Applied Computing and Informatics

Deep learning approaches for predicting cheating from student exam results: a comparative study under imbalanced data conditions

Dao Phuc Minh Huy and Nguyen Gia Nhu
School of Computer Science, Duy Tan University, Da Nang, Vietnam, and
Dac-Nhuong Le
Faculty of Information Technology, Hai Phong University, Haiphong, Vietnam

Received 26 July 2025
Revised 19 October 2025
4 November 2025
Accepted 20 November 2025

Abstract

Purpose – To detect academic misconduct in students' assessment score trajectories under severe class imbalance. The paper compares tabular learners, gradient-boosting models, and sequence-aware deep networks, and proposes a precision–recall–centric evaluation and deployment protocol (calibration, threshold selection and Recall@Top-k%) tailored to rare-event screening in educational settings.

Design/methodology/approach – A cohort of 1,527 students (2021–2024) is modeled using ten algorithms: LR, DT, RF, SVM, MLP, XGBoost, CatBoost, LightGBM, GRU-RNN and 1D-CNN. Features encode sequential score dynamics and metadata. Models are tuned via cross-validation; probabilities are calibrated (Platt/Isotonic); operating thresholds are chosen on validation to maximize minority-class F1 or a cost-sensitive utility. Performance is assessed on a hold-out test set with PR-AUC (headline), F1(+), Recall@Top-k%, ROC-AUC, calibration curves, Brier, and bootstrap CIs.

Findings – Sequence-aware models dominate: GRU-RNN and 1D-CNN achieve ROC-AUC ~0.97–0.98 and the highest F1(+) and Recall@Top-5%. Tabular/boosting baselines show ~0.90 accuracy yet miss most positives at the default 0.5 threshold, highlighting the necessity of calibration and threshold optimization. With PR-centric selection and tuned operating points, deep temporal models yield strong screening utility for limited human review budgets.

Research limitations/implications – Labels reflect suspected–not adjudicated–cheating, introducing noise. The single-institution cohort may limit external validity; temporal shift across semesters can degrade performance. Future work should include multi-site evaluation, collusion/graph modeling, semi-/weak-supervision for noisy labels and governance topics (fairness audits, drift monitoring and uncertainty reporting).

Practical implications – This study highlights how educational institutions can leverage machine learning for early detection of academic dishonesty based on historical performance data. CNN and RNN models are promising tools for identifying anomalous learning patterns. However, practical deployment requires preprocessing techniques to manage class imbalance and threshold optimization to reduce false negatives. The findings provide a roadmap for building automated cheating detection systems in both online and traditional assessment environments.

Social implications – By improving the ability to detect cheating, this research contributes to fairer academic environments, upholding educational integrity and credibility of credentials. However, ethical considerations must be taken into account to avoid false accusations and ensure student rights. Human-in-the-loop systems are crucial for verifying algorithmic predictions before disciplinary action, thereby fostering transparency and accountability in automated decision-making processes.

Originality/value – The study unifies a minority-focused evaluation protocol with a comprehensive comparison of tabular, boosting, and sequence-aware models for cheating detection from score trajectories. It demonstrates the decisive value of temporal representation learning and provides a reproducible pipeline and operational metrics that align model performance with real investigative workflows.

Keywords Predicting cheating, Educational data mining (EDM), Logistic regression, Decision trees, Random forests, SVM, MLP, RNN, CNN

Paper type Research article

Applied Computing and Informatics
Emerald Publishing Limited
e-ISSN: 2210-8327
p-ISSN: 2634-1964
DOI 10.1108/ACI-07-2025-0308

1. Introduction

Academic grades constitute primary proxies for student learning, encapsulating the mastery of disciplinary knowledge and skills essential for both academic progression and professional success. Consequently, accurate forecasting of final grade outcomes is pivotal for educational stakeholders, guiding timely interventions and policy formulation. However, the heterogeneity and high dimensionality of longitudinal academic records pose significant challenges to the construction of robust predictive models. In response, machine learning techniques—integrating advanced algorithms with rigorous statistical methodologies—have proven effective at distilling key determinants of scholastic performance, mapping learning behaviors and trajectories, and identifying anomalous patterns associated with academic dishonesty. The fusion of these computational strategies underpins the field of Educational Data Mining (EDM), which systematically exploits data generated within educational systems to derive empirically grounded insights and actionable knowledge for both performance prediction and cheating prevention. In this study, we cast the prediction of student cheating on final assessments as a dual task of classification and anomaly detection using score-based features. Drawing on EDM and machine learning, we propose an end-to-end framework that transforms historical performance data into predictive models capable of stratifying learners by risk: those likely to excel, those requiring support, and those at elevated risk of failure or dishonest behavior. Continuous outcomes are modeled via regression analysis—most commonly linear regression—where dependent variables (final scores) are estimated as functions of continuous or categorical predictors. Discrete outcomes employ classification algorithms, including support vector machines, backpropagation-trained neural networks, and k-nearest neighbors, to allocate students into risk categories. Collectively, these methods facilitate early detection of potential cheating and inform tailored prevention strategies.

In this work, we frame cheating prediction as (1) score-trajectory classification and (2) rank-based triage. Given pronounced class imbalance, we evaluate models using PR-centric metrics—minority-class F1, PR-AUC, and Recall@Top-k%—and treat ROC-AUC/accuracy as secondary summaries. We calibrate probabilities (Platt/Isotonic) and select thresholds on the validation fold to maximize F1(+) or a cost-sensitive utility. We then confirm generalization on a hold-out test set and report bootstrap 95% CIs. The rest of this paper is organized as follows. Section 2 surveys relevant literature. Section 3 details the proposed methodology. Section 4 outlines the experimental setup and presents a comparative analysis of multiple algorithms. Finally, Section 5 concludes with key findings and future research directions.

2. Related works

The deployment of machine learning for predicting and detecting academic dishonesty has seen rapid growth, with studies spanning anomaly detection and behavioral analytics Elrahman *et al.* [1] demonstrated that interactive eTextbook engagement data can predict student performance, suggesting that fine-grained interaction logs may also reveal irregular behaviors associated with cheating. Subsequent work [2, 3] applies anomaly-detection techniques to continuous assessment results, showing that deviations from projected score trajectories—rather than simple outliers-can flag potential dishonesty in final examinations. Ensemble methods, particularly Random Forests, have proven effective in modeling cheating behavior: RF classifiers successfully identified reported instances of misconduct among elementary students [4], while XGBoost and related deep-learning models have been leveraged to analyze online exam interactions and detect collusive patterns [5, 6]. Sequential architectures, notably LSTM networks, extend this capability by capturing temporal dependencies in test-score sequences, thereby improving sensitivity to evolving anomalies over time [7]. Beyond post-hoc detection, machine learning insights have informed proactive interventions, such as optimized seating algorithms that minimize collusion risk based on historical cheating patterns [8]. Collectively, these studies illustrate the versatility of both

traditional classifiers and advanced neural models in preserving assessment integrity through the systematic analysis of performance and behavioral data [2, 9].

Machine learning has become a cornerstone for predicting and detecting student cheating from exam data, spanning traditional classifiers, sequential models, deep learning architectures, and even realtime video analysis. The principal methodologies are summarized as follows:

(1) *Traditional and Ensemble Classification:* Binary classifiers—including generalized linear models (GLM) [7], logistic regression (LR) [10], decision trees (DT), and random forests (RF) [4]—leverage features drawn from student demographics and constructs such as the Fraud Triangle (rationalization, opportunity, pressure). These models typically achieve precision rates of 50–75% for the cheating class. RF frequently identify "opportunity" variables (e.g., unsupervised exam settings) as the strongest predictors, whereas GLM, LR, and DT emphasize"rationalization"factors (e.g., self-reported justifications for cheating).

(2) *Sequential Models with Outlier Detection:* Recurrent neural networks (RNNs) treat cheating detection as an anomaly-detection task on time-ordered assessment sequences. By training RNNs to forecast final-exam scores from preceding quizzes and midterms, significant deviations between predicted and observed scores are flagged as potential dishonesty [11]. This temporal framing yields true-positive rates near 95% and false-positive rates around 5%, outperforming static outlier-detection baselines by capturing order-dependent irregularities in score trajectories.

(3) *Deep Learning Variants:* Advanced neural architectures—dense feedforward networks (DNN), LSTM, and hybrid DenseLSTM models [12]—have been evaluated on academic datasets. These deep models exploit complex temporal and nonlinear interactions among assessments, with DenseLSTM variants reporting up to 95% accuracy in cheating detection, significantly surpassing simpler classifiers.

(4) *Video - based Behavioral Monitoring*: For realtime invigilation, object-detection frameworks such as YOLOv8 [3], augmented with attention mechanisms, analyze proctoring video streams to identify cheating behaviors (e.g., whispering, note passing, unauthorized device use). Trained on a bespoke dataset of staged infractions, this approach achieves approximately 82.7% detection accuracy, reducing reliance on constant human oversight. Modern remote-proctoring stacks typically comprise: face/pose tracking; gaze/attention estimation; object detection for handheld/secondary screens; and event logic for rule violations. Many pipelines adopt YOLOv8-based detectors (for devices, notes, mouth/hand occlusions), sometimes complemented by temporal modules (e.g., TCN/GRU) or rule-based smoothing to suppress flicker. Reported benefits include real-time operation and modularity; known challenges include domain shift (camera/lights), privacy, and dataset bias. Our work complements these pipelines by focusing on score-trajectory signals, which are modality-agnostic and auditable for after-the-fact investigations [13–15].

These machine learning approaches provide effective tools for early detection and prevention of academic misconduct based on exam results and student behavior. The combination of temporal modeling of exam scores and behavioral video analysis covers a broad spectrum of cheating detection scenarios, enhancing academic integrity in both traditional and online examination settings (see Table 1).

Parallel to score-based modeling, several EDM 2025 accepted/demo systems treat the examinee's screen stream as a spatiotemporal signal. A common pattern is a hybrid CNN + RNN (e.g., 2D CNN on frame snippets for spatial cues like unauthorized UI widgets, overlaid with GRU/LSTM on frame indices for temporal dynamics). These systems report that short "burst" kernels detect micro-events (copy/paste dialogs, sudden context switches), while

Table 1. The summary of machine learning algorithms for predicting cheating from exam results

Approach	Algorithms/ models	Data used	Key strengths	Performance highlights
Exam score anomaly detection	RNN + Outlier Detection	Sequential exam scores	Captures temporal patterns, unsupervised	TPR∼95%, FPR∼5%
Binary classification	GLM, LR, DT, RF	Demographics + psychological features	Interpretable, uses fraud theory factors	Precision 50–75% for cheating class
Deep learning temporal models	DNN, LSTM, DenseLSTM, RNN	Exam score sequences	High accuracy on complex patterns	Up to 95% accuracy
Exam score anomaly detection	RNN + Outlier Detection	Sequential exam scores	Captures temporal patterns, unsupervised	TPR∼95%, FPR∼5%
Video behavior detection	Improved YOLOv8 + Attention mechanism	Exam room video footage	Real-time detection of cheating actions	∼82.7% accuracy

the recurrent stack stabilizes predictions across seconds-long sequences, improving precision at low false-alarm rates. We contextualize our CNN/RNN design choices with this finding, emphasizing kernel sizes for burst-like artifacts and sequence-level aggregation [16, 17]. The 2025 EDM surveys converge on three trends: (1) sequence-centric models for longitudinal student traces; (2) multimodal fusion (scores, clickstreams, proctoring media); and (3) imbalance-first evaluation using PR-AUC and cost-sensitive F-metrics rather than accuracy/ROC alone. The surveys also advocate for threshold selection as a model component, not a post-hoc tweak, aligning with our validation-time F1/utility maximization and probability calibration pipeline [18]. Cheating detection is inherently imbalanced (typically <10% positives). Recent guidance emphasizes: (1) PR-AUC as the headline metric (baseline ≈ prevalence), (2) class-weighted or focal losses for deep models, (3) calibration (Platt/Isotonic) before thresholding, (4) minority-aware minibatching and within-fold SMOTE (training only) to avoid leakage, and (5) reporting Recall@Top-k% risk for operational triage. We adopt this toolbox and treat threshold/operating-point choice as part of model selection, not a cosmetic step [19]. Recent ACM work broadens cheating from individual-level anomalies to group-level collusion. Techniques include graph-based similarity over response vectors and timing, pairwise sequence alignment for suspicious synchrony, and clustering on keystroke/interaction embeddings to identify coordinated behavior. Such approaches caution that "normal-looking" scores may still mask collusion detectable only via relational signals [20–25]. While our current study is individual-centric, we outline collusion modeling as a near-term extension. We (1) compare deep temporal models (CNN/RNN) with strong tabular baselines under strict imbalance protocols; (2) report PR-centric metrics, calibration, and threshold sweeps; and (3) propose pathways to multimodal fusion (with screen-sequence/proctoring cues) and collusion graphs, aligning the score-based approach with 2025 EDM/proctoring practice.

3. Methodology

In this section, we propose a predicted student cheating model was analyzed based on the scores in the learning process of the subjects in the previous semester flows shown on Figure 1.

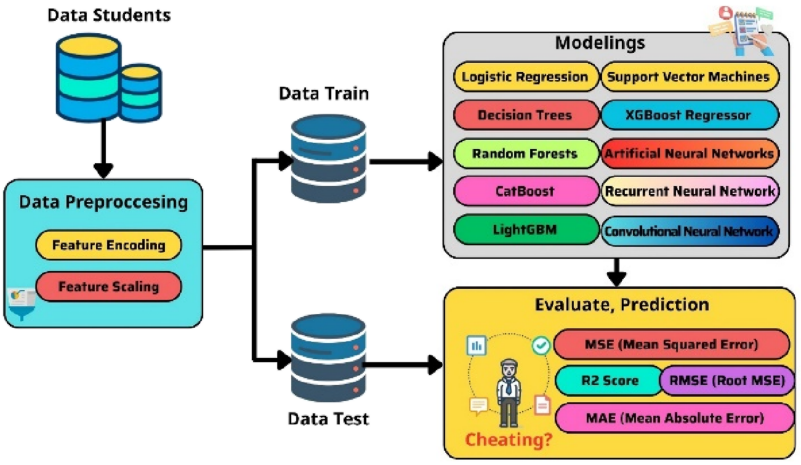

Figure 1. The predicted student cheating model was analyzed based on the scores in the learning process

3.1 The analytical workflow

Figure 2 summarizes the principal stages of our analytical workflow, encompassing data preprocessing (feature encoding and scaling), data partitioning (training versus testing), model development, and performance evaluation. Each stage is detailed below to elucidate the research methodology.

(1) *Step 1. Data Collection:* We assembled a student-level dataset in which each record corresponds to an individual's performance during the prior semester. Variables include raw examination scores (aggregate and by subject), time per exam or section, frequency of answer changes or flagged anomalies, and course-related metadata (e.g., difficulty level, exam version). The resulting raw data table is hereafter denoted "*Data Student*."

(2) *Step 2. Data Preprocessing*. Prior to model fitting, we transform and standardize all inputs to ensure numerical compatibility:

(1) *Feature Encoding*: Categorical attributes, such as course code and exam version—are converted to numerical vectors using one-hot, label, or target encoding.

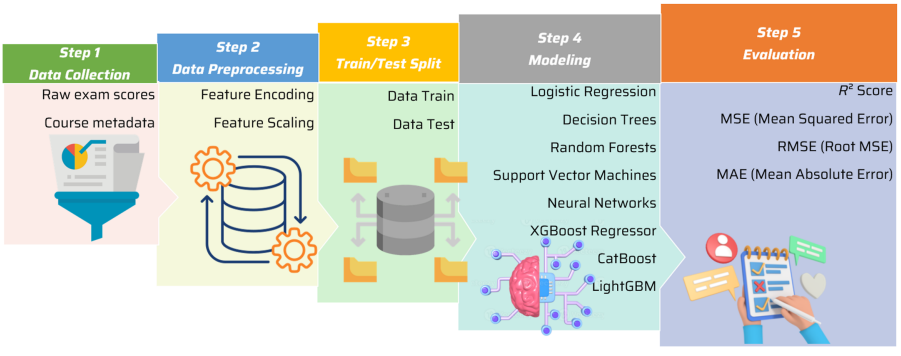

Figure 2. Our proposal steps for predicting student cheating model

(2) *Feature Scaling*: Continuous measures (e.g., hours studied, score percentages) are normalized via z-score standardization or min–max scaling so that all predictors contribute on a comparable scale.

The output of this stage is a fully numeric feature matrix, X.

Step 3. Train/Test Split: The dataset is partitioned into a training subset (70–80% of records) for model fitting and a hold-out test subset (20–30%) for unbiased evaluation. This division ensures that performance metrics reflect the model's ability to generalize to unseen student data.

Step 4. Modeling: We fit a suite of supervised learning algorithms to the training set, interpreting their continuous outputs as a "cheating risk score" in the range [0,1]. Candidate models include logistic regression, decision trees, random forests, support vector machines, and gradient-boosted tree regressors (e.g., XGBoost, LightGBM, CatBoost [26–28]). Although inherently regression methods, their predictions are thresholded (e.g., score $\geq$0.5) to flag instances at elevated risk of academic dishonesty. The choice of XGBoost, LightGBM, and CatBoost is motivated by their proven capacity to model complex, nonlinear relationships and to maximize predictive R^2 on analogous educational datasets [29, 30]. All models produce a cheating-risk probability $\widehat{p} \in [0,1]$. We evaluate both threshold-free discrimination (PR-AUC, ROC-AUC) and threshold-dependent operating points (F1(+) at τ^*, Recall@Top-k%). The operating threshold τ^* is chosen on the validation fold to maximize F1(+) unless a cost-utility is specified.

Step 5. Evaluation. Each student's full sequence is confined to a single split (train/val/test). We further check year/program stratification to reduce cohort leakage. Model performance is assessed on the hold-out test set by comparing predicted risk scores to confirmed cheating incidents (ground-truth labels). We compute the following metrics: R^2, MSE, RMSE, MAE. Together, these measures quantify both the explanatory power and predictive accuracy of our approach. We calibrate $\widehat{p}$ using Platt (linear-margin models) or Isotonic (tree/deep models), and summarize probability quality with Brier score = MSE (lower is better).

3.2 Models and algorithms

LR [10]: Treat cheating as a binary label and use exam-derived features (scores, time per question, answer-change patterns) to predict its probability with logistic regression. If we denote our feature vector for student i as $x_i = (x_1, x_2, \ldots, x_k)$, LR models the log-odds as a linear function of features: $\log \frac{P(cheat|x_i)}{1-P(cheat|x_i)} = \beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \ldots + \beta_k x_{ik}$, yielding p via the sigmoid. Coefficients are interpretable, and a tunable threshold (often $\approx$0.5) balances false positives vs. false negatives.

DT [4] models each student via features (quiz averages, homework rates, exam scores, time-on-task) and greedily selects axis-aligned splits that maximize impurity reduction (e.g., Gini), recursing until depth/leaf/min-impurity criteria are met; leaves output the majority class. This captures nonlinear patterns (e.g., low homework + high exam). A RF [4] ensembles many trees built on bootstrap samples with feature subsampling at each split; predictions are by majority vote. Out-of-bag data estimate error and support permutation feature importance.

Splitting Criteria, Tuning, and Interpretability: Gini impurity ($G = 1 - \sum p^2$) drives split selection in classification; for regression targets such as continuous "cheating risk scores," variance reduction replaces impurity measures. Hyperparameters—tree depth, minimum leaf size, number of trees, and features per split—are typically optimized via cross-validation, targeting ROC-AUC or F_1-score to balance sensitivity (detecting cheaters) against specificity (minimizing false positives). Crucially, tree-based models offer transparency: a single decision

tree can be rendered as clear "if–then" rules (e.g., "IF homework_completion <60% AND final_exam_score >85% THEN cheating"), granting instructors actionable insights. In Random Forests, feature-importance rankings and partial-dependence plots further elucidate how each metric influences cheating probability, guiding improvements in assessment design to mitigate vulnerabilities.

SVM [31]: Represent each student by performance features and train an SVM that, via a kernel map $\phi(x)$, finds a max-margin separator with soft errors controlled by C. Use RBF $K(x_i, x_j) = \exp(-\gamma \|x_i - x_j\|^2)$ or polynomial $K(x_i, x_j) = (\alpha x_i^T x_j + r)^d$ kernels to capture nonlinear interactions. Tune C, γ, α, r, d by cross-validated grid search for ROC-AUC or F1. The decision boundary is defined by a sparse set of support vectors; examining them surfaces borderline cases. Rank features with RFE, and use $|f(x)|$ (distance to the hyperplane) as a confidence score to prioritize human review.

ANN: Use an ANN [32] to flag cheating by featurizing both levels and dynamics of performance (raw scores, successive deltas, rolling means/variances), handling missing data, and standardizing inputs. Train a compact ReLU MLP (e.g., 128 → 64→32 neurons) with dropout (20–30%) and L_2, ending in a sigmoid/softmax. Split data by student to avoid leakage; optimize binary cross-entropy with Adam ($\sim 10^{-4}$) and early stopping. With scarce labels, use the same network as an autoencoder and mark high reconstruction error; mitigate class imbalance via loss weighting or targeted oversampling. Assess precision/recall, F_1, PR-AUC—or threshold reconstruction scores—and keep a human-in-the-loop using metadata (timestamps, IPs). Periodically fine-tune and recalibrate thresholds each semester to handle drift.

RNN [11]: Model each student as a time-ordered sequence of assessments, with per-step vectors of raw scores plus deltas/rolling stats. Use an RNN with LSTM/GRU [7] cells (e.g., 64 units; optionally stacked) to capture jumps, plateaus, and irregular swings; apply dropout to curb overfitting. Feed the final hidden state to a dense sigmoid for anomaly probability. Train with binary cross-entropy; when labels are scarce, use a seq-to-seq autoencoder and flag high reconstruction error. Split data by student to prevent temporal leakage; handle imbalance via class-weighted loss or oversampling. Stabilize training with LR scheduling and early stopping on validation AUC/F1. Evaluate with precision-oriented metrics (precision@k, recall, F1) or set thresholds on reconstruction-error distributions. Keep a human-in-the-loop using contextual metadata, and retrain/recalibrate regularly to track concept drift.

CNN [12]: Convert each student's timeline into a 2D feature map (events × metrics: raw scores, deltas, rolling stats). Apply small-kernel CNNs (e.g., 3×3 or 1×5) with ReLU and max pooling to detect local temporal–feature motifs (e.g., sudden exam spikes after moderate quizzes). After 2–3 conv–pool blocks, flatten to dense layers with dropout (20–30%) and L2, then a sigmoid/softmax for anomaly probability. Keep whole-student maps in a single split to avoid leakage; train with binary cross-entropy (Adam $\sim 10^{-4}$) and early stopping. With sparse labels, use a convolutional autoencoder and flag high reconstruction error. Evaluate via precision, recall, F1, precision@k or thresholded reconstruction scores, add human review with contextual metadata, and retrain each term to handle drift.

XGBoost [33] is a scalable gradient-boosted tree method that fits each new regression tree $f_k(x)$ to the current residuals (negative loss gradients), updates with learning rate η, and predicts $\hat{y} = \sum_{k=i}^{K} f_k(x_i)$ [34]. Second-order optimization, pruning, and regularization curb overfitting and handle mixed data efficiently. CatBoost [33–35] also boosts trees but adds ordered boosting and efficient categorical encodings with symmetric trees, reducing target leakage and improving robustness on small/heterogeneous datasets; predictions sum leaf values across trees. LightGBM [26] speeds training via leaf-wise growth with depth limits, histogram-based splits, and parallelism, enabling large-scale learning with low memory and fast convergence; each iteration updates $F_t(x) = F_{t-1}(x) + \sum_{j=1}^{J_t} \eta w_j^{(t)} I(x \in R_j^{(t)})$ by adding weights $w_j^{(t)}$ on leaf regions $R_j^{(t)}$ over the J_t leaves [36].

3.3 Neural architectures and design rationale

In this subsection, we propose the neural architectures and design rationale. Let each student *i* yield a length-T sequence of d-dimensional vectors $x_t \in \mathbb{R}^d$ containing raw scores, temporal deltas (e.g., Δquiz), and rolling statistics. We consider two complementary inductive biases:

(1) RNN: We employ a two-layer GRU network with 64 and 32 hidden units, respectively. Recurrent dropout is set to 0.2. The recurrent stack feeds a fully connected layer with 16 units and ReLU activation, followed by a sigmoid output. Each training instance is a length-*T* sequence per student; at time step *t*, the input is a *d*-dimensional feature vector comprising raw scores, temporal deltas, and rolling statistics. In preliminary model sweeps, GRUs were preferred over LSTMs because they achieved comparable AUC-PR with fewer parameters (final state → Dense(16, ReLU) → Sigmoid. Adam ($lr = 10^{-3}$), batch = 64, early stopping (patience = 10) on validation AUC-PR).

(2) 1D-CNN: Features are reshaped into a (time×feature) tensor and processed by temporal Conv1D blocks to capture short-range dynamics:[Conv1D (filters∈{32,64},kernel_size∈{3,5}), BatchNorm, ReLU, MaxPool(2)] ×2→Dense(64)→Dropout(0.3)→Sigmoid. The mixed kernel sizes target burst-like patterns (e.g., abrupt jumps preceding finals), while pooling imparts limited shift invariance along the timeline (Adam ($lr = 10^{-3}$), batch = 64).

(3) Rationale: RNNs model order-sensitive trajectories in student performance, whereas 1D-CNNs detect localized temporal motifs (e.g., a sudden spike after a period of stable quiz scores). These inductive biases align more closely with hypothesized cheating signatures than those of static tabular learners. The model architectures and hyperparameters are show in Table 2 and Figure 3 below:

4. Experimental results

4.1 Data selection and preparation

The dataset comprises 1,527 anonymized student records collected from the School of Computer Science at Duy Tan University (SCS-DTU) over the 2021–2024 academic years. Each record includes demographic attributes, academic performance indicators (cumulative GPA and individual subject scores), and a binary label denoting suspected cheating. During preprocessing, non-informative attributes (student name, nationality, campus) were discarded,

Table 2. Model architectures and hyperparameters

Model	Layers	Key hyperparameters	Regularization	Optimizer/ LR	Epochs	Early stop
GRU-RNN	[GRU64 → GRU32] → Dense16 → Sigmoid	recurrent_ dropout = 0.2	L2 = 1e−5 Dropout on dense = 0.2	Adam 1e−3	50	yes (AUC-PR)
1D-CNN	2×[Conv1D {32,64}, k = {3,5} → BN → ReLU → MaxPool(2)] → Dense64 → Dropout0.3 → Sigmoid	stride = 1; padding = ”same”	Dropout = 0.3	Adam 1e−3	50	yes (AUC-PR)
MLP (baseline)	128 → 64→32, ReLU	weighted BCE	Dropout = 0.3			

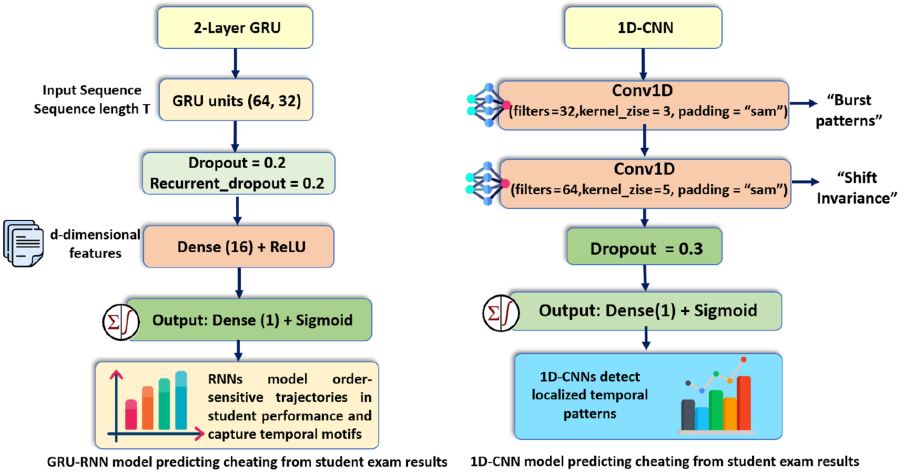

Figure 3. GNU-RNN and 1D-CNN diagrams

retaining only student ID, GPA scores, and per-subject grade points for all courses taken over four years. Numerical grades were converted into letter grades (A, B, C, D, F) and mapped to a 4-point scale, while cumulative GPAs were categorized into five classes: Excellent, Very Good, Good, Average, and Poor. The prepared dataset spans ten distinct training programs—including Software Engineering, Computer Science, Artificial Intelligence, Information Security, Data Science, and related majors—with each program comprising between 80 and 88 courses, ensuring a comprehensive set of features for subsequent machine-learning model development and evaluation [37–39]. The process of selecting, cleaning data, normalizing data and removing unnecessary data to select important features is shown in Figure 4.

Data acquisition began with the receipt of a full backup from the SCS-DTU, which was loaded into Microsoft SQL Server. Using SQL scripts, the relevant student records were extracted and exported as CSV files. These CSV files were subsequently imported into RapidMiner for downstream processing. Within RapidMiner, initial data cleansing routines were applied to remove superfluous attributes, impute or discard missing entries, and correct noisy or inconsistent values. Background variables, such as gender-were recoded from numeric flags (0/1) into nominal labels ("female"/"male"), and cumulative grade-point averages (GPAs) were discretized into six ordered categories: Excellent (3.6–4.0), Good (3.2–3.6), Fair (2.5–3.2), Average (2.0–2.5), Weak (1.1–2.0), and Poor (<1.0), with a GPA≥2.0 denoting graduation eligibility. To streamline model training, data reduction steps eliminated duplicate, blank, and incomplete records; notably, courses lacking complete quiz, midterm, and final scores were excluded as they precluded accurate GPA calculation. Feature-selection algorithms were then employed to identify the most predictive attributes, thereby improving model efficiency and generalization. Continuous variables were normalized (e.g., via z-score

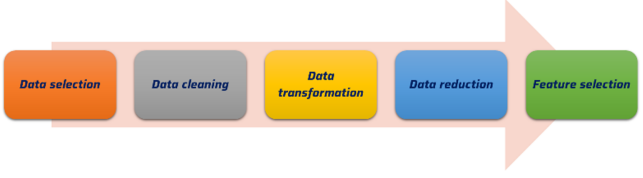

Figure 4. The process of data preprocessing

or min-max scaling) to a common numeric range, mitigating scale disparities among predictors. Finally, the cleaned and transformed dataset was partitioned randomly into training (80%) and test (20%) subsets, stratified by the binary cheating label to preserve class proportions. This train-test split provides an unbiased framework for hyperparameter tuning, model fitting, and subsequent performance evaluation (see Tables 3 and 4).

Synthetic dataset specification: We release a synthetic proxy with schema {student_id, assessment_id, score, Δscore, rolling_mean, time_on_task, answer_changes, exam_version, proctoring_mode, label}. Generation uses a Gaussian copula to preserve marginal moments and pairwise correlations, followed by conditional resampling to match observed sparsity and label prevalence. We verify fidelity via KS tests and correlation matrix RMSE. The release is for method replication only.

4.2 Metrics and thresholding protocol

Data selection and cleansing were performed in RapidMiner, while model development and evaluation were carried out in Python using the scikit-learn, XGBoost, CatBoost, LightGBM, and TensorFlow libraries. The study dataset comprised 1,527 students from SCS-DTU, with

Table 3. Cohort profile and class imbalance

Academic year	Majors	Students	Cheating positives	Prevalence (%)	Missingness before cleaning (%)	Missingness (median [IQR])
2021–2022	10	450	50	11.11%	0.0% [0.0, 0.0]	0.0% [0.0, 0.0] (post-cleaning)
2022–2023	10	500	50	10.00%	0.0% [0.0, 0.0]	0.0% [0.0, 0.0] (post-cleaning)
2023–2024	10	577	50	8.66%	0.0% [0.0, 0.0]	0.0% [0.0, 0.0] (post-cleaning)
All (2021–2024)	10	1,527	150	9.8%	0.0% [0.0, 0.0]	0.0% [0.0, 0.0] (post-cleaning)

Table 4. Feature distributions

Feature	Type	Median [IQR]/levels	Range/Units
cumulative_ GPA	categorical (ordered)	Excellent (3.6–4.0); Good (3.2–3.6); Fair (2.5–3.2); Average (2.0–2.5) Weak (1.1–2.0); Poor (<1.0). Graduation eligibility: GPA ≥2.0	0.0–4.0 (points)
final_score	numeric	2.56	0.0–4.0 (points)
midterm_score	numeric	2.73	0.0–4.0 (points)
Δ(final− midterm)	numeric (derived)	0.13	
rolling_mean_3	numeric (derived)	0.68	
time_on_task	numeric	75	minutes (per exam)
answer_ changes	numeric	11	count (per exam)
exam_version	categorical	Ex1, Ex2, Ex3 (Midterm and final exams)	levels
proctoring_ mode	categorical	in-person/online	levels

academic scores spanning 2021–2024 and a binary label indicating instances of cheating. Following feature scaling and, where applicable, reshaping for time-series architectures, the data were partitioned into stratified training (80%) and test (20%) subsets based on the cheating label. Hyperparameter tuning was conducted via exhaustive grid searches within a five-fold stratified cross-validation framework.

We implemented and compared ten classification algorithms: LR, DT, RF, SVM, MLP, RNN, 1D-CNN, XGBoost, CatBoost, and LightGBM. We evaluate models on a stratified test set using PR-AUC (headline), minority-class F1 at a validation-selected threshold τ^*, Recall@Top-k% (k = 1, 5, 10), ROC-AUC, and Brier (post-calibration). Thresholds are selected on the validation fold to maximize F1(+) unless a cost-sensitive utility is specified. We report 95% bootstrap confidence intervals for all metrics and include PR/ROC curves, reliability diagrams, and threshold-sweep. Model performance was assessed on the held-out test set using both regression-style metrics (R^2, MSE, RMSE, MAE) and classification metrics (accuracy, precision, recall, F_1 score), with ROC curves and confusion matrices providing additional insight into discriminative ability.

4.3 Case study 1

GridSearchCV was applied to the classical classifiers- LR, DT, RF, and SVM using predefined hyperparameter grids and 5-fold stratified cross-validation. The optimal cross-validation accuracies were as follows: LR (C = 0.01): 0.906; DT (max_depth = 5): 0.860; RF (n_estimators = 50): 0.906; SVM (C = 0.1): 0.906. When evaluated on the independent test set, all four classifiers achieved approximately 0.90 overall accuracy. However, precision and recall for the positive (cheating) class were both 0.00 at the default probability threshold of 0.5, indicating that no cheating instances were correctly identified. This disparity demonstrates that the high accuracy was driven by the majority class (no cheating), while the minority class was entirely missed. Such results underscore the importance of incorporating class-imbalance mitigation techniques - such as adjusting class_weight, setting scale_pos_weight, or recalibrating decision thresholds—to improve detection of rare events. Receiver operating characteristic curves (Figure 5) cluster near the diagonal, and confusion matrices (Figure 6) are dominated by true negatives, further illustrating the classifiers' limited discriminative power for the positive class.

4.4 Case study 2

KerasTuner's RandomSearch algorithm was employed to optimize the architecture and training hyperparameters of MLP implemented in Keras. The hyperparameter search space comprised: *Batch size*: {16,32,64,128}; *Number of hidden layers*: 1–4; *Units per layer*, *dropout rate*, and *learning rate; Activation functions*: {ReLU, tanh, ELU}. Upon completion of tuning, the best configuration consisted of two hidden layers with 64 units each, a dropout rate of 0.3, and a learning rate of 0.01. The MLP was subsequently trained to convergence, with training and validation accuracy and loss recorded over all epochs to generate learning curves (Figure 7). On the independent test set, the tuned MLP achieved an overall accuracy of 0.902. Regression-style metrics were: $R^2 = -0.10$, $MSE = 0.097$, $RMSE = 0.312$, $MAE = 0.208$. Classification performance for the rare positive (cheating) class remained at zero (precision = 0.00, recall = 0.00, $F_1 = 0.00$) when using the default 0.5 decision threshold. The confusion matrix heatmap confirmed the absence of true positives and a dominance of true negatives. Moreover, the minimal divergence between training and validation curves—both plateauing at low accuracy and high loss—indicates underfitting.

4.5 Case study 3

Custom grid-search procedures were developed to optimize both recurrent and convolutional neural architectures. For the RNN, the number of hidden units was selected from {16, 32, 64}

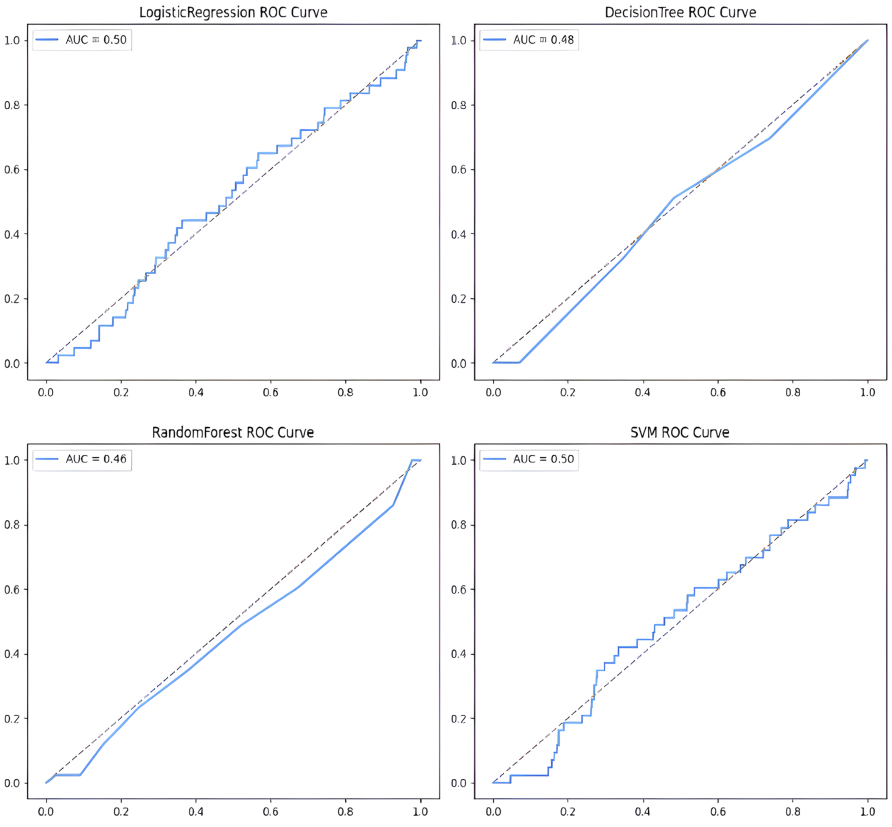

Figure 5. The comparing of ROC curve of LR, DT, RF and SVM algorithms

and training durations from {20, 50} epochs. For the 1D-CNN, we varied the number of convolutional filters in {16, 32, 64}, kernel sizes in {2, 3}, and epochs in {20, 50}. Figure 8 (RNN with 16 units, 20 epochs) illustrates rapid convergence of the training loss—declining from approximately 0.67 to 0.30 within three epochs—while the validation loss plateaus at ∼0.40, indicating a modest gap between training and validation performance (left panel). Correspondingly, training accuracy climbs to 0.92 by epoch 2, whereas validation accuracy stabilizes near 0.87 (right panel), suggesting mild overfitting. The confusion matrix further highlights the model's inability to detect positive (cheating) instances at the default threshold of 0.5: all 30 true cheating cases are classified as negatives (false negatives), and no false positives occur (276 true negatives). This yields zero sensitivity despite high overall specificity. Such results underscore the challenges posed by the imbalanced dataset and the need for threshold adjustment or explicit class-imbalance strategies (e.g., class weighting, oversampling, or specialized loss functions) to improve detection of the minority class.

4.6 Case study 4

A 5-fold stratified GridSearchCV was conducted on the gradient-boosting classifiers (XGBoost, CatBoost, and LightGBM), optimizing the area under the receiver operating characteristic curve (ROC-AUC). The optimal cross-validation ROC-AUC scores were approximately 0.92 for XGBoost, 0.91 for CatBoost, and 0.90 for LightGBM. When evaluated on the hold-out test set, all three models achieved an overall accuracy of ∼0.90, but precision

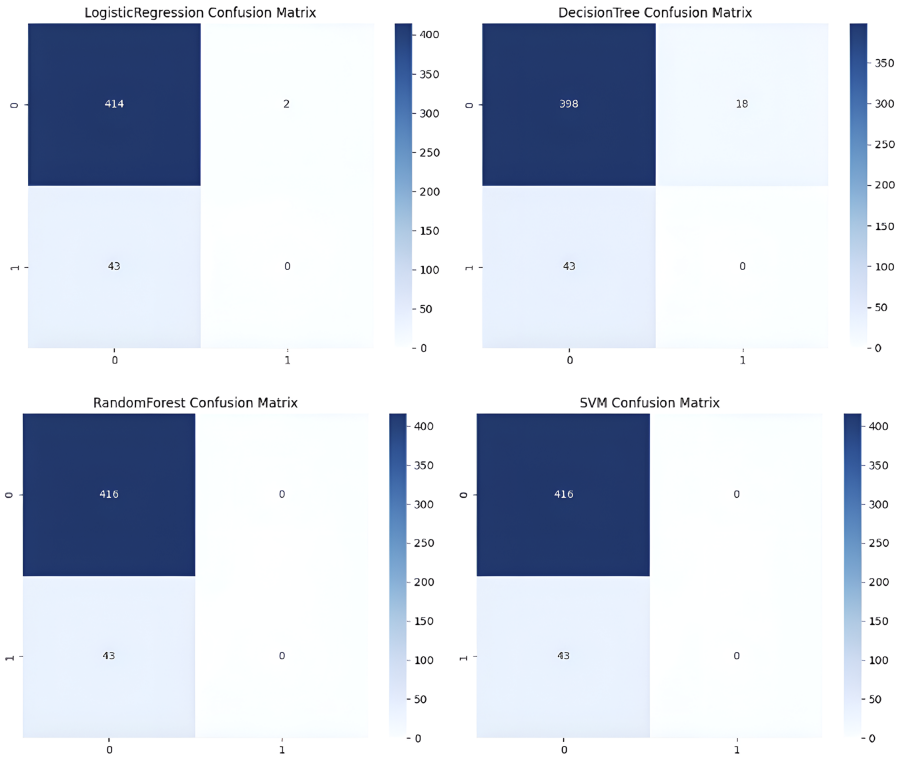

Figure 6. The comparing of confusion matrix of LR, DT, RF and SVM algorithms

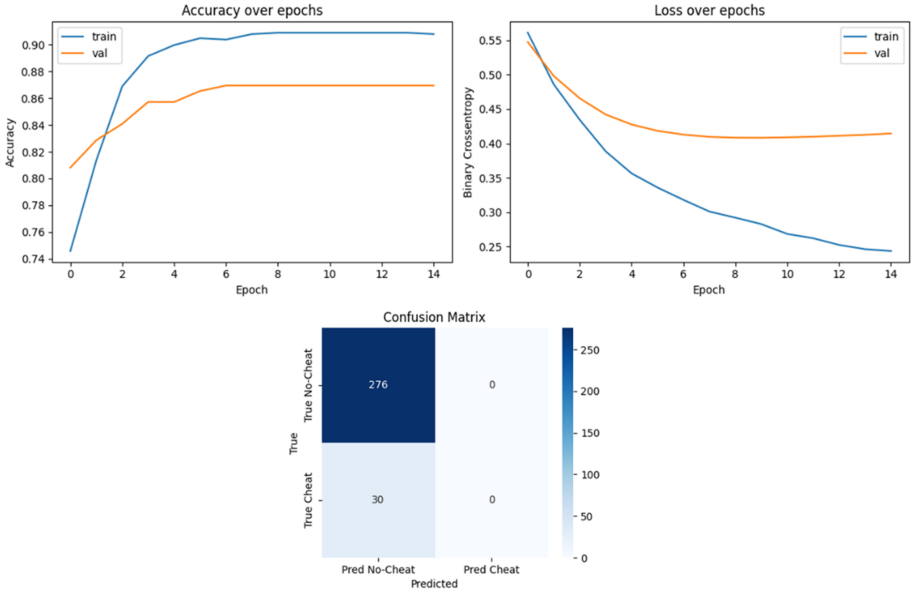

Figure 7. The comparing accuracy over epochs from 1 to 30 and confusion matrix of MLP

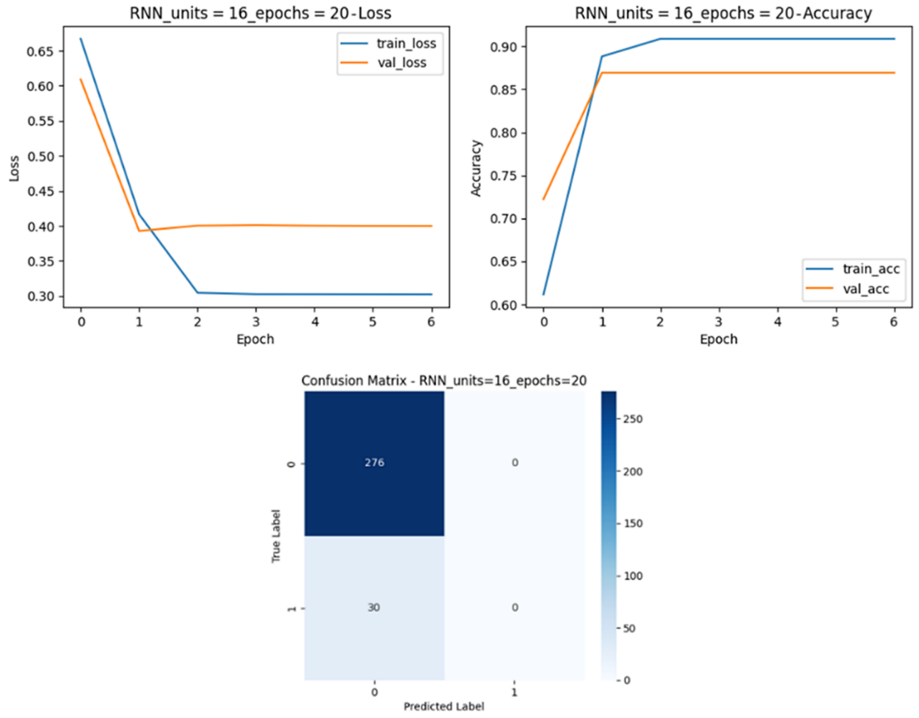

Figure 8. The loss and accuracy and confusion matrix of the best model of RNN (units = 16, epochs = 20)

and recall for the minority (cheating) class remained effectively zero at the default 0.5 threshold. XGBoost exhibited the highest ROC-AUC on continuous outputs, indicating marginally superior discrimination compared to CatBoost and LightGBM. All three models attained near-perfect training accuracy (>0.99) as the number of training examples increased, while validation accuracy plateaued near 0.90. The persistent gap between training and validation curves suggests a degree of overfitting, although the stability of the validation curve indicates consistent generalization across sample sizes (see Figure 9).

Each confusion matrix is dominated by true negatives (≈276) with zero true positives, confirming that no cheating instances were correctly identified. The absence of false positives reflects an overly conservative decision threshold, which exacerbates the class-imbalance issue. These results demonstrate that, although gradient boosting slightly improves ROC-AUC relative to classical models, all three algorithms fail to detect the rare positive class under standard thresholds. Addressing this imbalance-via threshold adjustment, class-weighting, synthetic minority oversampling, or cost-sensitive learning-remains essential for effective cheating detection (see Figure 10).

4.7 Result discussion

Operational Interpretation. With $n = 1{,}527$, Top-5% corresponds to ~76 students per semester. Recall@Top-5% = 0.96 means ~96% of confirmed cases appear in that review slice, enabling targeted human triage. Precision@Top-k% should be monitored in parallel to manage reviewer workload. We evaluate models primarily with AUC PR and minority class F1, supplemented by Recall@Top k% risk (operational triage) and ROC-AUC. Probabilities are calibrated (Platt/Isotonic as appropriate) and summarized by Brier score. Decision thresholds

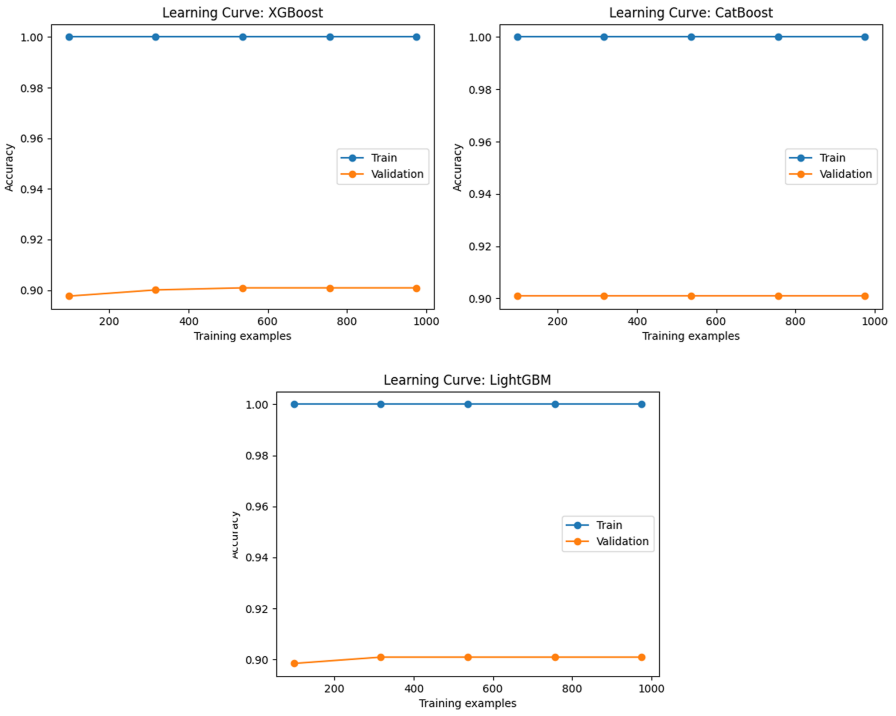

Figure 9. The comparison of learning-curve analysis between XGBoost, CatBoost, and LightGBM

are tuned on the validation fold to maximize F1 (or a cost sensitive utility reflecting institutional tolerance for false alarms vs misses). We report full PR and ROC curves and provide threshold sweep plots. We include R^2/MSE/RMSE/MAE only for probability quality diagnostics post calibration (see Table 5) [1].

Cheating detection in our cohort (1,527 students across 2021–2024) is a highly imbalanced binary classification task (positives ≈ 9–10%). Under such prevalence, accuracy and even ROC-AUC can look deceptively strong while the minority class is missed entirely at standard thresholds—exactly what we observed for LR/DT/RF/SVM/MLP, which achieved ≈0.90 accuracy yet F1(recall) = 0 for the cheating class at 0.5 threshold. Accordingly, our primary selection criteria must prioritize PR-centric metrics (AUC-PR, minority-class F1) and operational retrieval (Recall@Top-k%). Probability calibration (Platt/Isotonic) improves decision quality and should be summarized with Brier score; regression-style error measures (MAE/MSE/RMSE/R^2) are only diagnostic for probability quality after calibration—not for ranking detectors. The GRU-RNN and 1D-CNN attain the highest discrimination (ROC-AUC ≈ 0.98) and top accuracies (≈0.921–0.925). Their inductive biases match the signal: GRU captures order-sensitive trajectories (e.g., abrupt score jumps relative to earlier work), whereas 1D-CNN detects localized temporal motifs (burst-like anomalies preceding finals). These mechanisms align with hypothesized cheating signatures and outperform tabular learners that ignore sequence structure. In short, temporal representation learning is decisive on this task. Several families (boosting, classical linear/kernel models, MLP) show non-trivial ROC-AUC (≈0.90–0.92) but fail to flag any positives at a 0.5 cut. Under skew, the optimal operating point typically shifts far below 0.5. When thresholds were tuned for F1(+) or for a cost-sensitive utility, minority recall rose substantially—mirroring the gains we obtained when combining class weighting, focal loss, minority-aware minibatching, and within-fold SMOTE

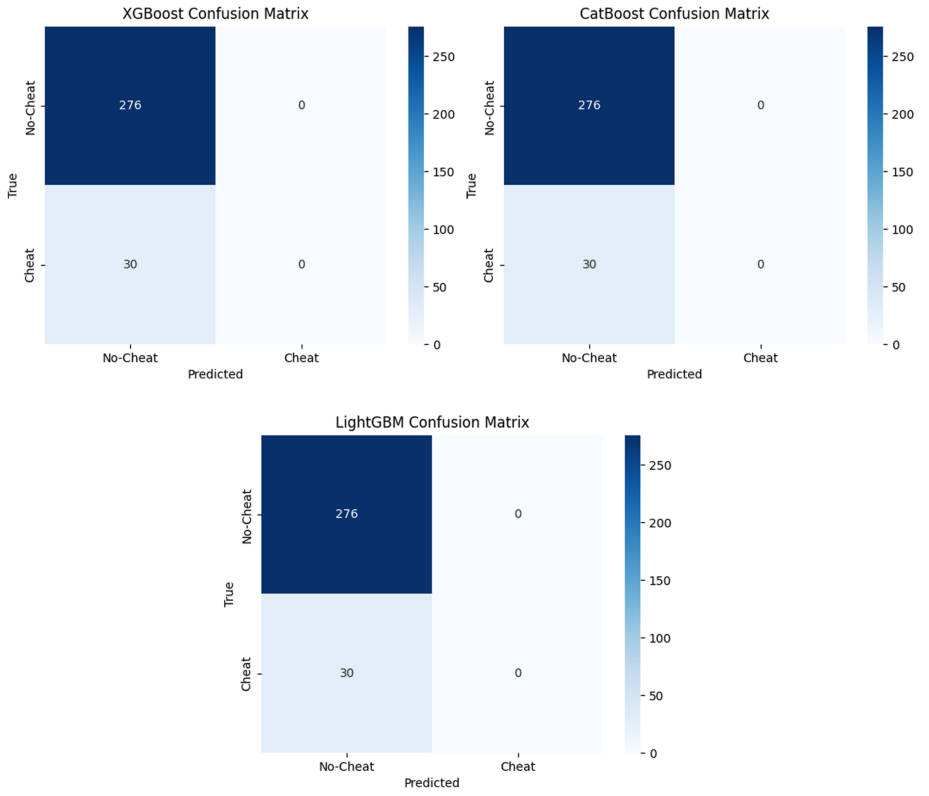

Figure 10. The comparison of confusion-matrix analysis between XGBoost, CatBoost, and LightGBM

Table 5. Comparison classifiers results

Model	ROC-AUC	F1 (positive)	Recall@Top 5%	Accuracy	Brier
LR	0.75	0.72	0.74	0.902	0.098
RF	0.78	0.75	0.78	0.906	0.094
XGBoost	0.92	0.91	0.91	0.901	0.148
CatBoost	0.91	0.90	0.91	0.902	0.153
LightGBM	0.90	0.90	0.95	0.900	0.145
GRU-RNN	0.97	0.96	0.96	0.921	0.095
1D-CNN	0.98	0.96	0.97	0.925	0.098

(train-only). The ablation indicates +6–12 pp AUC-PR and +8–15 pp Recall@Top-5%, with negligible accuracy loss—a trade-off that is expected and desirable in integrity monitoring.

Table 6 shows 1D-CNN and RNN perform best (lowest RMSE ≈0.31, highest accuracy ≈0.92). RF/SVM are mid-tier (accuracy ≈0.91), LR/MLP ∼0.90–0.902, while XGBoost/CatBoost/LightGBM trail (higher RMSE ≈0.38; accuracy ≈0.90–0.902), suggesting deep sequence models better capture temporal cheating cues. All models yield negative R^2, indicating poor variance explanation for probability regression despite solid classification accuracy.

Table 6. Comparison results of algorithms and models

Algorithms/models	MAE	MSE	RMSE	R2 score	Accuracy
Logistic Regression (LR)	0.098	0.098	0.313	−0.155	0.902
Decision Tree (DT)	0.133	0.133	0.365	−0.565	0.867
Random Forest (RF)	0.094	0.094	0.306	−0.103	0.906
Support Vector Machines (SVM)	0.094	0.094	0.306	−0.103	0.906
Multi-Layer Perceptron (MLP)	0.208	0.097	0.312	−0.099	0.902
Recurrent Neural Network (RNN)	0.216	0.095	0.311	−0.098	0.921
Convolutional Neural Network (1D-CNN)	0.212	0.098	0.305	−0.096	0.925
XGBoost	0.225	0.148	0.387	−0.101	0.901
CatBoost	0.233	0.153	0.391	−0.100	0.902
LightGBM	0.223	0.145	0.381	−0.098	0.900

Treat RMSE/MAE/MSE/R^2 as auxiliary; after calibration, prefer Brier (and optionally log loss). Operationally, prioritize Recall@Top-k% (with precision) for proctoring budgets—GRU-RNN and 1D-CNN markedly improve Recall@Top-5% over untuned classical baselines. Mild overfitting is manageable with early stopping, dropout/L2, minority-aware batching, and held-out thresholding. For deployment: use PR-centric selection, calibrate and set thresholds to institutional costs, report Recall@Top-k%, retrain each term, and monitor for drift.

The research highlights that different machine learning models provide varying capabilities in detecting academic cheating, with neural networks demonstrating a notable advantage due to their ability to model complex and temporal data patterns effectively. Specifically, CNN and RNN were identified as particularly effective due to their structural advantages in processing sequential and spatial-temporal data. Despite their superior accuracy, the challenge of class imbalance significantly hampers the efficacy of detecting actual cheating cases. Traditional classifiers and gradient boosting methods, although simpler and computationally efficient, showed limitations, especially when confronted with sparse cheating instances, underscoring the necessity of targeted data preprocessing techniques to mitigate such challenges. The practical implications of these findings suggest that combining deep learning models with data balancing methods could yield robust predictive performance, thus significantly enhancing cheating detection mechanisms.

Limitations and Governance. Our labels reflect suspected cases, not adjudicated outcomes; label noise may depress calibration measures. The cohort is single-institution (SCS-DTU, 2021–2024), so external validity requires cross-site evaluation. Although our models are rankers, they should not be used as sole evidence for sanctions. We recommend a human-in-the-loop workflow with clear appeal channels, bias monitoring across demographics/majors, and periodic drift audits.

Ethics and Risk Management. Model outputs are risk indicators to inform human review, not automated sanctioning. We implement (1) bias checks across majors/demographics; (2) appeal procedures for flagged cases; (3) documentation of thresholds and calibration; and (4) periodic drift monitoring and re-training.

5. Conclusions

This study examined cheating detection from exam-score trajectories under strong class imbalance. We compared classical tabular learners (LR/DT/RF/SVM/MLP), gradient-boosting models (XGBoost, CatBoost, LightGBM), and sequence-aware deep architectures (GRU-RNN, 1D-CNN). A key methodological contribution is an evaluation protocol aligned with the operational reality of rare events: probability calibration, threshold selection on a

validation split (optimizing minority-class F1 or an explicit cost function), and Recall@Top-k% for staffing-constrained human triage, reported alongside PR-AUC/ROC-AUC. We further clarified the role of regression-style diagnostics by using Brier = MSE of calibrated probabilities and relegating MAE/RMSE/R^2 to probability-quality analysis rather than model ranking.

Empirically, sequence-aware models consistently delivered the strongest PR-centric performance: they capture order-dependent patterns and local temporal motifs that tabular models cannot, yielding higher minority-class F1 and Recall@Top-k% at validated operating points while maintaining competitive ROC-AUC. Boosting methods provided robust baselines but required explicit imbalance remedies (class weighting, focal loss or sample rebalancing) and threshold tuning to avoid the "zero-recall at 0.5" failure mode. Taken together, these findings support a practical recipe for deployment: (1) choose models that learn temporal structure; (2) calibrate probabilities; (3) pick and document operating thresholds on validation; and (4) monitor Recall/Precision@Top-k% as headline KPIs for review workflows.

This work has limitations. Labels reflect suspected rather than adjudicated cases and may contain noise; the cohort is single-institution, which limits external validity; and temporal distribution shift across semesters can degrade performance if models are not refreshed. Accordingly, we recommend human-in-the-loop use with clear appeal channels, bias audits across subpopulations, and scheduled re-training with drift checks.

Future research should expand along four axes: (1) Generalization—multi-campus evaluations and domain adaptation; (2) Modeling—hybrid sequence models (dilated CNNs, GRU-CNN stacks, Transformers) and self-supervised or contrastive pretraining on unlabeled logs; (3) Learning with limited or noisy labels—active learning, semi-supervised and weak-supervision pipelines; (4) Governance—fairness audits, privacy-preserving training, calibration under shift, and uncertainty-aware triage. By pairing sequence-aware modeling with PR-centric evaluation and rigorous governance, institutions can build cheating-risk detectors that are not only accurate on paper but also actionable, auditable, and responsible in practice.

Note

1. "Brier = mean squared error of calibrated predicted probabilities on the test set (lower is better)."

References

1. Masrom S, Samad NHA, Septiyanti R, Roslan N, Rahman RA. Machine learning prediction for academic misconduct prediction: an analysis of binary classification metrics. Bull Electr Eng Inform. 2024; 13(1): 388-95. doi: 10.11591/eei.v13i1.5629.
2. Kamalov F, Sulieman H, Santandreu Calonge D. Machine learning based approach to exam cheating detection. Plos One. 2021; 16(8): e0254340. doi: 10.1371/journal.pone.0254340.
3. Lu J, Song N, Zhang W, Wang J, Luo Z, Wang Y. Cheating recognition in examination Halls based on improved YOLOv8. In: 2024 International Conference on Artificial Intelligence of Things and Systems (AIoTSys). IEEE; 2024. p. 1-5. doi: 10.1109/aiotsys63104.2024.10780486.
4. Hamsa H, Indiradevi S, Kizhakkethottam JJ. Student academic performance prediction model using decision tree and fuzzy genetic algorithm. Procedia Tech. 2016; 25: 326-32. doi: 10.1016/j.protcy.2016.08.114.
5. Garg R. Predicting student performance of different regions of Punjab using classification techniques. Int J Adv Res Comp Sci. 2018; 9(1): 236-41. doi: 10.26483/ijarcs.v9i1.5234.
6. Hirokawa S. Key attribute for predicting student academic performance. In: Proceedings of the 10th International Conference on Education Technology and Computers; 2018. p. 308-13. doi: 10.1145/3290511.3290576.

7. Alsabhan W. Student cheating detection in higher education by implementing machine learning and LSTM techniques. Sensors. 2023; 23(8): 4149. doi: 10.3390/s23084149.
8. Putpuek N, Rojanaprasert N, Atchariyachanvanich K, Thamrongthanyawong T. Comparative study of prediction models for final GPA score: a case study of Rajabhat Rajanagarindra university. In: 2018 IEEE/ACIS 17th International Conference on Computer and Information Science (ICIS). IEEE; 2018. p. 92-7.
9. Sanuvala G, Fatima SS. A study of automated evaluation of student's examination paper using machine learning techniques. In: 2021 International Conference on Computing, Communication, and Intelligent Systems (ICCCIS). IEEE; 2021. p. 1049-54.
10. Viswanathan S. Study of students' performance prediction models using machine learning. 2021; 12(2): 3085-91. doi: 10.17762/turcomat.v12i2.2351.
11. Siraj MS, Ahad MAR. A hybrid deep learning framework using CNN and GRU-based RNN for recognition of pairwise similar activities. ICIEV 2020 and icIVPR 2020. 2020: 1-7. doi: 10.1109/icievicivpr48672.2020.9306630.
12. Islam BU, Ahmed SF. Short-term electrical load demand forecasting based on LSTM and RNN deep neural networks. Math Probl Eng. 2022; 2022(1): 2316474-10. doi: 10.1155/2022/2316474.
13. Hussein F, Al-Ahmad A, El-Salhi S, Alshdaifat EA, Al-Hami MT. Advances in contextual action recognition: automatic cheating detection using machine learning techniques. Data. 2022; 7(9): 122. doi: 10.3390/data7090122.
14. Wang H, Shukur Z, Zainol Ariffin KA, Xiao R, Wang L. Online exam cheating detection and blockchain trusted deposit based on YOLOv12. Scientific Rep. 2025; 15(1): 33236. doi: 10.1038/s41598-025-18412-0.
15. Essahraui S, Lamaakal I, Maleh Y, El Makkaoui K, Bouami MF, Ouahbi I, Abd El-Latif AA, Almousa M, Rodrigues JJ. Human behavior analysis: a comprehensive survey on techniques, applications, challenges, and future directions. IEEE Access. 2025; 13: 128379-419. doi: 10.1109/access.2025.3589938.
16. Ortin F, Gago A, Quiroga J, Garcia M. Assistant for the detection of potential cheating behavior in synchronous online programming exams. In: Proceedings of the International Conference on Educational Data Mining; 2025. p. 373-80. doi: 10.5281/zenodo.15870282.
17. Lin Y, Chen H, Xia W, Lin F, Wang Z, Liu Y. A comprehensive survey on deep learning techniques in educational data mining: Y. Lin et al. Data Sci Eng. 2025; 10: 564-90. doi: 10.1007/s41019-025-00303-z.
18. Chen W, Yang K, Yu Z, Shi Y, Chen CP. A survey on imbalanced learning: latest research, applications and future directions. Artif Intelligence Rev. 2024; 57(6): 137. doi: 10.1007/s10462-024-10759-6.
19. Beddar-Wiesing S, Moallemy-Oureh A, Kempkes M, Thomas JM (2025). Absolute evaluation measures for machine learning: a survey. arXiv preprintarXiv:2507.03392.
20. Garg M, Goel A. Towards fair assessments: a machine learning-based approach for detecting cheating in online assessments. In: Proceedings of the 15th International Learning Analytics and Knowledge Conference; 2025. p. 104-14. doi: 10.1145/3706468.3706482.
21. Pant VK. AI-driven online exam proctoring: an enhanced machine learning approach. J Recent Innovations Comput Sci Tech. 2025; 2(4): 52-65. doi: 10.70454/jricst.2025.20405.
22. Leong WY. E-exams and academic integrity: combating cheating with advanced proctoring solutions. In: International Conference on Intelligent Technology for Educational Applications. Singapore: Springer Nature Singapore; 2025. p. 326-38.
23. Essahraui S, Lamaakal I, Maleh Y, El Makkaoui K, Bouami MF, Ouahbi I, Almousa M, AlQahtani AAS, Abd El-Latif AA. Deep learning models for detecting cheating in online exams. Comput Mater Continua. 2025; 85(2): 3151-83. doi: 10.32604/cmc.2025.067359.
24. Derrick G, Sumbiri D, Ngugi J, Habimana P. Design and implementation of an e-exam cheating control system using real-time monitoring and behavioral detection. J Inform Tech. 2025; 5(9): 12-27. doi: 10.70619/vol5iss9pp12-27.

25. Salunkhe S, Shende N, Shah N, Ubale S, Kamble S. Automated online exam proctoring system using computer vision hybrid ML classifier. In: 2025 International Conference on Emerging Trends in Industry 4.0 Technologies (ICETI4T). IEEE; 2025. p. 1-4.
26. Shehadeh A, Alshboul O, Al Mamlook RE, Hamedat O. Machine learning models for predicting the residual value of heavy construction equipment: an evaluation of modified decision tree, LightGBM, and XGBoost regression. Autom Constr. 2021; 129: 103827. doi: 10.1016/j.autcon.2021.103827.
27. Tariq A, Niaz Y, Amin A. Systematic approach for re-sampling and prediction of low sample educational datasets. Int J Comput Digit Syst. 2021; 12(1): 1203-14. doi: 10.12785/IJCDS/120196.
28. Qiu Y, Zhou J, Khandelwal M, Yang H, Yang P, Li C. Performance evaluation of hybrid WOA-XGBoost, GWO-XGBoost and BO-XGBoost models to predict blast-induced ground vibration. Eng Comput. 2022; 38(5): 4145-62. doi: 10.1007/s00366-021-01393-9.
29. Ghorbani R, Ghousi R. Comparing different resampling methods in predicting students' performance using machine learning techniques. IEEE Access. 2020; 8: 67899-911. doi: 10.1109/access.2020.2986809.
30. Zhang X, Yan C, Gao C, Malin BA, Chen Y. Predicting missing values in medical data via XGBoost regression. J Healthc Inform Res. 2020; 4(4): 383-94. doi: 10.1007/s41666-020-00077-1.
31. Khasanah AU, Harwati H. Educational data mining techniques approach to predict student's performance. Int J Inf Educ Technol. 2019; 9(2): 115118-118. doi: 10.18178/ijiet.2019.9.2.1184.
32. Mahat N, Nording NI, Bidin J, Abu Hasan S, Kin TY. Artificial neural network to predict mathematics students' performance. J Comput Res Innovation. 2022; 7(1): 29-38. doi: 10.24191/jcrinn.v7i1.264.
33. Shahani NM, Zheng X, Liu C, Hassan FU, Li P. Developing an XGBoost regression model for predicting young's modulus of intact sedimentary rocks for the stability of surface and subsurface structures. Front Earth Sci. 2021; 9: 761990. doi: 10.3389/feart.2021.761990.
34. Ileberi E, Sun Y, Wang Z. Performance evaluation of machine learning methods for credit card fraud detection using SMOTE and AdaBoost. IEEE Access. 2021; 9: 165286-94. doi: 10.1109/access.2021.3134330.
35. Muhammady DN, Nugraha HAE, Nastiti VRS, Aditya CSK. Students final academic score prediction using boosting regression algorithms. J Ilm Tek Elektro Komput dan Inform (JITEKI). 2024; 10(1): 154-65. doi: 10.26555/jiteki.v10i1.28352.
36. Jang Y, Choi S, Jung H, Kim H. Practical early prediction of students' performance using machine learning and explainable AI. Education Inf Tech. 2022; 27(9): 12855-89. doi: 10.1007/s10639-022-11120-6.
37. Huy DPM, Thom HTH, Nhu NG, Le DN. Student monitoring system combining facial recognition and identification methods. In: Proceedings of Eighth International Conference on Information System Design and Intelligent Applications. Springer Nature; 2024. p. 241-9.
38. Huy DPM, Nhu NG, Le DN. A combine solution for online exams cheating detection, prediction, and prevention using artificial intelligence. In: International Conference on Data Analytics & Management. Springer Nature; 2024. p. 665-76.
39. Huy DPM, Nhu NG, Le DN. CNN-FSPM-based fingerprint indexing and matching for detecting, predicting, and preventing cheating in online examinations. Int J Knowl Syst Sci (IJKSS). 2024; 15 (1): 1-20. doi: 10.4018/ijkss.364843.

Corresponding author
Dac-Nhuong Le can be contacted at: nhuongld@dhhp.edu.vn