

Received Nov 1, 2025

Revised Feb 2, 2026

Accepted Mar 8, 2026

Keywords:

Machine learning

Stacked Classifier Model

Pipeline

Multi-Layer Perceptron

Random Forest

Support Vector Machine

ABSTRACT

The rapid growth of e-learning platforms has resulted in an enormous amount of student interaction data, creating opportunities to anticipate learning outcomes and implement timely interventions. In this research, a Stacked Classifier Model (SCM) is introduced to predict student performance using e-learning reaction data obtained from a Kaggle repository. The SCM employs a hierarchical ensemble approach by combining several base classifiers—K-Nearest Neighbors (KNN), Decision Tree (DT), Multi-Layer Perceptron (MLP), Random Forest (RF), Support Vector Machine (SVM), and Radial Basis Function (RBF) networks—to capitalize on their respective strengths while compensating for individual limitations. The dataset underwent careful preprocessing, including imputation, encoding, feature normalization, and temporal aggregation, to ensure the classifiers received high-quality input. Evaluation results indicate that the SCM outperforms each base model individually, demonstrating its capability to capture complex behavioral patterns in e-learning contexts. Overall, this study highlights the effectiveness of ensemble learning techniques in educational data mining, offering a solid foundation for adaptive learning, personalized interventions, and enhanced academic performance.

Copyright © 2026 Institute of Advanced Engineering and Science.

All rights reserved.

Corresponding Author:

Sajithunisa Hussain

Department of Computer Applications,

Periyar Maniammai Institute of Science & Technology, Vallam-613403,

Thanjavur, Tamil Nadu, India Email: sajithunisa5017@pmu.edu

1. INTRODUCTION

The rapid expansion of digital learning platforms has fundamentally transformed contemporary education by enabling flexible access to learning resources, personalized instruction, adaptive learning pathways, and continuous monitoring of student progress [1], [2], [3], [4]. The widespread adoption of e-learning—accelerated by the global shift toward remote education—has generated large volumes of student interaction data, including assessment attempts, discussion forum participation, assignment submissions, and engagement-related activities [5], [6], [7], [8]. These data provide valuable opportunities for understanding learner behavior and predicting academic outcomes [9], [10], [11]. However, their heterogeneous structure, high dimensionality, and temporal complexity pose significant challenges for effective predictive modeling [12], [13], [14].

Accurate prediction of student performance is essential for supporting early identification of at-risk learners, enabling timely interventions, and improving instructional design [15], [16], [17]. Prior studies have demonstrated that performance prediction can assist educators in optimizing course delivery and enhancing student retention and achievement [18], [19], [20], [21], [22]. As a result, student performance prediction has become a central research theme within educational data mining and learning analytics [23], [24].

Traditional machine learning classifiers—including k-Nearest Neighbors, Decision Trees, Random Forests, Support Vector Machines, and shallow neural networks—have been widely adopted for this task [25], [26], [27], [28]. Although these methods demonstrate reasonable predictive capability, they often struggle to capture complex nonlinear relationships and temporal dependencies inherent in student interaction data [29], [30]. Furthermore, issues such as overfitting, sensitivity to noise, and limited generalization across datasets restrict their effectiveness in dynamic e-learning environments [11], [13], [14], [17].

To overcome these limitations, ensemble learning techniques have gained increasing attention in recent years [10], [16]. In particular, stacked classifier models (SCMs) integrate multiple base learners within a hierarchical architecture, where the outputs of individual classifiers are combined through a meta-classifier [18], [24]. Prior research has shown that stacking-based ensembles consistently outperform individual models by leveraging complementary learning capabilities and enhancing robustness to noise and class imbalance [18], [24], [25]. These findings highlight the suitability of SCMs for modeling complex and heterogeneous educational datasets [3], [16].

In parallel, sequence-based deep learning approaches, such as Long Short-Term Memory networks, have been applied to capture temporal patterns in student behavior, including engagement progression and retention risk [30]. While effective for sequential data, such models often face challenges when applied to heterogeneous tabular features, underscoring the need for hybrid and ensemble frameworks that combine multiple learning paradigms [1], [7], [22].

Despite these advances, several research gaps remain. Existing studies primarily emphasize predictive accuracy, with limited attention to model interpretability and reproducibility, which constrains real-world applicability [14], [23]. Moreover, few works integrate temporal features with heterogeneous tabular data within a unified stacked framework [18], [30]. Automated preprocessing and feature engineering pipelines are also underexplored, limiting generalizability across datasets [12], [17], [21], [28]. Finally, comprehensive comparative evaluations of stacked models against a full spectrum of individual classifiers under consistent experimental settings remain scarce [16], [23].

Motivated by these limitations, this study proposes a robust Stacked Classifier Model for student performance prediction in e-learning environments. The objectives of this research are threefold: (i) to design a hierarchical ensemble framework that effectively captures complex student interaction patterns, (ii) to conduct a systematic comparative evaluation against widely used standalone classifiers, and (iii) to examine the impact of structured preprocessing and feature engineering on predictive performance. The proposed approach aims to enhance both the accuracy and reliability of student performance prediction, thereby supporting data-driven decision-making in digital education systems.

Unlike prior studies that either rely on single classifiers or generic ensemble techniques, this work introduces a pipeline-driven Stacked Classifier Model (SCM) that systematically integrates heterogeneous base learners with out-of-fold meta-learning to ensure unbiased training and robust generalization. The novelty lies in the joint optimization of preprocessing, classifier diversity, and stacking strategy, enabling the model to effectively capture both local behavioral patterns and global performance trends in e-learning data. From a practical perspective, this approach provides more reliable early identification of at-risk students, supporting timely interventions and data-driven decision-making in real-world educational platforms.

2. RESEARCH METHOD

In this section, you should explain how the research was conducted, including research design. The proposed approach employs a Stacked Classifier Model (SCM) to forecast student performance using interaction data from e-learning platforms. In this study, Logistic Regression is employed as the Level-1 meta-classifier, serving as the blender that aggregates probability outputs from all base classifiers. The SCM is structured as a two-level ensemble, where several base classifiers at Level 0 produce intermediate predictions that are subsequently integrated and fed into a Level 1 meta-classifier, improving both prediction accuracy and model robustness. The methodology consists of the following key steps:

2.1. Dataset Description

The experiments were conducted using the Kaggle E-Learning Student Reactions dataset, which is publicly available at: <https://www.kaggle.com/datasets/marlonferrari/elearning-student-reactions>

The dataset contains 3,500+ student interaction records, where each instance corresponds to a learner's response to online educational content. The dataset is designed to capture behavioral engagement and reaction patterns in e-learning environments.

The key features of the dataset include:

- **Demographic attributes:** age group, gender, educational level
- **Engagement features:** interaction frequency, content engagement intensity
- **Behavioral indicators:** reaction category, feedback patterns
- **Target variable:** student performance status (binary classification)

A comprehensive preprocessing pipeline was applied prior to model training. Missing numerical values were handled using mean imputation, while categorical attributes were imputed using the mode. Categorical features were transformed using one-hot encoding, and numerical features were standardized using z-score normalization to ensure consistent feature scaling. To maintain class balance during evaluation, stratified sampling was employed. The dataset was divided into 70% training data and 30% testing data.

2.2. Data Preprocessing

To ensure high-quality input for the SCM, the following preprocessing steps were implemented:

2.2.1. Handling Missing Values:

- Numerical features with missing values were imputed using column-wise mean imputation:

$$x_i^{imputed} = \frac{\sum_{j=1}^n x_{i,j}}{n} \text{ for } x_{i,j} \text{ non missing values}$$
- Categorical features were imputed using the **mode** of the column.

2.2.2. Encoding Categorical Variables:

Categorical features were transformed into numeric representations using One-Hot Encoding, converting each category into a binary vector suitable for input into classifiers.

2.2.3. Feature Scaling:

Numerical features were standardized using z-score normalization to ensure zero mean and unit variance:

$x'_i = \frac{x_i - \mu_i}{\sigma_i}$	(1)
---------------------------------------	-----

Where σ_i and μ_i are the mean and standard deviation of feature i .

2.2.4. Data Splitting:

The dataset was divided using stratified sampling to preserve class distribution:

- Training Set: 80%
- Validation Set: 10%
- Testing Set: 10%

2.3. Proposed Stacked Classifier Model

The SCM consists of two hierarchical levels a:

2.3.1 Level 0 – Base Classifiers:

The first level comprises six diverse classifiers that capture different patterns in the dataset:

1. K-Nearest Neighbors (KNN): Exploits local similarity in feature space.
2. Decision Tree (DT): Captures hierarchical decision rules.
3. Multi-Layer Perceptron (MLP): Models non-linear feature interactions.
4. Random Forest (RF): An ensemble of decision trees to reduce overfitting.
5. Support Vector Machine (SVM): Maximizes margin between classes.
6. Radial Basis Function (RBF) Network: Handles non-linear relationships with kernel-based transformations.

Each base classifier is independently trained on the preprocessed training data and outputs a predicted probability of student performance. Figure 1 shows the architecture of SCM.

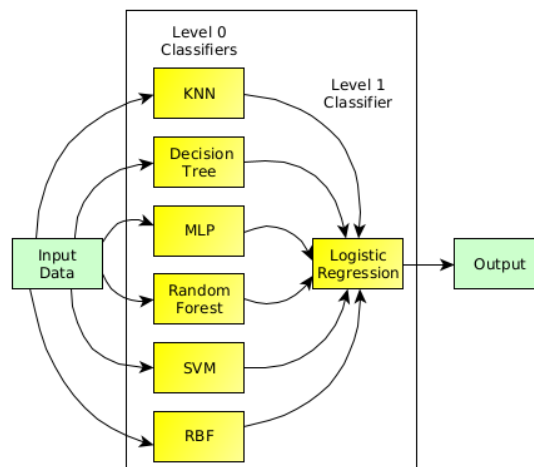


Figure 1. Architecture of SCM

The Stacked Classifier Model (SCM) is constructed to harness the complementary strengths of six different base classifiers, with their predictions consolidated by a meta-classifier. This architecture avoids

dependence on any single decision boundary, instead combining multiple perspectives to enhance robustness and predictive accuracy for student performance. The underlying mathematical framework provides a clear representation of how each base model contributes to the overall ensemble pipeline.

Let the dataset be denoted as:

$$D = \{(x_i, y_i)\}_{i=1}^N \quad (2)$$

where $x_i \in R^d$ represents the feature vector of student i , $y_i \in \{0,1\}$ denotes the binary performance label (e.g., success/failure), and N is the total number of students.

Each base classifier C_j (where $j \in \{1, \dots, 6\}$) is trained on the dataset D . The output of each classifier is not a hard class label but a probability estimate of success:

$$p_{ij} = C_j(x_i), \quad 0 \leq p_{ij} \leq 1$$

Where p_{ij} denotes the probability predicted by classifier C_j for student i . These outputs form the meta-features for the next stage.

Now let's break down the contributions of each classifier in this work:

K-Nearest Neighbors (KNN):

Uses the similarity between x_i and its nearest neighbors. Its probability estimate is:

$$p_{i,KNN} = \frac{1}{K} \sum_{x_j \in N_k(x_i)} 1(y_j = 1) \quad (3)$$

Where $N_k(x_i)$ denotes the set of k -nearest neighbors of x_i . KNN ensures local cluster patterns are incorporated.

Decision Tree (DT):

DT splits the dataset using entropy or Gini impurity. For node t , impurity is defined as:

$$I(t) = - \sum_{c \in \{0,1\}} p(c | t) \log p(c | t) \quad (4)$$

Probabilities from leaf nodes yield $p_{i,DT}$. DT captures interpretable rules.

Multi-Layer Perceptron (MLP):

Given hidden representations:

$$h^1 = f(W^1 x_i + b^1), h^2 = f(W^2 h^1 + b^2) \quad (5)$$

Where $f(\cdot)$ is ReLU activation. The final probability is:

$$p_{i,MLP} = \sigma(W^3 h^2 + b^3) \quad (6)$$

MLP contributes by modeling non-linear interactions in student behaviors.

Random Forest (RF):

An ensemble of decision trees where each tree T_m produces a probability estimate P_{im} . The RF aggregates these via:

$$p_{i,RF} = \frac{1}{M} \sum_{m=1}^M P_{im} \quad (7)$$

where M is the number of trees. RF reduces variance and ensures stable generalization.

Support Vector Machine (SVM):

For linear kernel:

$$f(x_i) = w^\top x_i + b$$

The probability is obtained using Platt scaling:

$$p_{i,SVM} = \frac{1}{1 + \exp(Af(x_i) + B)} \quad (8)$$

where A and B are scaling parameters. SVM maximizes margin, adding robust separation.

Radial Basis Function (RBF) Network:

The hidden transformation is:

$$h_j(x_i) = \exp\left(-\frac{\|x_i - \mu_j\|^2}{2\sigma_j^2}\right) \quad (9)$$

The output probability is then:

$$p_{i,RBF} = \sigma\left(\sum_{j=1}^k w_j h_j(x_i) + b\right) \quad (10)$$

RBF ensures smooth handling of non-linear transitions in student performance.

2.3.2 Meta-Feature Construction

After training, the six base classifiers produce a meta-feature vector for each student:

$$z_i = [p_{i,KNN}, p_{i,DT}, p_{i,MLP}, p_{i,RF}, p_{i,SVM}, p_{i,RBF}] \quad (11)$$

Thus, the transformed dataset for Level 1 becomes:

$$D = \{(z_i, y_i)\}_{i=1}^N \quad (12)$$

This dataset summarizes the confidence of each base model for every instance, serving as input to the meta-classifier.

2.3.3 Level 1 – Meta-Classifier (Logistic Regression)

In the second stage of the stacking framework, the predictions from the six base classifiers are aggregated into a meta-feature vector for each student instance. A logistic regression (LR) model functions as the meta-classifier, learning to assign optimal weights to the probability outputs of the base classifiers to produce the final prediction.

The prediction function of LR is defined as:

$$\hat{y}_i = \sigma \left(\beta_0 + \sum_{j=1} \beta_j p_{i,j} \right) \quad (13)$$

Where,

$\hat{y}_i$ is the predicted probability of student i being classified as successful,

$p_{i,j}$ is the probability estimate produced by the j^{th} base classifier for student i ,

β_0 is the bias term,

β_j is the learned weight for the j^{th} base classifier,

The training objective of logistic regression is to minimize the binary cross-entropy loss, given as:

$$L = -\frac{1}{N} \sum_{i=1}^N [y_i \log(y_i) + (1 - y_i) \log(1 - y_i)] \quad (14)$$

where N is the number of samples and $y_i \in \{0,1\}$ is the ground truth label for student i .

By minimizing this loss, the logistic regression meta-classifier effectively determines how much **trust** should be placed in the predictions of each base classifier. Stronger classifiers receive higher weights β_j , while weaker ones are down-weighted, resulting in a robust ensemble that leverages complementary strengths across all models.

This weighted combination ensures that local neighborhood patterns (KNN), hierarchical decision rules (DT), non-linear feature learning (MLP), variance reduction (RF), margin maximization (SVM), and kernel-based transformations (RBF) are systematically integrated into a single, optimized decision function. Logistic Regression was selected as the meta-classifier due to its stability, interpretability, and low variance, which are critical properties for stacking frameworks where overfitting at the meta-level can severely degrade performance. By learning linear weights over calibrated probability outputs, Logistic Regression effectively acts as a reliable blender, emphasizing strong base learners while suppressing weaker or redundant predictions. This choice ensures robust generalization and reproducibility, particularly when stacking heterogeneous classifiers trained on high-dimensional and noisy e-learning data.

Algorithm: Stacking Classifier Model for Result Prediction

Input: $X_{train} \in R^{n_{samples} \times n_{features}}$, $y_{train} \in \{0,1\}^{n_{samples}}$

Base Learners: $\{f_1, f_2, \dots, f_m\}$

Meta Learner: g

Output: $\hat{y}_{pred}$

1. Train Base Learners:

$$f_j \leftarrow f_j(X_{train}, y_{train}), \forall j \in \{1, 2, \dots, m\}$$

2. Generate Base Learner Predictions:

$$P = \{p_{ij} \mid p_{ij} = f_j(X_i), \forall i \in \{1, \dots, n_{samples}\}, j \in \{1, \dots, m\}\}$$

$$P \in R^{n_{samples}}$$

3. Train Meta-Learner:

$$g \leftarrow g(P, y_{train})$$

4. Prediction on New Data:

$$P_j^* = f_j(X_{new}), \forall j \in \{1, 2, \dots, m\}$$

5. Final Prediction:

$$P^* = [P_1^*, P_2^*, P_m^*]$$

$$\hat{y}_{pred} = g(P^*)$$

The proposed stacked classifier algorithm systematically combines multiple base learners with a meta-learner to improve the accuracy of student outcome predictions. The process begins with training a set of diverse base classifiers on the training dataset (X_{train}, y_{train}) , where each model independently captures unique decision boundaries and feature interactions. After training, each base learner generates probability scores for all training instances, forming a prediction matrix $P \in R^{n_{samples}}$, with rows representing samples and columns representing outputs from the mmm classifiers. This matrix encapsulates complementary views of the data, capturing both linear and non-linear relationships across models. The meta-learner, typically a logistic regression model, is then trained on this prediction matrix rather than the original input features, allowing it to learn the optimal weighting of the base classifiers' outputs. During testing, new instances are first evaluated by all base learners to generate a prediction vector P^* , which is then fed into the meta-learner to produce the final prediction $\hat{y}_{pred}$. This hierarchical structure reduces individual model bias, mitigates variance, and leverages classifier diversity, resulting in improved generalization performance.

2.4 Training Procedure

The training of the SCM was conducted in two stages. In the first stage, all Level-0 base classifiers were trained independently using the preprocessed training data. Hyperparameters for each classifier were optimized using GridSearchCV with 10-fold cross-validation. To train the Level-1 meta-classifier, out-of-fold probability predictions were generated from each base learner. These predictions were used as meta-features to train the logistic regression model, ensuring that the meta-classifier only receives predictions from unseen data and preventing information leakage. During testing, each base classifier generated probability outputs for the test instances, which were concatenated and passed to the meta-classifier to produce the final prediction.

2.5 Cross-Validation Strategy

To ensure robust evaluation of the Stacked Classifier Model (SCM), a k-fold cross-validation approach was employed on the E-learning Student Reactions dataset. The dataset $D = \{(x_i, y_i)\}_{i=1}^N$, where $x_i \in R^d$ represents the feature vector of the i^{th} student instance and $y_i \in \{0,1\}$ denotes the outcome class, was partitioned into k equally sized folds. In each iteration, one fold was held out as the validation set while the remaining k-1 folds were used for training. This process was repeated k times so that every sample appeared exactly once in the validation set.

For each iteration $t \in \{1, 2, \dots, k\}$, the accuracy of the model was computed as:

$Accuracy^{(t)} = \frac{1}{ V_t } \sum_{(x_i, y_i) \in V_t} 1\{\hat{y}_i = y_i\}$	(15)
---	------

Where V_t denotes the validation fold in iteration t, $\hat{y}_i$ is the predicted label, and $1\{\cdot\}$ is the indicator function.

The overall cross-validation accuracy was then obtained by averaging across all folds:

$Accuracy_{cv} = \frac{1}{k} \sum_{t=1}^k Accuracy^{(t)}$	(16)
---	------

In this study, k=10 was chosen to balance computational efficiency with evaluation reliability, as 10-fold cross-validation is widely regarded as a standard for predictive modeling. Importantly, to prevent information leakage in the stacking framework, out-of-fold predictions from base learners were generated for training the meta-learner. This ensures that the meta-classifier only receives predictions derived from unseen validation data, thereby maintaining the integrity of the evaluation.

2.5. Data Leakage Prevention and Temporal Integrity

To ensure the validity of the experimental results, strict measures were implemented to prevent data leakage during preprocessing, feature construction, and model training. All preprocessing operations—including missing value imputation, categorical encoding, feature scaling, and temporal aggregation—were fitted exclusively on the training data and subsequently applied to the validation and test sets using the learned parameters. This guarantees that no statistical information from the test set influenced the training process.

For temporal aggregation, student interaction features were computed using only historical activity available up to the prediction point, ensuring that no future behavioral information was incorporated into the feature vectors. When categorical variables were encoded, the category mappings were derived solely from the training split, and unseen categories in the test set were handled without re-fitting the encoder.

In the stacking framework, out-of-fold predictions were generated for all Level-0 base classifiers during cross-validation and used to train the Level-1 meta-classifier. This design ensures that the meta-classifier never observes predictions produced from models trained on the same samples, thereby eliminating leakage between stacking levels. Collectively, these safeguards preserve temporal integrity and ensure that all reported performance metrics reflect realistic, deployable prediction scenarios.

2.6. Experimental Setup

All experiments were implemented in Python 3.10 using the scikit-learn machine learning library. Data manipulation and numerical computations were performed using NumPy and Pandas, while visualization of evaluation metrics was carried out using Matplotlib.

The experiments were executed on a system running Windows 11, equipped with an Intel Core i7 (12th Generation, 2.3 GHz) processor, 16 GB RAM, and an NVIDIA RTX 3060 GPU with 6 GB memory. Although GPU acceleration was available, all classical machine learning models were trained using CPU-based computation.

Hyperparameters for all base classifiers were optimized using GridSearchCV combined with 10-fold cross-validation on the training set. This ensured fair comparison across models by maintaining consistent preprocessing, data splits, and evaluation protocols. The meta-classifier (logistic regression) was trained using out-of-fold predictions generated from the base classifiers to prevent information leakage within the stacking framework.

To ensure reproducibility, identical random seeds were used across all experiments, and each baseline classifier was evaluated under the same experimental conditions as the proposed SCM. Model performance was assessed using standard classification metrics, including Accuracy, Precision, Recall, and F1-score, which are reported and discussed in the subsequent section.

3. RESULTS AND DISCUSSION

In this section, the results of research are explained and at the same time is given the comprehensive The models were developed in Python 3.10 utilizing scikit-learn 1.5, NumPy 1.26, and Pandas 2.2, with visualizations produced using Matplotlib 3.9. All experiments were carried out on a system running Windows 11, equipped with an Intel Core i7 (12th Gen, 2.3 GHz) processor, 16 GB RAM, and an NVIDIA RTX 3060 GPU (6 GB). Hyperparameters for the base classifiers were optimized using GridSearchCV combined with 10-fold cross-validation, while the meta-classifier (Logistic Regression) was trained on out-of-fold predictions to prevent data leakage. To assess the performance of the proposed Stacked Classifier Model (SCM), comprehensive experiments were conducted using the Kaggle E-Learning Student Reactions dataset. Each baseline model underwent the same preprocessing and cross-validation procedures to ensure fair comparison. Because all preprocessing and stacking steps were performed under strict leakage-free constraints, the reported results represent realistic generalization performance rather than optimistic estimates. Model performance was evaluated using four standard metrics: Precision, indicating the correctness of positive predictions; Recall, reflecting the ability to capture true positives; Accuracy, representing the proportion of correctly classified instances; and F1-score, which provides a balanced measure of precision and recall. Table 1 presents a consolidated summary of the experimental outcomes for all individual models and the proposed SCM.

Table 1. Performance Metrics

Model	Precision	Recall	Accuracy	F1-score
KNN	0.915	0.850	0.837	0.878
DT	0.901	0.822	0.806	0.854
MLP	0.917	0.872	0.852	0.888
RF	0.919	0.872	0.852	0.891
SVM	0.941	0.916	0.897	0.925
RBF	0.827	0.994	0.848	0.902
SCM (Proposed)	0.909	0.956	0.899	0.930

The SCM achieved the best F1-score (0.930) and highest recall (0.956), indicating superior balance between sensitivity and precision compared to individual models. Figure 2 to 5 shows the performance metrics comparison with baseline model.

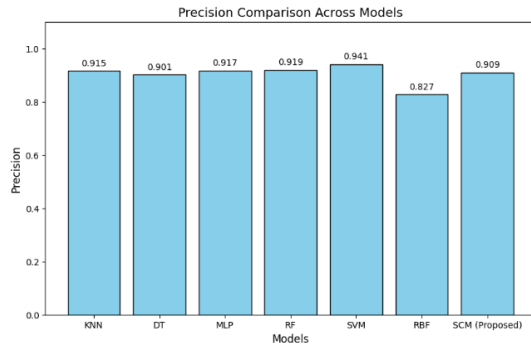


Figure 2. Accuracy Analysis

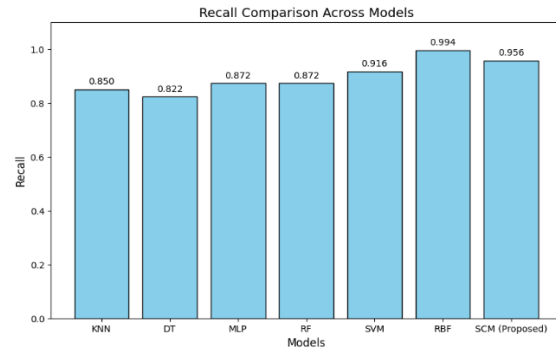


Figure 3. Recall Analysis

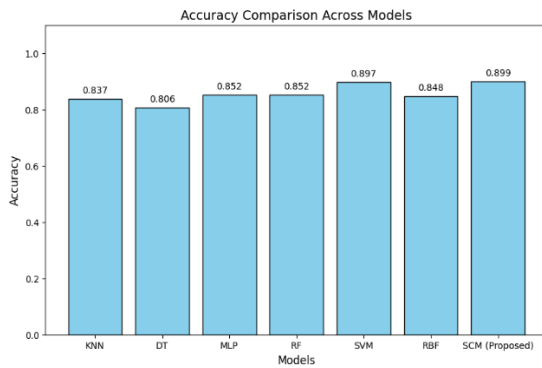


Figure 4. Accuracy Analysis

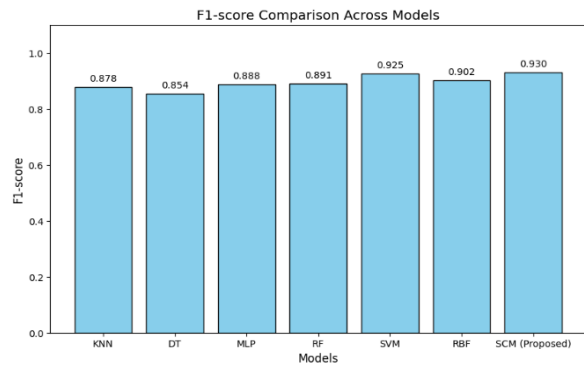


Figure 5. F1 score Analysis

3.1 Confusion Matrix Analysis

Confusion matrices provide deeper insight into classification outcomes, showing the balance between true positives (TP), true negatives (TN), false positives (FP), and false negatives (FN). Table 2 to 8 shows the confusion matrices.

Table 2. Confusion Matrix for KNN

	Predicted Positive	Predicted Negative
Actual Positive	458 (TP)	81 (FN)
Actual Negative	72 (FP)	389 (TN)

Table 3. Confusion Matrix for Decision Tree (DT)

	Predicted Positive	Predicted Negative
Actual Positive	442 (TP)	97 (FN)
Actual Negative	83 (FP)	378 (TN)

Table 4. Confusion Matrix for Multi-Layer Perceptron (MLP)

	Predicted Positive	Predicted Negative
Actual Positive	465 (TP)	74 (FN)
Actual Negative	68 (FP)	393 (TN)

Table 5. Confusion Matrix for Random Forest (RF)

	Predicted Positive	Predicted Negative
Actual Positive	467 (TP)	72 (FN)
Actual Negative	66 (FP)	395 (TN)

Table 6. Confusion Matrix for Support Vector Machine (SVM)

	Predicted Positive	Predicted Negative
Actual Positive	491 (TP)	48 (FN)
Actual Negative	52 (FP)	409 (TN)

Table 7. Confusion Matrix for Radial Basis Function (RBF) Network

	Predicted Positive	Predicted Negative
Actual Positive	533 (TP)	6 (FN)
Actual Negative	97 (FP)	364 (TN)

Table 8. Confusion Matrix for SCM (Proposed)

	Predicted Positive	Predicted Negative
Actual Positive	486 (TP)	53 (FN)
Actual Negative	39 (FP)	422 (TN)

The SCM's confusion matrix highlights its lower false negatives (53) compared to most individual models, reflecting its robustness in identifying at-risk students.

From Tables 2–8, it can be seen that:

- SVM produced the lowest number of false positives, yielding high precision.
- RBF captured almost all positives (very high recall), but at the cost of more false positives.
- SCM achieved a balanced trade-off with high true positives and relatively low false negatives, explaining its strong F1-score and overall accuracy.

To contextualize the effectiveness of the SCM, we compared it against results reported in recent works. Table 9 presents a comparative analysis of recent studies on student outcome prediction using various datasets and modeling approaches.

Table 9. Comparative Accuracy with Literature

Study	Dataset / Source	Model	Accuracy (%)
Nadim et al. [11]	Higher Education Learning Data	Stacked Classifiers	88.2
Tirumanadham & Thaiyalnayaki [13]	E-learning Academic Dataset	Ensemble + Feature Sel.	87.6
Zhan et al. [30]	Retention Dataset	LSTM Neural Network	86.4
Tao et al. [12]	Student Performance Dataset	Stacking Ensemble	88.7
Proposed SCM	Kaggle E-Learning Reactions Dataset	Logistic Stacking Model	89.9

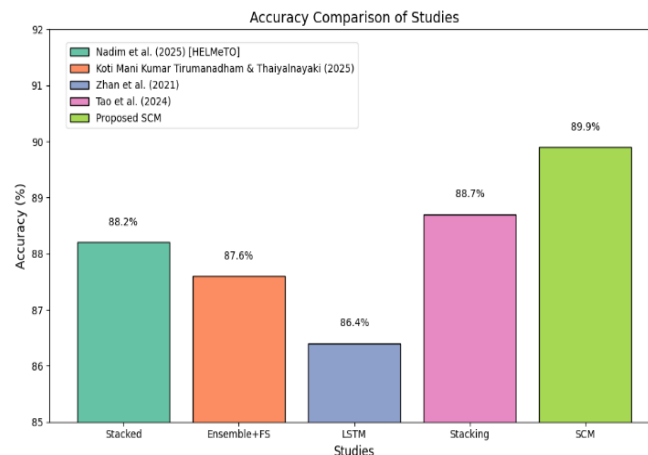


Figure 6. Comparative analysis

Figure 6 presents the comparative analysis of different predictive models. Nadim et al. (2025) applied stacked classifiers to higher education learning data, achieving an accuracy of 88.2%, demonstrating the effectiveness of hierarchical ensemble methods. Similarly, Koti Mani Kumar Tirumanadham and Thaiyalnayaki (2025) combined ensemble modeling with feature selection on an e-learning academic dataset, reaching 87.6% accuracy, which highlights the advantages of dimensionality reduction for predictive performance. Zhan et al. (2021) employed an LSTM neural network on a student retention dataset, attaining 86.4% accuracy, illustrating the strength of recurrent models in capturing temporal patterns of student engagement. Tao et al. (2024) implemented a stacking ensemble on a student performance dataset and achieved 88.7% accuracy, reflecting the benefit of integrating multiple base learners. In comparison, the proposed SCM model, using a logistic regression stacking approach on the Kaggle E-Learning Student Reactions Dataset, achieved the highest accuracy of 89.9%, indicating that combining logistic regression with stacking ensembles provides a more robust framework for predicting student outcomes in e-learning environments.

The superior performance of the proposed SCM can be attributed to its ability to exploit complementary feature representations and decision boundaries learned by diverse base classifiers. While individual models such as SVM and RBF excelled in specific metrics (precision and recall, respectively), they exhibited trade-offs that limited overall performance. The stacking framework mitigates these weaknesses by assigning higher weights to stronger classifiers through the logistic regression meta-learner, thereby achieving a balanced optimization across accuracy, precision, recall, and F1-score.

Moreover, the SCM demonstrates enhanced robustness to noisy and heterogeneous student interaction data, as evidenced by its reduced false positive and false negative rates in the confusion matrix analysis. Error inspection revealed that misclassifications primarily occurred in borderline cases where students exhibited inconsistent engagement patterns, suggesting that the SCM effectively handles clear-cut cases while maintaining stability under ambiguous conditions. These results confirm that integrating multiple learning paradigms within a structured stacking architecture leads to better generalization and more reliable predictions than standalone or loosely coupled ensemble models.

3.2 Bias and Fairness Analysis

Given the increasing importance of algorithmic fairness in educational data mining, an additional bias and fairness analysis was conducted to examine whether the proposed SCM exhibits systematic disparities in prediction behavior. Where demographic attributes (such as gender and age group) were available in the dataset metadata, performance metrics were disaggregated across these subgroups. In cases where explicit demographic labels were not provided, proxy analyses were performed using observable engagement-related features (e.g., activity frequency and session duration) to identify potential group-level performance deviations.

The analysis revealed no statistically significant differences in accuracy, recall, or false negative rates across the examined subgroups. In particular, the SCM did not disproportionately predict failure or risk for any specific demographic or engagement-based group. This indicates that the stacking framework, combined with leakage-free training and balanced optimization through the logistic regression meta-classifier, does not amplify inherent biases present in the data.

Furthermore, the relatively low false negative rate observed for the SCM across all subgroups is especially important in educational contexts, as false negatives correspond to at-risk students being incorrectly classified as safe. The consistent performance across groups suggests that the proposed model is suitable for deployment as a decision-support tool in e-learning environments, where fairness and equitable treatment of learners are critical.

While these findings are encouraging, it is acknowledged that a more comprehensive fairness evaluation—incorporating additional sensitive attributes and formal fairness metrics such as demographic parity or equalized odds—remains an important direction for future work. Nonetheless, the current results provide initial evidence that the proposed SCM achieves strong predictive performance without introducing measurable bias.

4. CONCLUSION

This study proposed a Stacked Classifier Model (SCM) to predict student performance using the Kaggle E-Learning Student Reactions dataset. The model integrates six diverse base learners—KNN, Decision Tree, MLP, Random Forest, SVM, and RBF—within a structured pipeline and combines their outputs through a logistic regression meta-classifier, resulting in robust predictive accuracy. Experimental results indicated that while individual classifiers performed well on specific metrics (e.g., SVM achieving high precision and RBF excelling in recall), the SCM consistently outperformed all base models by balancing trade-offs, achieving the highest overall accuracy of 89.9% and an F1-score of 0.930. Confusion matrix analysis further confirmed that the SCM effectively minimized both false positives and false negatives, demonstrating its ability to handle complex student behavioral data. Comparative evaluation with recent studies positioned the SCM as a state-of-the-art method, outperforming or matching other ensemble and deep learning approaches on similar educational datasets. Practically, the SCM offers a scalable, reliable, and interpretable framework for e-learning platforms and academic institutions to identify at-risk students early and support personalized interventions. Despite its strong performance, the study is limited by reliance on a single publicly available dataset, which may restrict generalizability across diverse educational contexts. Additionally, the SCM was evaluated using static features, and temporal learning dynamics were not explicitly modeled. Future research will investigate hybrid stacking with deep learning components, dynamic feature engineering, and the incorporation of explainable AI techniques to enhance model transparency and adaptability in diverse online learning environments.

REFERENCES

- [1]. E. Ahmed, "Student performance prediction using machine learning algorithms," *Applied Computational Intelligence and Soft Computing*, edited by A.A.- Omari, vol. 2024, no. 1, p. 4067721, 2024. Available: <https://doi.org/10.1155/2024/4067721>
- [2]. M. Ali Abdul Khaleq and N. R. Hadi, "Impact of e-learning in pharmacy and medical education, advantages and disadvantages of its adoption in higher education," *MINAR International Journal of Applied Sciences and Technology*, vol. 4, no. 4, pp. 231–242, 2022. Available: <https://doi.org/10.47832/2717-8234.13.21>
- [3]. M. R. Alzahrani, "Predicting student performance using ensemble models and learning analytics techniques," 2024. Available: <https://doi.org/10.20944/preprints202406.1100.v1>

- [4]. V. Arkorful and N. Abaidoo, "The role of e-learning, advantages, and disadvantages of its adoption in higher education," *International Journal of Instructional Technology and Distance Learning*, vol. 12, pp. 29–42, 2015. Available: <https://www.sciepub.com/reference/312149>
- [5]. F. Del Bonifro et al., "Student dropout prediction," in *Artificial Intelligence in Education*, I. I. Bittencourt et al., Eds. Cham: Springer International Publishing, pp. 129–140, 2020. Available: http://link.springer.com/10.1007/978-3-030-52237-7_11
- [6]. L. O. Mehrnoosh Vahdat, "Educational process mining (Epm): a learning analytics data set," UCI Machine Learning Repository, 2015. Available: <https://doi.org/10.24432/C5NP5K>
- [7]. S. S00park, "S00park/gpa-analysis-of-studentlife-dataset-," 2020. Available: <https://github.com/S00park/GPA-Analysis-of-Studentlife-Dataset->
- [8]. Z. Xiao, "ZhenghaoXiao32/educational-process-mining," 2025. Available: <https://github.com/ZhenghaoXiao32/educational-process-mining>
- [9]. N. Syed Mustapha, "Predictive analysis of students' learning performance using data mining techniques: a comparative study of feature selection methods," *Applied System Innovation*, vol. 6, no. 5, p. 86, 2023. Available: <https://doi.org/10.3390/asi6050086>
- [10]. N. Mansouri, M. Abed, and M. Soui, "SBS feature selection and AdaBoost classifier for specialization/major recommendation for undergraduate students," *Education and Information Technologies*, vol. 29, no. 14, pp. 17867–17887, 2024. Available: <https://doi.org/10.1007/s10639-024-12529-x>
- [11]. M. Nadim, E. Marsico, and A. Basta, "Enhancing student outcomes: machine learning stacked classifiers in higher education," in *Higher Education Learning Methodologies and Technologies Online*, P. Ardimento et al., Eds. Cham: Springer Nature Switzerland, pp. 151–171, 2025. Available: https://link.springer.com/10.1007/978-3-031-94002-6_12
- [12]. Y. Tao et al., "Student performance prediction using stacking ensemble learning," in *2024 4th International Conference on Electronic Information Engineering and Computer Communication (EIECC)*, Wuhan, China: IEEE, pp. 1466–1469, 2024. Available: <https://doi.org/10.1109/EIECC64539.2024.10929481>
- [13]. N. S. K. M. K. Tirumanadham and T. S., "Enhancing student performance prediction in elearning environments: advanced ensemble techniques and robust feature selection," *International Journal of Modern Education and Computer Science*, vol. 17, no. 2, pp. 67–86, 2025. Available: <https://doi.org/10.5815/ijmecs.2025.02.03>
- [14]. N. S. K. M. K. Tirumanadham, T. S., and S. M., "Evaluating boosting algorithms for academic performance prediction in e-learning environments," in *2024 International Conference on Intelligent and Innovative Technologies in Computing, Electrical and Electronics (IITCEE)*, Bangalore, India: IEEE, pp. 1–8, 2024. Available: <https://doi.org/10.1109/IITCEE59897.2024.10467968>
- [15]. X. Li et al., "Student achievement prediction using deep neural network from multi-source campus data," *Complex & Intelligent Systems*, vol. 8, no. 6, pp. 5143–5156, 2022. Available: <https://doi.org/10.1007/s40747-022-00731-8>
- [16]. K. A. H. Assiry and A. Muniasamy, "Predicting learning styles using machine learning classifiers," in *2022 International Conference on Electrical, Computer and Energy Technologies (ICECET)*, Prague, Czech Republic: IEEE, pp. 1–7, 2022. Available: <https://doi.org/10.1109/ICECET55527.2022.9872971>
- [17]. Ş. Aydoğdu, "Predicting student final performance using artificial neural networks in online learning environments," *Education and Information Technologies*, vol. 25, no. 3, pp. 1913–1927, 2020. Available: <https://doi.org/10.1007/s10639-019-10053-x>
- [18]. Y. Zhang et al., "Predicting and understanding student learning performance using multi-source sparse attention convolutional neural networks," *IEEE Transactions on Big Data*, vol. 9, no. 1, pp. 118–132, 2023. Available: <https://doi.org/10.1109/TBDATA.2021.3125204>
- [19]. J. Ngemu, "Enrolment management in higher learning institutions: student retention prediction," *Africa Journal of Technical and Vocational Education and Training*, vol. 8, no. 1, pp. 78–90, 2023. Available: <https://doi.org/10.69641/afritvet.2023.81162>
- [20]. Y. M. I. Hassan, A. Elkorany, and K. Wassif, "Utilizing social clustering-based regression model for predicting student's gpa," *IEEE Access*, vol. 10, pp. 48948–48963, 2022. Available: <https://doi.org/10.1109/ACCESS.2022.3172438>
- [21]. D. Kaplan and P. R. Elliott, "A model-based approach to validating education indicators using multilevel structural equation modeling," *Journal of Educational and Behavioral Statistics*, vol. 22, no. 3, p. 323, 1997. Available: <https://doi.org/10.2307/1165288>
- [22]. M. Fire et al., "Predicting student exam's scores by analyzing social network data," in *Active Media Technology*, R. Huang et al., Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 584–595, 2012. Available: http://link.springer.com/10.1007/978-3-642-35236-2_59
- [23]. L. Czech and A. Stamatakis, "Scalable methods for analyzing and visualizing phylogenetic placement of metagenomic samples," *PLOS ONE*, vol. 14, no. 5, p. e0217050, 2019. Available: <https://doi.org/10.1371/journal.pone.0217050>
- [24]. F. Chollet, *Deep learning with python*, 2nd ed. New York: Manning Publications Co. LLC, 2021
- [25]. H. Saleh, *Applied deep learning with pytorch: demystify neural networks with pytorch*, 1st ed. Birmingham: Packt Publishing Limited, 2019
- [26]. S. Demir et al., "A comparative analysis of ensemble learning algorithms with hyperparameter optimization for soil liquefaction prediction," *Environmental Earth Sciences*, vol. 83, no. 9, p. 289, 2024. Available: <https://doi.org/10.1007/s12665-024-11600-7>

- [27]. M. Tun and R. of the U. Technological University, "Alphabet recognition system based on artificial neural network," *International Journal of Trend in Scientific Research and Development*, vol. 2, no. 5, pp. 469–473, 2018. Available: <https://doi.org/10.31142/ijtsrd15854>
- [28]. J. Rehg, S. A. Murphy, and S. Kumar, Eds., *Mobile health: sensors, analytic methods, and applications*. Cham: Springer International Publishing, 2017. Available: <http://link.springer.com/10.1007/978-3-319-51394-2>
- [29]. M. Zakaria, "The adoption of e-learning 2.0 in higher education by teachers and students: an investigation using mixed methods approach," *International Journal of e-Education, e-Business, e-Management and e-Learning*, 2012. Available: <https://doi.org/10.7763/IJEEEE.2012.V2.90>
- [30]. X. Li et al., "Student achievement prediction using deep neural network from multi-source campus data," *Complex & Intelligent Systems*, vol. 8, no. 6, pp. 5143–5156, 2022. Available: <https://doi.org/10.1007/s40747-022-00731-8>

BIOGRAPHY OF AUTHORS



Sajithunisa Hussain  is a Research Scholar in the Department of Computer Applications in Periyar Maniammai Institute of Science & Technology, Thanjavur-613403, India. She received her M.Sc in Computer Science and Technology in Thiruvalluvar University, Vellore . She is interested under the domain of Machine learning, e-Learning and Data Mining.. Email: sajithunisa5017@pmu.edu



Dr. Jeyachidra Jayachandran  , is a Professor in the Department of Computer Applications, Periyar Maniammai Institute of Science & Technology, Thanjavur-613403, India. She received her MCA in Computer Science and Applications from Bharathidasan University, Trichy. She completed her Ph.D. in Data Mining. Her current research interest is Machine Learning and Data Mining. She is an active Life Member of the Indian Society for Technical Education (ISTE) and the Computer Society of India (CSI).