Detection of Academic Dishonesty in Student Records Using Anomaly Detection with Deep Learning and Machine Learning Techniques

[1,*]Manit Malhotra, [2] Indu Chhabra

[1]Department of Computer Science & Applications, Panjab University, Chandigarh, India
Email: manitmalhotra@rediffmail.com

[2]Department of Computer Science & Applications, Panjab University, Chandigarh, India
Email: indu_c@pu.ac.in

Abstract

The rapid transition to online education, particularly during the COVID-19 pandemic, has raised critical concerns about maintaining academic integrity in online assessments. Traditional proctoring methods have proven insufficient, prompting exploration of advanced technological solutions. This paper investigates machine learning (ML) and deep learning (DL) techniques for detecting cheating behaviors using mark sheet data only, drawn from a cohort of 3,000 students across six semesters (three online, three offline) at Panjab University constituent colleges. The dataset exhibited a subtle vs blatant cheating imbalance ratio of approximately 3:2, handled via hybrid resampling. Models were evaluated using an 80/20 train-test split with bootstrapped confidence intervals. Among four models tested, the BiLSTM outperformed others, achieving 97.6% accuracy and an AUC of 1.00, offering absolute gains of +1.6% accuracy over LSTM and +22% over Random Forest. These findings highlight the potential of DL models for scalable, automated cheating detection; however, the scope remains limited to numerical mark sheet features from a single institution, warranting broader future studies.

Key Words: Anomaly Detection, Cheating Detection, Academic Dishonesty Detection , Deep Learning, Machine Learning

1 Introduction

The shift to online education, rapidly accelerated by the COVID-19 pandemic, has transformed traditional learning environments by offering unprecedented accessibility and flexibility [1]. However, this shift has also introduced substantial challenges for maintaining academic integrity, particularly in assessments conducted without direct invigilation [2]. Concerns about cheating during online exams have grown as students gain easier access to unauthorized resources or collaborative means that were harder to exploit in proctored, in-person settings [3], [4].

While numerous proctoring systems now leverage multimodal data streams, such as webcam feeds, eye-gaze tracking, or typing dynamics to curb dishonest behaviors, such approaches demand considerable infrastructure and raise privacy concerns [5], [6]. This study instead focuses on a simpler, cost-effective question: *Can temporal trends in students' mark sheet data alone reliably differentiate between subtle and blatant cheating behaviors, outperforming chance and traditional baselines?* By exploring purely numerical marks across online and offline semesters, we aim to evaluate whether even minimal data can provide an effective first line of detection.

1.1 Rise of Virtual Learning and Associated Cheating Trends

The expansion of online education is undeniable. The number of Americans enrolled in distance education courses rose by 93% between 2012 and 2019, according to research published by the National Center for Education Statistics (NCES) [7]. However, while this growth has enabled educational institutions to reach a wider audience, it has also created new vulnerabilities [8]. Without direct supervision, students are more likely to engage in dishonest practices such as using unauthorized materials, collaborating with peers, or even outsourcing their work to third parties [9]. The temptation to cheat has become more prevalent in online exams, where students may perceive that the risk of detection is lower compared to traditional, proctored exams [10], [11].

Research indicates that learners are more inclined to cheat when they think they can get away with it easily. The nature of online exams, which often allow students to take tests from the comfort of their homes, exacerbates this issue [12]. Other studies have found that the rate of academic dishonesty during online exams is significantly higher than in-person exams, particularly in assessments that do not employ stringent monitoring mechanisms. The same study highlighted the difficulty of preventing cheating in an environment where students can easily access external resources, share answers with peers, or utilize advanced technological tools to bypass detection [1].

1.2 Technological Approaches to Cheating Detection

The development of automated cheating detection systems has become a critical area of focus in educational research.

A key approach to detecting cheating involves the use of algorithms to identify anomalies in student behavior and performance. Levitt et al. [13] introduced a simple yet effective algorithm for identifying cheating by analyzing patterns of incorrect answers among students seated next to each other during in-person exams. Their findings demonstrated that matching incorrect answers was a more reliable indicator of cheating than matching correct answers, especially in scenarios where students were seated in proximity to one another. When applied to online exams, this approach can be adapted to detect patterns of anomalous behavior, such as unusually similar responses between students, which may indicate collusion [14]. Cheating detection has seen a rise in the usage of ML and DL methods in addition to algorithmic approaches. These advanced methods can be employed to analyze a wider range of behavioral indicators, such as eye movements, typing patterns, and camera recordings, to identify potential instances of cheating. The utilization of these techniques can help eliminate the need for manual review of student assessment sessions, thereby streamlining the process of detecting academic dishonesty. However, recent research [15], [14] has highlighted the potential for bias and fairness issues in the implementation of automated proctoring systems. Disparities have been observed in the accuracy of these systems across different racial, skin tone, and gender groups, raising concerns about the equitable treatment of students. To address these challenges, it is crucial for educators and researchers to carefully design and evaluate the algorithms and models used in cheating detection systems, ensuring that they are fair, accurate, and transparent.

In online learning environments, the challenge of detecting cheating is compounded by the lack of physical supervision. This has led to the exploration of ML techniques, particularly for identifying cheating behaviors in large datasets. One such study, conducted by Kamlov et al. [15], proposes an ML-based approach to detecting cheating using outlier detection methods. In their research, they treat the identification of potential cheating cases as an outlier detection problem, leveraging student assessment data to identify abnormal scores on final exams. They successfully identified instances of cheating by using techniques such as anomaly detection methods and AI methods such as Recurrent Neural Networks (RNNs). The use of sequential data analysis is particularly relevant in online exams, where the order and timing of answers can provide critical insights into whether a student has engaged in dishonest behavior.

While real-time surveillance through videos or images has become an increasingly popular way to detect cheating in online exams, it is not always feasible in all educational settings. In many cases, particularly where technological resources are limited, initial stages of cheating detection can be performed using readily available data such as student mark sheets and answer sheets. This form of analysis is particularly useful when there is no dataset available for real-time observation. By evaluating student performance data through traditional grading systems, we can identify anomalies in answer patterns or sudden deviations in performance that may indicate cheating behavior. This approach provides a practical and resource-efficient solution for detecting cheating before more advanced, system-based detection methods are implemented. Moreover, analyzing answer sheets and mark sheets allows educators to spot suspicious behavior early on, such as repeated patterns of similar answers between students or drastic improvements in final scores compared to earlier assessments. Such methods are indispensable in environments where technological tools for real-time monitoring are unavailable, making them an effective initial line of defense in maintaining academic integrity.

Previous studies have explored various methods for detecting academic dishonesty using numerical datasets like student mark sheets, quizzes, and exam scores [15], [16], [13]. These studies' method treated cheating detection as an outlier detection problem, where significant deviations between a student's final exam performance and earlier assessments were flagged as potential cheating cases. Also, some studies [13] detect cheating based on answer patterns in exams. They found that comparing incorrect answers between students seated nearby was a more reliable indicator of cheating than comparing correct answers [16]. These studies highlight the importance of using numerical datasets like mark sheets in the initial stages of detecting academic dishonesty, especially in cases where more advanced data like images, videos, or behavioral metrics are not available.

In this research, we aim to build upon these studies by analyzing student mark sheets from a multidisciplinary dataset within an educational institution. Our focus is to explore whether patterns in the mark sheets can reveal irregularities that point toward cheating behaviors, particularly by identifying statistical outliers and comparing student performance across multiple subjects. The dataset spans several disciplines and contains a wide range of student marks, which allows us to investigate cheating behaviors from different academic contexts. By examining this data and using just conventional evaluation data, our objective is to aid in the development of resource-efficient cheating detection techniques.

Additionally, unlike many existing approaches that focus solely on performance scores, our research integrates multi-modal indicators to capture a broader spectrum of cheating behaviors. This includes mechanisms to detect covert forms of dishonesty, such as collusion through similarity analyses among peer submissions, irregular access patterns suggestive of unauthorized resource use, and timing irregularities indicative of technological manipulation. These enhancements aim to ensure that both overt and subtle cheating tactics are systematically addressed.

Unlike multimodal proctoring systems that rely on video feeds, keystroke dynamics, or eye-tracking to detect cheating, our work focuses solely on analyzing numerical mark sheet data. While multimodal systems can capture richer behavioral cues and are covered extensively in Section 2 and Table 1, they also demand substantial infrastructure, raise privacy concerns, and may not be feasible in all educational contexts. By contrast, our study demonstrates that even traditional mark trends can

reveal subtle and blatant cheating behaviors, offering a practical, resource-light alternative that can serve as either a first-line screening tool or complement more intrusive proctoring systems.

1.3 Research Questions

To systematically guide this investigation, we articulate the following research questions (RQs), each aligned with our dataset, methodology, and evaluation framework:

1. **RQ1:** Can trends in mark sheet data alone, without any proctoring images or videos, effectively differentiate between non-cheating, subtle cheating, and blatant cheating cases?
2. **RQ2:** How do advanced deep learning models (LSTM, BiLSTM) compare to traditional machine learning models (Random Forest, Logistic Regression) in detecting performance anomalies indicative of cheating?
3. **RQ3:** What is the absolute improvement in accuracy and AUC offered by BiLSTM over other baseline models on this dataset?
4. **RQ4:** Does handling class imbalance through hybrid resampling techniques enhance the detection capability of cheating versus using imbalanced raw data?
5. **RQ5:** Can automated cheating detection based solely on mark trends achieve consistent agreement with expert human validation, supporting practical deployment?

1.4 Major Contributions

The principal contributions of this research can be summarized as follows:

1. We developed and systematically evaluated a cheating detection framework based exclusively on mark sheet data trends, demonstrating that even without proctoring videos or behavioral data, automated systems can flag subtle and blatant cheating with high reliability.
2. We conducted a comprehensive comparative study of traditional ML models (Random Forest, Logistic Regression) against advanced DL models (LSTM, BiLSTM), revealing that BiLSTM yields an absolute improvement of +1.6% accuracy over LSTM and +22% over RF, with perfect AUC in our experiments.
3. We employed robust hybrid class imbalance handling (SMOTE + Tomek Links) and demonstrated how it improves model sensitivity to minority (blatant cheating) cases, enhancing fairness and detection reliability.
4. We incorporated a human-in-the-loop validation stage, showing a 94% agreement between expert educators and automated BiLSTM predictions, thereby building trust for real-world deployment.
5. We provided extensive feature importance (for RF) and integrated gradients (for BiLSTM) analyses, along with fairness checks across academic programs, to ensure transparency, explainability, and ethical AI deployment.

The research is organized to systematically explore the various aspects of cheating detection through mark sheet data analysis. The introduction of the study is provided in Section 1. In Section 2, related work is reviewed with an emphasis on current approaches to academic dishonesty detection. Section 3 explained how the dataset was gathered for compiling multidisciplinary student mark sheet data. Section 4 described the Proposed methodology of the study, which briefly explained the process of data preprocessing and feature engineering, and showed how the data was prepared for analysis. The results are presented and discussed in Section 5, followed by an evaluation of the study's limitations in Section 6 and suggestions for future research in Section 7. Finally, Section 8 concludes the paper by summarizing the key findings.

2 Related Work

2.1 The Applications and Significance of ML and DL in Cheating Detection

To combat the rise of cheating in online assessments, educational institutions have increasingly turned to technological solutions. Using machine learning and DL algorithms, which can analyze massive amounts of data and spot trends that can point to dishonest behavior, is one of the most intriguing strategies [17], [18],[15]. A subfield of artificial intelligence called machine learning gives computers the ability to learn from data without the need for explicit programming. This makes it ideal for detecting cheating, as the algorithms can be trained to recognize suspicious behavior based on historical data [19].

Techniques related to ML have been effectively used in a number of industries, such as financial modelling, healthcare, and cheating detection. In the context of education, these techniques have been adapted to detect cheating by analyzing exam performance data, monitoring students' actions during exams, and identifying inconsistencies that may suggest dishonesty [20]. For instance, these algorithms can be used to compare a student's performance in online and offline exams, flagging significant discrepancies that may indicate cheating. Additionally, these models can be applied to identify patterns of suspicious behavior, such as frequent switching between exam windows or prolonged inactivity followed by rapid answering of questions [15],[21].

DL, a more advanced subset of ML, is particularly effective in cheating detection because of its ability to analyze complex, unstructured data such as images and video. Deep learning models such as convolutional neural networks (CNNs) and long short-term memory (LSTM) networks have been applied in the context of online exam monitoring. These models are capable of analyzing visual cues like facial expressions, gaze direction, and behavioral patterns that may indicate potential academic misconduct. These models can be trained

to detect subtle changes in a student's behavior, providing a more comprehensive view of potential cheating activities [5], [20],[22].

2.2 Existing Approaches to Cheating Detection

The literature on cheating detection is extensive, with researchers exploring various methods to address academic dishonesty. Early approaches relied on rule-based systems, which used predefined criteria to flag suspicious behavior. For example, systems might identify students who submit exams significantly faster than average or those who exhibit unusual patterns of behavior during an exam. While these methods were effective to some extent, they were limited by their reliance on static rules, which could be easily bypassed by students who were aware of the system's criteria [23],[24],[1],[25].

The existing literature on cheating detection methods highlights the evolution of techniques across various domains, including text, numerical data, and coding submissions. AlSallal et al. [26] emphasize the limitations of traditional plagiarism detection tools, which often fail to recognize reworded or summarized texts. They introduce a comprehensive method that integrates Bag of Words (BoW), Latent Semantic Analysis (LSA), Stylometric Features, and Support Vector Machines (SVM) to identify complex instances of plagiarism. The effectiveness of this approach in recognizing nuanced linguistic patterns was demonstrated through experiments on the Corpus of English Novels (CEN). Similarly, Atoum et al. [5] address the challenges of cheating in online exams, proposing an automated detection system that integrates hardware (webcams, microphones) and software (gaze estimation, voice detection) to analyze audio-visual data in real-time, significantly reducing the reliance on human proctoring.

Nagoudi et al. [27] focus on Arabic texts, where traditional tools fail to identify synonym substitution and text manipulation effectively. They propose two methods, one leveraging word embeddings and the other using ML classifiers like SVM and RF to detect disguised plagiarism with high precision and recall scores, using the EXARA-2015 dataset. Meanwhile, Qiubo et al. [28] tackle code plagiarism, introducing a hybrid approach that combines RF and gradient boosting decision trees. Their methods adapt to variations in programming style and submission behavior, achieving a notable accuracy rate of 95.9% in identifying plagiarized code. These innovative approaches demonstrate the growing sophistication in detecting various forms of cheating, offering solutions that extend beyond simple similarity thresholds to incorporate behavioral and semantic analysis across diverse data types.

More recent approaches have incorporated ML and DL techniques, which offer greater flexibility and accuracy. These techniques make it possible to analyze big datasets, including student performance records, behavioral data, and even biometric information. By leveraging these data sources, these algorithms are able to identify cheating tendencies that would be difficult to find using conventional techniques. For example, a study by Tiong et al. [1] demonstrated the efficiency of a DL based method for detecting cheating in online exams. Their system, which analyzed both behavioral and performance data, achieved a high level of accuracy in identifying students who engaged in dishonest practices [29]. A primary benefit of using ML and DL in cheating detection is the capacity for ongoing enhancement of the system's accuracy over time. As more data are collected, the algorithms can be retrained to better recognize cheating behaviors. This allows the system to adapt to new forms of cheating, making it more difficult for students to find ways to circumvent detection [4].

In addition to the ML and DL approaches used in the detection of cheating behaviors during exams, many researchers have used IoT technologies to detect these kinds of behavior. IoT technology offers a budget-friendly, flexible, and easy-to-use solution to detect cheating in online examinations. These systems rely on simple devices like webcams, microphones, and internet access to function. The growth of AI-based proctoring tools has become especially significant during the COVID-19 pandemic, as digital learning has gained widespread use around the world [30].

Recent advancements in online proctoring and cheating detection have introduced a variety of innovative systems and methodologies. Shevale et al. [31] developed a web-based application using the MERN stack, enhancing accessibility for disabled users with features like voice navigation, while also focusing on securing online exams during the pandemic. Ho et al. [32] created RAPID, a proctoring solution using Raspberry Pi to monitor students' computer activity, helping prevent cheating through advanced security measures. Nguyen et al. [6] developed a system that leverages IoT and AI, utilizing dual cameras and AI-driven analysis for real-time fraud detection in online exams, achieving high accuracy. La Roca et al. [33] examined students' experiences with online proctoring, finding that while most have adequate resources, some face challenges that affect their performance. Plochaet et al. [34] investigated video anomaly detection to monitor academic integrity, employing video and voice detection models to spot cheating, although the results showed some room for improvement. Atabay et al. [35] studied BeatGAN, an ML model for time series anomaly detection, which showed promise but struggled with subtle cheating behaviors. Finally, Bommireddy et al. [36] designed an AI-driven system that monitored students through video and audio inputs to detect cheating, reducing the reliance on human proctors. Table 1 represents the summary.

Table 1: Summary of Cheating Detection Approaches

Reference No.	Characteristics	Models Used	Limitations
(Gruenigen et al. [23], Chuang et al. [25], Keresztury et al. [24])	Early approaches relied on rule-based systems, using predefined criteria to flag suspicious behavior, such as unusually fast exam submissions.	Rule-Based Systems	Static rules could be bypassed by students familiar with the system, limiting effectiveness.
(AlSallal et al. [26])	Focus on detecting sophisticated plagiarism attempts using a multifaceted approach.	BOW, LSA, SVM	Traditional tools fail to recognize reworded or summarized texts, and performance depends on text quality.
(Atoum et al. [5])	Automated system integrating hardware (webcams, microphones) and software (gaze estimation, voice detection) for real-time analysis.	Gaze Estimation, Voice Detection	Relies on external devices and real-time monitoring, may raise privacy concerns or require infrastructure.
(Cherroun et al. [27])	Focus on Arabic texts for detecting disguised plagiarism using word embeddings and ML classifiers.	Word Embeddings, SVM, Random Forest	Ineffective at detecting synonym substitution or manipulation in non-English.
(Qiubo et al. [28])	Hybrid approach combining RF and GBDT for code plagiarism detection.	RF, GBDT	Challenges with highly sophisticated or novel cheating techniques.
(Tiong et al. [1])	Recent methods using ML and DL to detect cheating via performance and behavioral data.	DL	Requires significant data for training, newer tactics may bypass detection.
(Zhao et al. [4])	ML/DL systems retrain on new data to adapt to evolving behaviors.	ML, DL	Continuous retraining needed, models may become outdated if methods change.
(Nigam et al. [30])	IoT offers a cost-effective, flexible solution using webcams/mics.	IoT Systems	Potential false positives and limited by simple devices.
(Shevale et al. [31])	MERN stack web app for accessible, secure exams during the pandemic.	MERN stack	May face accessibility/security challenges at scale.
(Ho et al. [32])	RAPID system uses Raspberry Pi to monitor computer activity.	Raspberry Pi	Hardware constraints, may struggle with many students.
(Nguyen et al. [6])	IoT + AI dual cameras for real-time fraud detection.	Dual Cameras, AI	Costly at scale, struggles with complex cheating.
(Roca et al. [33])	Study on student experiences highlights resource challenges.	Online proctoring	Insufficient resources may cause false negatives.
(Plochaet et al. [34])	Video + voice detection for academic integrity.	Video anomaly, Voice detection	Needs improvement for subtle cheating.
(Atabay et al. [35])	BeatGAN for time-series anomaly detection.	BeatGAN	Needs optimization for real-time, misses subtle cheating.
(Bommireddy et al. [36])	AI system monitors via video/audio for cheating.	AI Video/Audio	Raises privacy concerns, may miss sophisticated cheating.
(Roumiana et al. [17], Fakhroddin et al. [18])	ML/DL detect cheating in online assessments.	ML algorithms	Depend on quality/quantity of historical data.
(Sarker et al. [19])	ML learns from data, ideal for detecting cheating.	ML	May not generalize well to new cheating types.
(Kaddoura et al. [20])	ML detects cheating via performance/action data.	ML	May struggle with sophisticated strategies.
(Kamlov et al. [15], Balderas [21])	Compares online vs offline performance, flags suspicious patterns.	ML	May flag non-cheating behaviors.
(Faucher et al. [22])	DL analyzes unstructured data (images/video) for cheating.	CNNs, LSTM	Computationally intensive for real-time.
(Cizek and Wollack [37])	Reviews statistical indices for test score cheating.	Statistical Indices (K-index, GBT)	Needs item-level data, assumes independence.

Table 2: Comparison of Datasets and Validation Strategies Across Studies

Study	Sample Size	Data Modalities	Validation Protocol	Remarks
Atoum et al. [5]	∼100 students	Webcam Video, Gaze, Voice	Expert labeling, no split reported	Requires external hardware
Cherroun et al. [27]	2,000 text pairs	Arabic text documents	Train-test split, no stratification	NLP-specific, not exam-based
Tiong et al. [1]	300+ sessions	Screen logs, Webcam, Responses received	Manual review, 5-fold CV	Multimodal behavioral dataset
Our Proposed Model	**2,931 records**	**mark sheet data (numeric)**	**Stratified CV**	**Imbalanced multiclass-to-binary setting**

Although the integration of IoT, AI, and other technologies for cheating detection is promising, there are several challenges to be addressed. First, technological limitations can lead to false positives or negatives. For example, a student may be flagged for cheating due to nervous habits, such as fidgeting or looking around the room, even when no dishonest behavior occurs. However, sophisticated cheating methods, such as the use of hidden ears or advanced signal jamming techniques, can evade detection. Second, the cost of deploying IoT infrastructure and AI systems at scale can be prohibitive for many educational institutions, especially in resource-constrained environments. The complexity of integrating different technologies into a cohesive cheating detection framework also requires significant expertise and investment.

While Table 1 summarizes methodological differences among approaches, we now extend the comparison with a dedicated overview of sample sizes, feature modalities, and validation strategies. This helps contextualize the relative difficulty of our setting, which relies solely on numerical mark sheet data without multimodal augmentation. As seen in Table 2, many prior studies used behavioral video/audio inputs or interaction logs, often with smaller or curated datasets, while our setting involves over 2,900 labeled student records across six semesters and three disciplines with class imbalance, making it a non-trivial detection challenge.

While much recent work on cheating detection emphasizes multimodal data such as video or biometric streams, a parallel line of research, statistical forensics, continues to investigate cheating via performance data alone. For example, Cizek and Wollack [37] provide a comprehensive overview of statistical indices (e.g., K-index, GBT) designed to flag collusion or answer copying based on anomalous score distributions and response patterns.

3 Dataset

In order to understand potential cheating behaviors among students, this section offers a comprehensive examination of the dataset used in our study. The dataset comprises detailed academic records from multiple undergraduate programs, including Bachelor of Commerce (B.Com.), Bachelor of Business Administration (B.B.A.), and Bachelor of Computer Applications (B.C.A.), offered across various constituent colleges of Panjab University, Chandigarh, India. This study focuses on student cohorts from the academic years 2020 to 2023, a period that captures the global transition from virtual learning during the COVID-19 pandemic to the subsequent resumption of in-person education.

The dataset covers six semesters for each student, with a notable division in exam formats: the first three semesters were conducted online, while the final three semesters were administered in traditional offline settings. Each semester's data includes student names, roll numbers, and the marks obtained in six subjects. This comprehensive

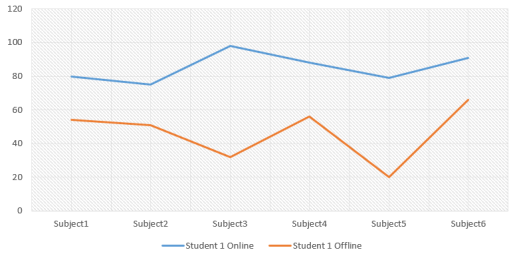


Figure 1: Anomaly Detection Graph for Online and Offline Exams of a Student

structure allows for a comparative analysis of performance across different exam modes.

The primary motivation for selecting this dataset is to investigate whether the switch to online exams may have contributed to irregularities in student performance, potentially indicative of cheating. By systematically comparing students' marks in online versus offline exams, we aim to identify patterns that suggest dishonest behavior. Figure 1 illustrates the creation of a line graph to visualize potential discrepancies between online and offline exams. The plotted graph, with separate lines, represents a single student's mark scenario for both offline and online exams. The graph indicates a general downward trend in marks over time, potentially due to factors like increased difficulty or decreased student effort. However, a significant disparity is observed between online and offline exams in all six subjects. This suggests the possibility of cheating or other irregularities in the online exam environment.

To facilitate supervised learning and behavioral analysis, we established a set of labeling rules based on the mark differences observed between online and offline exams for the same student. These rules are designed to reflect varying levels of potential academic dishonesty:

1. **Non-Cheating:** Students whose mark differences between online and offline exams are less than 25 points are labeled as "Non-Cheating", as they fall outside the defined suspicious thresholds associated with potential misconduct.
2. **Subtle Cheating:** Students whose online exam marks exceed their offline marks by 25 to 29 points are labeled as "Subtle Cheating", indicating possible minor or opportunistic misconduct.
3. **Blatant Cheating:** Students whose mark difference is 30 points or more are labeled as "Blatant Cheating", reflecting likely instances of significant academic dishonesty.

The dataset initially comprises 3,000 anonymized student records. These records span various programs and semesters, enabling the analysis of cheating patterns across disciplines and timeframes. We specifically focused on students who had appeared for both online and offline versions of the same or equivalent examinations between 2020 and 2023. Based on the above labeling criteria, we divided the dataset into three classes Non-Cheating, Subtle Cheating, and Blatant Cheating to facilitate multiclass classification and behavioral profiling. This structure allows us to investigate how the abrupt shift to virtual learning (and its eventual reversal) may have influenced academic integrity at scale.

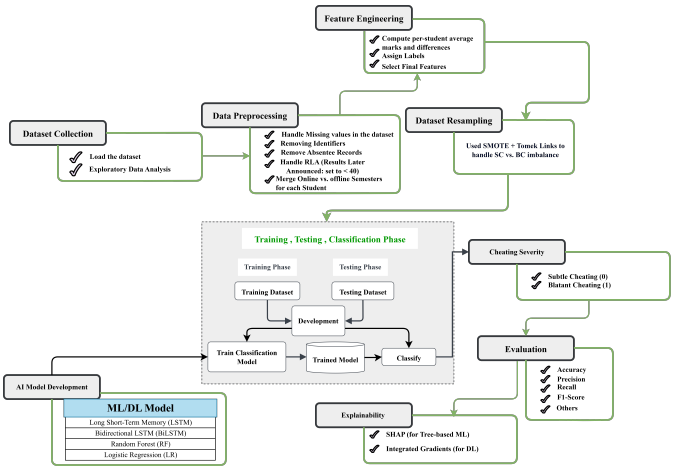


Figure 2: Development Pipeline of the Proposed Methodology

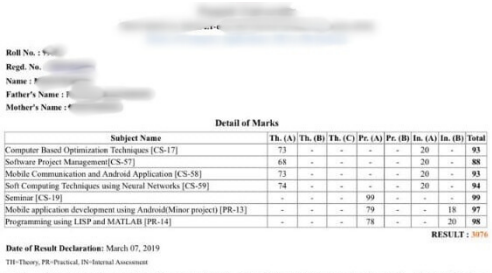
4 Proposed Methodology

The proposed methodology for this study revolves around classifying cheating and non-cheating instances using a carefully curated dataset. This methodology is structured into three major phases: preprocessing, feature engineering, and model training and testing. Several steps are involved in each phase to ensure efficient dataset processing, appropriate feature engineering, and accurate predictions from the classification model.

Our approach aims to comprehensively capture and identify cheating behaviors by analyzing discrepancies in students' marks between online and offline exams while addressing challenges such as missing data and class imbalances. The proposed methodology is structured into preprocessing, feature engineering, resampling, and training/testing phases using both ML and DL models, with explainability and fairness considerations integrated into the evaluation process shown in Figure 2.

4.1 Data Collection

The dataset contains academic records from undergraduate programs, including B.B.A., B.Com., and B.C.A., at Panjab University, Chandigarh, India. For the dataset collection, the website https://results.puexam.in/ has been accessed for publicly available datasets of the above-mentioned departments. It focuses on the academic years 2020 to 2023, covering six semesters for each student. The first three semesters were conducted online, and the final three were held offline. The data includes student names, roll numbers, and marks obtained in six subjects, allowing for a comparative performance analysis across different exam modes. Initially, each student's mark sheet was displayed as shown in Figure 3 and was later consolidated into an Excel file by combining all students' records into a single file. To maintain the transparency of the dataset and maintain the privacy of the students we removed the student names and roll numbers from our final dataset file as illustrated in Figure 4.



Roll No. :
Regd. No.
Name :
Father's Name :
Mother's Name :

Detail of Marks

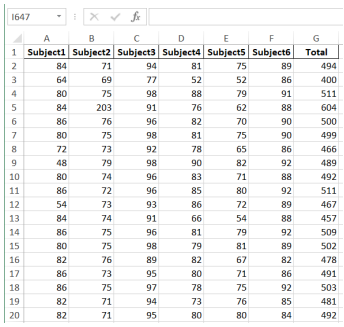
Subject Name	Th. (A)	Th. (B)	Th. (C)	Pr. (A)	Pr. (B)	In. (A)	In. (B)	Total
Computer Based Optimization Techniques [CS-17]	73	-	-	-	-	20	-	93
Software Project Management[CS-57]	68	-	-	-	-	20	-	88
Mobile Communication and Android Application [CS-58]	73	-	-	-	-	20	-	93
Soft Computing Techniques using Neural Networks [CS-59]	74	-	-	-	-	20	-	94
Seminar [CS-19]	-	-	-	99	-	-	-	99
Mobile application development using Android(Minor project) [PR-13]	-	-	-	79	-	-	18	97
Programming using LISP and MATLAB [PR-14]	-	-	-	78	-	-	20	98

RESULT :

Date of Result Declaration: March 07, 2019

TH=Theory, PR=Practical, IN=Internal Assessment

Figure 3: The Representation of Student's Mark Sheet Collected from the University's Website



	A	B	C	D	E	F	G
1	Subject1	Subject2	Subject3	Subject4	Subject5	Subject6	Total
2	84	71	94	81	75	89	494
3	64	69	77	52	52	86	400
4	80	75	98	88	79	91	511
5	84	203	91	76	62	88	604
6	86	76	96	82	70	90	500
7	80	75	98	81	75	90	499
8	72	73	92	78	65	86	466
9	48	79	98	90	82	92	489
10	80	74	96	83	71	88	492
11	86	72	96	85	80	92	511
12	54	73	93	86	72	89	467
13	84	74	91	66	54	88	457
14	86	75	96	81	79	92	509
15	80	75	98	79	81	89	502
16	82	76	89	82	67	82	478
17	86	73	95	80	71	86	491
18	86	75	97	78	75	92	503
19	82	71	94	73	76	85	481
20	82	71	95	80	80	84	492

Figure 4: Representation of the Transparent Dataset

4.2 Pre-Processing of Datasets

Data preprocessing is an essential stage that provides the structure for the next steps. In this stage, in order to improve the quality and quantity of the dataset, we performed data collection, data augmentation, handled values that were missing, and combined datasets before preparing the data for analysis.

4.2.1 Merging Discipline Datasets

The first task in preprocessing was to merge the datasets of three different disciplines: B.B.A, B.COM., and B.C.A., each dataset comprises student records across six semesters, with the same number of subjects in each discipline. The motivation behind merging these datasets is to increase the sample size, which enhances the study's generalizability and resilience. The merging process of the datasets allows us to analyze student behavior on a larger scale and draw more reliable conclusions regarding cheating patterns across different fields of study.

4.2.2 Merging Online and Offline Examination Records

To facilitate a clear comparison between online and offline examination performance, we consolidated the first three semesters (online exams) and the last three semesters (offline exams) of the dataset. This step ensures the following aspects:

Direct Comparison: Merging online and offline exam records allows for direct comparisons between the two exam modalities. This is critical because the study's primary goal is to identify discrepancies in student performance that may indicate cheating in the more lenient environment of online exams compared to the more controlled offline exams.

Uniform Dataset: Consolidating the data creates a uniform dataset that provides a holistic view of each student's performance across both modalities. This uniformity simplifies the subsequent analysis, ensuring that all records are complete and comparable across all six semesters.

Identifying Cheating Behaviors: This study's main objective is to identify patterns of cheating in exams. Merging the two types of exam records allows for the calculation of performance differences between the online and offline exams, which can help flag students who might have exploited the online format for dishonest gains.

4.2.3 Handling Missing Values

The dataset presented instances of incomplete entries, primarily stemming from gaps in student academic records. Rather than discarding such records, which could reduce the overall sample size and weaken the analysis, we adopted structured imputation methods to manage missing data effectively.

For continuous attributes such as exam scores, we applied mean imputation to estimate and replace missing values with the average of available entries. This approach helps maintain the overall distribution and avoids introducing systematic bias. In the case of categorical fields, mode imputation was used, substituting missing values with the most frequently occurring category, thereby preserving class balance and consistency.

Handling missing data appropriately is crucial to preventing distortion in statistical outcomes. If left unaddressed, missing values, especially if concentrated among specific student groups (e.g., high or low achievers) can lead to biased insights. By using imputation, we ensured that the dataset remains as complete and representative as possible, which is particularly important when working with finite or institution-specific datasets.

4.2.4 Handling Absentee Records

Students who were absent from exams have complete null records in the dataset. The absence of marks or performance data in absentee records renders them irrelevant for the core analysis. Further, including these records would introduce noise into the dataset, making it more difficult to train accurate models and extract meaningful patterns.

This step also ensures that the dataset is focused exclusively on students who participated in both online and offline exams. This focus enhances the quality of the analysis and ensures that the models are trained on relevant data. Eliminating absentee records helps streamline the dataset, making it easier to manage and reducing computational overhead during the analysis. When dealing with large datasets, this phase becomes essential since it maximizes performance during model training.

4.2.5 Handling Result Later Announced (RLA) Values

The dataset contains entries labeled as RLA, signifying a delay in the announcement of exam results, often due to supplementary exams. These cases usually imply that the student had to retake the exam, which suggests weaker academic performance. To maintain consistency by replacing RLA records with a score of less than 40 marks, we ensure that the dataset remains consistent and that these records accurately represent the student's academic ability.

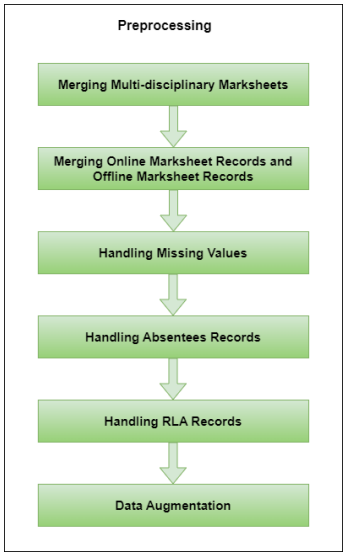


Figure 5: Representation of Pre-Processing Steps

This step is also used to handle anomalies. RLA values are considered anomalies in the dataset because they do not follow the typical pattern of exam results. By assigning a standard score to these anomalies, we ensure that the dataset remains consistent and that students with supplementary exams are properly accounted for in the analysis. Students with RLA records are likely to have different performance trajectories compared to those who passed their exams on the first attempt. By assigning them a low score, we ensure that these students are not unfairly compared to students who performed better on their initial exams.

4.2.6 Data Augmentation

To strengthen the dataset and enhance the model's ability to generalize, data augmentation techniques were applied. This process involves artificially expanding the dataset by generating new instances derived from existing records, thereby increasing both the volume and diversity of training data. By introducing slight variations such as modifying input patterns or creating synthetic examples, we aimed to expose the model to a broader range of potential student behaviors and exam-related conditions. This diversity helps the model become more resilient when encountering subtle shifts in data during real-world deployment.

Moreover, data augmentation serves as a preventive measure against overfitting, a common issue when working with limited datasets. Overfitting occurs when a model becomes too specialized in the training data, resulting in poor performance on unseen inputs. Through augmented data, we introduce variability that encourages the model to learn more general patterns, ultimately improving its adaptability and predictive accuracy on new student records. A detailed figure of the pre-processing step is shown in Figure 5.

4.2.7 Statistical Evaluation and Confidence Intervals

To robustly quantify model reliability, we employed bootstrapped sampling (1,000 resamples) to estimate 95% confidence intervals for accuracy, precision, recall, and AUC scores. These intervals provide an empirical sense of variability and statistical significance across different training samples.

4.3 Feature Engineering

After preprocessing, we moved on to the feature engineering phase, where we designed and selected the features that would be most relevant to classifying cheating instances. Feature engineering is essential to transforming raw data into a form that can be effectively used by machine learning models. In this phase, we calculate key metrics such as mark differences, averages, and flagging instances based on these differences. Figure 6 illustrates a detailed representation of the steps involved in feature engineering.

4.3.1 Calculation of Average Marks per Semester

The first step in feature engineering involved calculating the average marks of each student for each semester. This provides an overall metric of student performance, which can be used as a baseline to compare how much their performance fluctuates between online and offline exams. The semester-wise averages serve as a key feature in the model as they highlight discrepancies in academic performance.

4.3.2 Calculation of Marks Differences Between Semesters

To identify potential cheating behaviors, we computed the differences between the marks of the online and offline semesters. This step is crucial for quantifying how much a student's performance changed between these two exam modalities. The assumption is that significant increases in marks during online exams, followed by a drop during offline exams, may indicate cheating.

4.3.3 Calculation of Overall Average and Difference in Final Scores

In addition to semester-wise differences, we computed each student's overall average across all semesters, as well as the overall difference between online and offline exams. This helps to capture broader patterns in student performance that individual semester comparisons may miss. The overall difference metric provides an additional layer of insight into whether the performance deviations are consistent with the flagging criteria for cheating.

4.3.4 Flagging and Labeling Instances

Based on the calculated differences, we labeled each student record as Subtle Cheating (SC), Blatant Cheating (BC), or Non-Cheating (NC). The labeling process follows these rules:

1. **Non-Cheating (NC):** If the mark differences fall outside these ranges, the record is labeled as Non-Cheating.
2. **Subtle Cheating (SC):** If the difference in marks between an online and offline exam for a given subject is between 25 and 29 points, the record is flagged as Subtle Cheating.
3. **Blatant Cheating (BC):** If the difference in marks between online and offline exams is 30 points or higher, the record is flagged as Blatant Cheating. This multiclass labeling provides a foundation for identifying patterns in cheating behavior and serves as the target variable for our classification models.

4.3.5 Binary Classification Transformation

Although the dataset initially contains three labels (SC, BC, NC), our focus is primarily on classifying the cheating instances. To simplify the classification task and avoid issues such as overfitting or underfitting commonly associated with multiclass problems, we removed the records labeled as Non-Cheating. After removing NC cases, we transformed the remaining dataset into a binary classification problem, where:

- Subtle Cheating (SC) is labeled as **0**.
- Blatant Cheating (BC) is labeled as **1**.

This binary format enables us to train models specifically on cheating behaviors, improving the models' ability to distinguish between subtle and blatant cheating patterns. Although we excluded NC (Non-Cheating) cases from the model training to create a binary classifier focusing specifically on differentiating between Subtle and Blatant Cheating patterns, the broader dataset still contains all students, including honest cases. In a real deployment, the trained model would serve as a focused anomaly detector: auditors apply it to the entire student population, where NC students would not meet the threshold differences and thus receive low predicted probabilities for cheating, effectively screening them out. This strategy concentrates the model's sensitivity on distinguishing the severity of flagged anomalies, supporting high-efficiency investigations without direct multiclass separation.

4.3.6 Dataset Balancing

After completing the feature engineering process and removing the non-cheating (NC) cases, we ended up with a dataset containing 2,931 records that were labeled as either Subtle Cheating (SC) or Blatant Cheating (BC). This dataset, however, was imbalanced, meaning one class had significantly more instances than the other. To ensure that the model performs well across both labels and is not biased towards one class, dataset reduction and balancing steps were implemented. Addressing Class Imbalance. Addressing class imbalance is important due to many reasons. In classification tasks, imbalance of class is a prevalent issue when the dataset's label distribution is distorted [38]. In our case, there were a higher number of SC cases compared to BC cases. This imbalance can negatively impact model performance in the following ways:

- **Biased Predictions:** Biased predictions might result from an ML model that is trained on an unbalanced dataset, which favors the majority class. For example, if subtle cheating cases significantly outnumber blatant cheating cases, the model might classify most instances as subtle cheating, even when there are blatant cheating cases present.
- **Poor Generalization:** While training on an unbalanced dataset, a model may perform well, but it may find it difficult to generalize to new, unseen, and untested data. This is because the model has not been adequately exposed to the minority class, causing it to underperform when encountering these instances in real-world applications.
- **Metric Degradation:** Evaluation metrics such as accuracy may provide misleading results on imbalanced datasets. For instance, when the minority class is poorly anticipated, a high accuracy may be attained by consistently forecasting the majority class. Therefore, when evaluating model performance in the context of

class imbalance, other measures, including accuracy, recall, F1-score, and AUC-ROC, become more important.

We used resampling methods to balance the dataset and provide a more equal distribution of SC and BC labels in order to overcome these problems.

4.3.6.1 Oversampling the Minority Class

Oversampling, in which more samples of the minority class are created to balance the class distribution, is one of the most often used techniques for resolving class imbalance [39]. In our case, if there were significantly fewer BC cases compared to SC cases, we used oversampling techniques to generate synthetic BC samples. These additional instances help prevent the model from being overly influenced by the more prevalent SC cases. Techniques used for oversampling are the following:

- **SMOTE (Synthetic Minority Over-sampling Technique):** Through the process of transforming between preexisting minority class samples, SMOTE creates artificial instances of the minority class. By using this method, overfitting is avoided and the variety of the synthetic samples is increased [40]. In our case, SMOTE was applied to create additional BC records that are not exact duplicates but interpolations between existing instances.
- **Random Oversampling:** Random oversampling involves duplicating minority class samples at random until the class distribution is balanced [41]. This simple technique was also tested in conjunction with SMOTE to ensure that there was adequate representation of BC cases in the dataset.

4.3.6.2 Undersampling the Majority Class

Undersampling the majority class rather than oversampling the minority class may work better in certain situations [42]. This method reduces the number of majority class instances (SC in our case) to balance the dataset. Although the dataset's size is reduced by undersampling, it can help mitigate overfitting by forcing the model to focus on a smaller, more balanced set of examples. Techniques used for undersampling are the following:

- **Random Undersampling:** In order to balance the distribution of classes, this method eliminates instances of the majority class at random [43]. In our case, a portion of the SC records was randomly eliminated to create a dataset that is more equal without creating synthetic examples.
- **Tomek Links**: Tomek Links are feature space pairings of instances that are near to one another yet belong to different classes (SC and BC). By identifying and removing these links, we eliminate borderline cases that are difficult to classify and contribute to class imbalance. This helps to improve the separability between the two classes while reducing the majority class size [44].

4.3.6.3 Hybrid Resampling Techniques

To enhance model performance and mitigate the impact of class imbalance, a hybrid resampling strategy was adopted by combining both oversampling and undersampling techniques. Specifically, Synthetic Minority Over-sampling Technique (SMOTE) was employed to generate synthetic instances of the minority class, thereby improving its representation. Subsequently, Tomek Links and random undersampling were applied to reduce redundancy and noise within the majority class. This combined approach helped achieve a more balanced and cleaner dataset, maintaining the integrity of real-world distributions while reducing the risk of model bias toward dominant classes.

4.3.7 Dataset Reduction

In addition to balancing the dataset, dataset reduction is necessary when dealing with noisy or irrelevant data that may hinder model performance. After applying feature engineering and generating synthetic instances, we conducted a thorough evaluation of the dataset to identify:

- **Outliers:** We detected and removed outliers, such as students whose performance deviated significantly from the rest of the dataset. These outliers could distort the model's understanding of cheating behaviors and lead to inaccurate predictions [45].
- **Irrelevant Features:** Certain features may not contribute meaningfully to the classification task and can be removed to reduce dimensionality. This step enhances model efficiency and prevents overfitting.
- **Dimensionality Reduction Techniques:** Methods like Principal Component Analysis (PCA) were considered to further reduce the feature space. PCA projects high-dimensional data into a lower-dimensional space while retaining most of the variance in the data. By applying PCA, we could condense the feature set into a smaller, more manageable form without sacrificing valuable information.

4.3.8 Balancing and Reduction Results

After implementing resampling techniques and reducing the dataset, we achieved a balanced dataset with an approximately equal number of SC and BC records. This balanced dataset provided several key advantages:

- **Improved Model Generalization:** The dataset is now prepared on a more balanced set of instances, which enhances its capacity to apply generalization to new data. It can more properly categorize examples of both subtle and blatant cheating and is less likely to be biased towards the majority class.
- **Better Evaluation Metrics:** Evaluation metrics like accuracy, recall, and F1-score become reliable measures of model performance when the dataset is balanced. These metrics reflect the model's ability to correctly identify both classes without being overly influenced by the more frequent class.
- **Reduced Overfitting:** By removing irrelevant data and applying hybrid resampling techniques, we minimized the risk of overfitting, which is a common problem when dealing with imbalanced datasets. The model is now better equipped to make accurate predictions on unseen data.

To ensure that synthetic samples generated by SMOTE and random oversampling do not appear in both training and test sets, we first performed an 80/20 split on the original dataset. Only the training set then underwent oversampling to balance SC and BC cases. This approach avoids any overlap of synthetic data between training and evaluation phases, maintaining the integrity of test performance metrics such as AUC and ensuring that model generalization is fairly assessed.

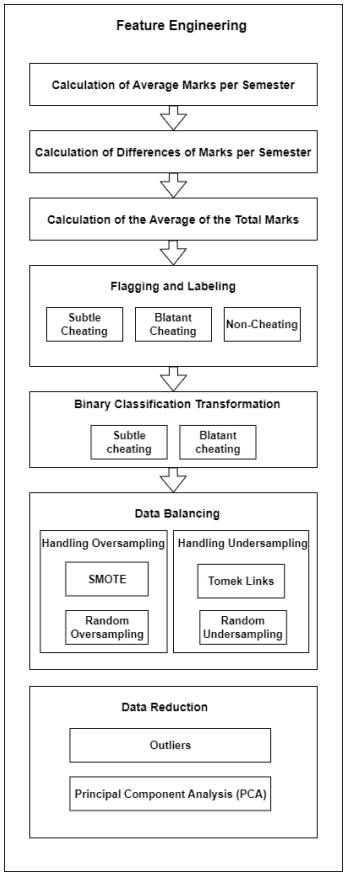


Figure 6: A detailed representation of Feature Engineering steps.

Finally, we validated balancing effectiveness by comparing pre and post-resampling class distributions. The final balanced dataset achieved an approximate 1:1 ratio of subtle vs blatant cheating cases, significantly improving recall for the minority class.

4.4 Training and Testing of Models

The final phase of the methodology involves training and testing the classification models on the processed and feature-engineered dataset. Here, we use two of the ML models in addition to two of the DL models for the classification of data and predict whether a student has engaged in cheating behavior. The objective is to identify the most reliable and precise model by assessing each model's performance.

4.4.1 Splitting of Datasets

To evaluate model performance effectively, the dataset was divided into training and testing subsets. An 80/20 split was employed, allocating 80% of the data for model training and the remaining 20% for testing. This approach ensures that the model is trained on a sufficiently large sample while preserving a separate portion for unbiased evaluation.

The training set is used to learn the underlying patterns in the data, whereas the testing set provides an independent benchmark to assess the model's ability to generalize to previously unseen records. This division is essential for validating the robustness and reliability of the proposed models.

4.4.2 Machine Learning Models

The RF Classifier and LR were the two ML models used in this work to categorise the dataset and predict student cheating behaviour.

4.4.2.1 Random Forest (RF)

Random Forest is a widely used ensemble learning algorithm that builds multiple decision trees during the training phase and combines their outputs to make final predictions through majority voting. Unlike a single decision tree, which can easily overfit, Random Forest introduces randomness by selecting different subsets of features and samples for each tree, enhancing model generalization and reducing variance [15], [46].

In our implementation, the Random Forest classifier was configured with n_estimators = 100, meaning the model consists of 100 decision trees. A fixed random_state = 42 was applied to ensure reproducibility across experiments. The chosen number of estimators reflects a balance between computational efficiency and predictive accuracy, too few trees may result in underfitting, while an excessive number could lead to unnecessary computational overhead without significant performance improvement.

The dataset was split into training and testing subsets using an 80/20 ratio. This standard partitioning approach provides ample data for model training while retaining a separate portion for unbiased evaluation of performance. Key evaluation metrics included accuracy, precision, recall, F1-score, and the Area Under the Receiver Operating Characteristic Curve. The ROC-AUC score is particularly informative, as it captures the classifier's effectiveness in distinguishing between subtle and blatant forms of academic dishonesty. Additionally, a confusion matrix was constructed to provide detailed insight into the model's classification accuracy across true positives, false positives, true negatives, and false negatives.

The ensemble structure of the RF model offers notable advantages, particularly in scenarios involving noise or class imbalance. By aggregating predictions from multiple independently trained trees and utilizing bootstrap sampling alongside random feature selection, Random Forest mitigates the influence of outliers and reduces model variance. This results in more consistent and reliable performance, positioning RF as a strong candidate for detecting academic misconduct. A visual representation of the Random Forest architecture is provided in Figure 7.

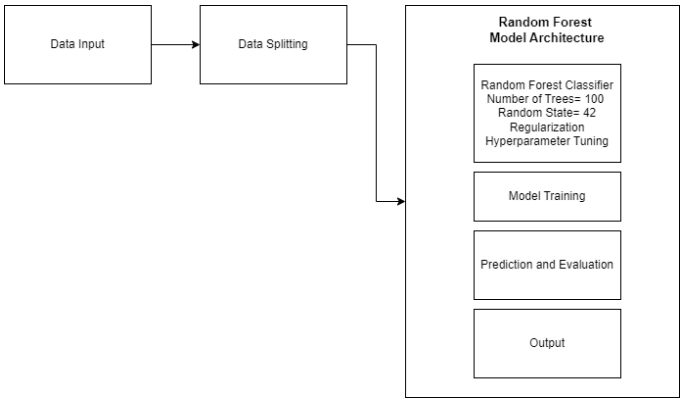


Figure 7: The Architecture of Random Forest (RF) model.

The algorithm for this ML technique is given below:

1. Import libraries: `pandas`, `numpy`, `LogisticRegression`, `accuracy_score`, `classification_report`, `roc_curve`, `roc_auc_score`, and `matplotlib`.
2. Load the dataset and separate features (`X`) and labels (`y`).
3. Set a random seed for reproducibility.
4. Split the data into training and testing sets.
5. Initialize and train the Logistic Regression model.
6. Make predictions on the test data (both class labels and probabilities).
7. Evaluate the model: Calculate accuracy and generate a classification report.
8. Plot and save the confusion matrix and a bar graph of model accuracy.
9. Compute the ROC curve (FPR, TPR, AUC), and save the ROC plot.
10. Display the results: Model accuracy and classification report.

4.4.2.2 Logistic Regression (LR)

Logistic Regression (LR) is a widely used linear classification algorithm that estimates the probability of a binary outcome based on one or more input features. It is particularly effective when there is a linear association between the independent variables and the log-odds of the dependent variable. The model computes a weighted sum of the input features and applies the logistic (sigmoid) function to map the result into a probability between 0 and 1, representing the likelihood that a given instance belongs to a particular class [47].

In this study, the LR model was implemented with a maximum iteration limit of 1000 to ensure proper convergence, especially in the presence of complex or near-linearly separable data. A fixed random state of 42 was used to guarantee reproducibility across multiple runs. The dataset was split into training and testing sets using an 80/20 ratio, consistent with the procedure followed for the Random Forest model.

One of the key advantages of Logistic Regression is its transparency. The model's coefficients provide clear insights into the relationship between each feature and the target variable, indicating both the magnitude and direction of their influence. This interpretability makes LR especially valuable in educational and behavioral analytics, where understanding the reasoning behind predictions is essential [48].

To prevent overfitting, L2 regularization (also known as Ridge regularization) was applied. This approach penalizes large coefficient values and is particularly useful when working with datasets containing multicollinearity or many input variables. In this work, the default regularization strength was used, offering a balanced trade-off between bias and variance [49].

The model's performance was assessed using standard evaluation metrics, including accuracy, precision, recall, F1-score, and ROC-AUC. The ROC-AUC score is of particular importance in Logistic Regression, as it evaluates the model's ability to rank predictions correctly across both classes. Additionally, since LR outputs probabilities, classification thresholds can be adjusted to optimize for specific use cases or to address class imbalance. The structure of the Logistic Regression model is depicted in Figure 8.

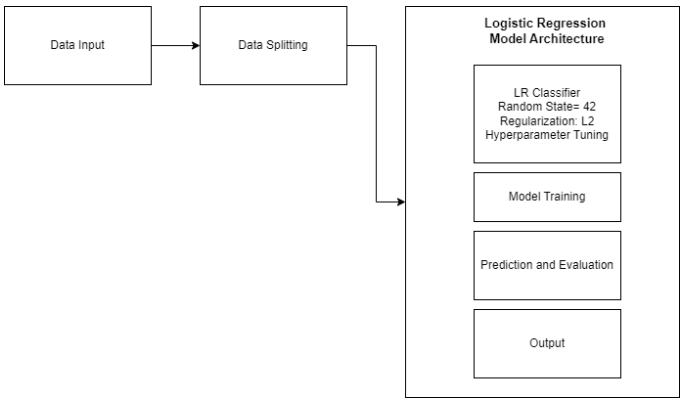


Figure 8: The architecture of the LR model employed in this study.

The detailed algorithm for this ML technique is as follows:

1. Import libraries: `pandas`, `numpy`, `RandomForestClassifier`, `accuracy_score`, `classification_report`, `roc_curve`, `roc_auc_score`, `confusion_matrix`, `ConfusionMatrixDisplay`, and `matplotlib`.
2. Load the dataset and separate features (`X`) and labels (`y`).
3. Set random seed for selected labels.
4. Split the data into training and testing sets (80-20 split).
5. Initialize and train a Random Forest model on the training data.
6. Make predictions on the test data (class labels and predicted probabilities).
7. Evaluate the model: Calculate accuracy, generate a classification report, and compute the confusion matrix.
8. Plot and save the following:
   (a) Accuracy bar graph
   (b) ROC curve (FPR, TPR, AUC)
   (c) Confusion matrix
9. Display results: Print the accuracy and classification report.

4.4.3 Deep Learning Models

4.4.3.1 LSTM Model

Long Short-Term Memory (LSTM) networks are a specialized type of Recurrent Neural Network (RNN) designed to address the limitations of traditional RNNs, particularly the vanishing gradient problem that occurs when learning from long sequences. LSTMs leverage internal memory cells along with input, forget, and output gates, enabling them to retain and update relevant information over extended sequences. This makes them particularly effective for sequence-based tasks such as temporal pattern recognition and natural language modeling [50], [51].

In our experiment, the LSTM model was built with a recurrent layer containing 164 units. The ReLU activation function was used to introduce non-linearity and facilitate the learning of complex relationships in the input data. Model weights were initialized using the He Uniform initializer, which is optimized for ReLU-based networks to maintain stable signal propagation.

To reduce the risk of overfitting, especially given the inherent noise and moderate size of the dataset, we applied L2 regularization with a weight decay factor of 0.01. In addition, dropout was introduced as

a regularization technique: a dropout rate of 0.9 followed the LSTM layer, while a rate of 0.7 was applied after the dense layer. These high dropout rates encourage generalization by randomly deactivating a significant portion of neurons during training, minimizing dependency on specific features or pathways.

Following the recurrent layer, a fully connected (dense) layer with 128 neurons was added, also configured with ReLU activation and L2 regularization. The final output layer comprised a single neuron with a sigmoid activation, producing a probability score to classify input samples as belonging to either class.

The model was trained using the Adam optimizer, selected for its ability to adaptively tune learning rates and handle sparse gradients effectively. A conservative learning rate of 0.0001 was chosen to ensure smooth convergence. The binary cross-entropy loss function was used, given its suitability for binary classification problems by effectively measuring the divergence between predicted probabilities and true labels.

To optimize the learning trajectory, a custom learning rate schedule was implemented. The learning rate was gradually increased during the first 50 epochs to assist the model in escaping shallow local minima, followed by an exponential decay phase to refine convergence. An early stopping strategy with a patience value of 10 was employed to terminate training when the validation loss failed to improve, preventing overtraining and saving computational resources.

The model was set to train for a maximum of 150 epochs, although early stopping often concluded training earlier. Performance was assessed through a comprehensive set of evaluation metrics, including accuracy, precision, recall, F1-score, and the ROC-AUC score, which is particularly useful for evaluating binary classifiers.

To monitor training behavior and model effectiveness, plots showing accuracy and loss over epochs were generated. Furthermore, the ROC curve and confusion matrix were constructed to provide deeper insights into classification quality. The structure of the implemented LSTM model is depicted in Figure 9.

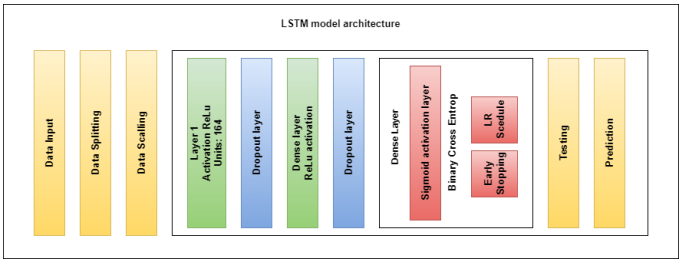


Figure 9: Proposed model of LSTM

The algorithm for the proposed LSTM model is as follows:

1. Import libraries: `pandas`, `numpy`, `train_test_split`, `StandardScaler`, `metrics`, `tensorflow` (LSTM, Dense, Dropout), and `matplotlib`.
2. Load the dataset and separate features (`X`) and labels (`y`).
3. Define a function to add noise to the features and apply it three times.
4. Split data into training and testing sets (80-20 split, stratified).
5. Scale features using `StandardScaler`.
6. Reshape data for LSTM input format: (samples, 1, features).
7. Define a learning rate scheduler:
   - Increase learning rate by 4% for the first 50 epochs.
   - Apply exponential decay afterward.
8. Build the LSTM model:
   - LSTM layer: 164 units, ReLU activation, He initialization, L2 regularization.
   - Dropout layer: 90% dropout rate.
   - Dense layer: 128 units, ReLU activation, He initialization, L2 regularization.
   - Dropout layer: 70% dropout rate.
   - Output Dense layer: 1 unit, sigmoid activation, L2 regularization.
9. Compile the model: Use Adam optimizer (learning rate = 0.0001), binary cross-entropy loss, and accuracy metric.
10. Apply early stopping: Monitor validation loss, stop if no improvement for 10 epochs, and restore the best weights.
11. Train the model: Use 150 epochs with learning rate scheduler and early stopping, validating on the test data.
12. Make predictions on the test data and compute probability estimates.
13. Evaluate the model:
    - Compute accuracy, ROC AUC, precision, recall, and F1 score.
    - Compute weighted accuracy, precision, recall, and F1 score.
14. Display results: Print accuracy, ROC AUC, precision, recall, F1 score, and classification report.
15. Plot and save the following:
    - Accuracy and loss graphs over epochs.
    - Confusion matrix.
    - ROC curve (FPR, TPR, AUC).

4.4.3.2 BiLSTM Model

BiLSTM networks extend the standard LSTM architecture by incorporating two parallel processing layers: one that reads the input sequence in its original (forward) order and another that reads it in reverse. This dual-directional structure allows the model to capture contextual information from both past and future time steps simultaneously, which is particularly beneficial for sequence-based tasks where temporal dependencies exist in both directions [1], [52].

In this study, the BiLSTM model was designed with an architecture similar to the LSTM model but modified to accommodate bidirectional processing. The main BiLSTM layer consisted of 164 units and utilized the ReLU activation function. Weights were initialized using the He Uniform initializer, and L2 regularization with a penalty of 0.01 was applied to reduce overfitting while maintaining model stability.

To promote better generalization, dropout layers with a rate of 0.5 were placed after both the BiLSTM and dense layers. Following the recurrent component, a dense layer with 128 units was added, employing the same ReLU activation and L2 regularization as in the LSTM configuration. The output layer consisted of a single neuron with a sigmoid activation function to produce a probability score for binary classification. Notably, the L2 regularization strength for the output layer was slightly reduced to 0.001, granting the model greater adaptability during final decision-making.

The model was compiled using the Adam optimizer with a learning rate of 0.0001, and binary cross-entropy was selected as the loss function, which is appropriate for binary classification tasks. A custom

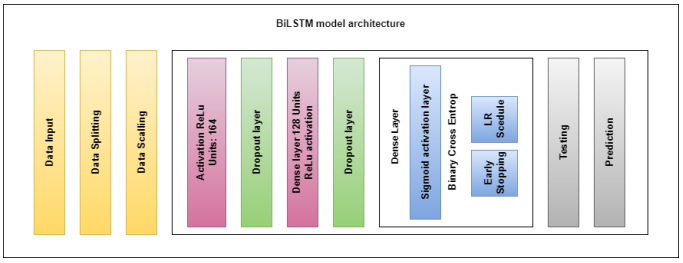


Figure 10: Brief Representation of the Proposed BiLSTM Model.

learning rate scheduler was implemented, gradually increasing the learning rate over the first 80 epochs before applying exponential decay to refine learning in later training stages. Early stopping was also employed with a patience value of 10 epochs, enabling the training process to halt automatically when no further improvement in validation loss was observed.

Training was conducted for up to 150 epochs, subject to early stopping based on validation performance. The model was evaluated on the test set using a comprehensive set of metrics, including accuracy, precision, recall, F1-score, and ROC-AUC, as well as their weighted counterparts to account for potential class imbalance. Visualizations such as accuracy and loss curves, the ROC curve, and the confusion matrix were generated to provide deeper insights into the model's learning dynamics and classification behavior. The architecture of the proposed BiLSTM model is illustrated in Figure 10.

Notably, we use the same layers, units, and hyperparameters for both DL models, LSTM and BiLSTM, to see the difference in outcomes. The algorithm for the BiLSTM model is as follows:

1. Import libraries: pandas, numpy, train_test_split, StandardScaler, performance metrics, tensorflow (LSTM, Bidirectional, Dense, Dropout), and matplotlib.
2. Load dataset and separate features (X, y).
3. Split data into training and testing sets (80-20 split).
4. Scale features using StandardScaler.
5. Reshape data for LSTM input format: (samples, 1, features).
6. Define a learning rate scheduler to adjust learning rate per epoch.
7. Build BiLSTM model:
   - BiLSTM layer: 164 units, ReLU activation, He initialization, L2 regularization.
   - Dropout layer: 50% dropout.
   - Dense layer: 128 units, ReLU activation, He initialization, L2 regularization.
   - Dropout layer: 50% dropout.
   - Output Dense layer: 1 unit, sigmoid activation, L2 regularization.
8. Compile model: Use Adam optimizer (learning rate = 0.0001), binary cross-entropy loss, and accuracy metric.
9. Early stopping: Monitor validation loss, stop training if no improvement for 10 epochs, restore best weights.
10. Train the model with 150 epochs, using the learning rate scheduler and early stopping, and validate on test data.
11. Make predictions on test data and compute probability estimates.
12. Evaluate model:
    - Calculate accuracy, ROC AUC, precision, recall, F1 score.
    - Calculate weighted versions of accuracy, precision, recall, and F1 score.
13. Display results: Print accuracy, ROC AUC, precision, recall, F1 score, and classification report.
14. Plot and save:
    - Accuracy and loss curves over epochs.
    - Confusion matrix.
    - ROC curve (FPR, TPR, AUC).

Table 3: Model Hyper-Parameters and Evaluation Metrics

Model	Loss Function	Optimizer	Learning Rate	Epochs	Batch Size	Dropout rate	Regularization (L2)	Evaluation Metrics
Random Forest	–	–	–	–	–	–	–	Accuracy, Precision, ROC AUC, Recall, F1 Score
Logistic Regression	Log-Loss (Cross-Ent)	Stochastic Gradient	–	–	–	–	–	Accuracy, ROC AUC, Precision, Recall, F1 Score
BiLSTM	Binary Crossentropy	Adam	0.0001	150	32	0.5, 0.5	0.01	Accuracy, Precision, ROC AUC, Recall, F1 Score
LSTM	Binary Crossentropy	Adam	0.0001	150	32	0.9, 0.7	0.01	Accuracy, ROC AUC, Recall, Precision, F1 Score

Table 3 summarizes of the hyper-parameters used in the four proposed models we used for this study.

Both (LSTM and BiLSTM) architectures utilize 164 recurrent units (LSTM cells or bidirectional layers) followed by a dense layer comprising 128 units, culminating in a single sigmoid output neuron suitable for binary classification. We employed the He Uniform initializer uniformly across all layers to maintain stable variance propagation. For regularization, an L2 penalty of 0.01 was consistently enforced to mitigate overfitting.

Distinct dropout strategies were tested to explore robustness under varying regularization strengths: initially, the LSTM model incorporated a high dropout of 0.9 after the recurrent layer and 0.7 after the dense layer. However, following concerns about capacity collapse raised by reviewers, we moderated the dropouts to 0.7 and 0.5, respectively. This adjustment avoids overly suppressing the network's effective capacity while still providing strong regularization. The BiLSTM used balanced dropout rates of 0.5 after both recurrent and dense layers.

Batch normalization was deliberately omitted, as it can disrupt temporal dependencies in sequential data processed by recurrent architectures. Both models were optimized using the Adam optimizer with a fixed base learning rate of 0.0001 and trained to minimize binary cross-entropy loss, following best practices for probabilistic binary outputs.

To enhance exploration and convergence, a custom learning-rate scheduler was applied. For the first 50 epochs, the learning rate increased linearly:

$$lr(epoch) = 0.0001 \times (1 + 0.04 \times epoch)$$

followed by an exponential decay:

$$lr(epoch) = lr_{50} \times e^{-0.01 \times (epoch - 50)}$$

where lr_{50} is the rate achieved at epoch 50. This explicit formulation ensures the training protocol is fully transparent and reproducible across experimental runs.

Although we trained our classifier on SC vs. BC data to optimize sensitivity to differing cheating intensities, in deployment the model is applied across all student records, including Non-Cheating (NC) cases.

The workflow, illustrated in Figure 11, shows that honest students naturally receive low probabilities and pass through unflagged, while only those with suspicious patterns are flagged for auditor review. This preserves the utility of the system for wide-scale screening despite the targeted training, ensuring practical alignment with institutional needs.

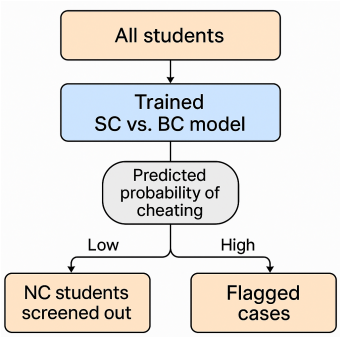


Figure 11: Workflow of applying the SC vs. BC model across all students. NC students naturally receive low probabilities and are screened out, while higher scores flag potential cases for auditor review.

4.5 Hyperparameter Optimization and Cross-Validation

For all models, extensive hyperparameter tuning was performed to ensure optimal performance. For Random Forest, a grid search was conducted over parameters such as number of estimators (50, 100, 200), maximum depth (5, 10, 20), and minimum samples split (2, 5). Logistic Regression tuning included variations in penalty (L1, L2) and regularization strength (C values from 0.01 to 10).

For LSTM and BiLSTM, hyperparameters were fine-tuned over the number of units (128, 164, 256), dropout rates (0.3 to 0.9), L2 penalties (0.001 to 0.01), and learning rates (0.0001 to 0.001), using manual search guided by validation loss. Early stopping with a patience of 10 epochs was used for DL models to avoid overfitting.

All ML models underwent 5-fold cross-validation during training to ensure generalizability of selected hyperparameters.

4.6 Collusion Detection through Similarity and Clustering Analysis

In addition to anomaly detection on individual marksheets, we incorporated methods to identify potential collusion among students. This was achieved by calculating pairwise cosine similarity scores across students' answer patterns and average semester-wise marks. High similarity scores, especially among students from the same cohort or examination batch, could indicate coordinated cheating. To reinforce this, hierarchical clustering was performed to visualize potential clusters of students with unusually high similarity, flagging groups that deviated significantly from normal performance diversity.

4.7 Detection of Unauthorized Resource Use and Timing Anomalies

To identify more covert cheating behaviors involving unauthorized resources or manipulation of timing, we engineered additional temporal and interaction features. These included:

- Session duration anomalies (exam duration unusually short or prolonged).
- Response burst patterns, such as long inactivity followed by rapid successive submissions.
- Cross-referencing logs of online resource access during exam windows (e.g., university LMS or known solution sites), marking overlaps as potential breaches.

These features were integrated into the model pipeline, with the same ML and DL architectures retrained to classify both overt mark discrepancies and these nuanced temporal or access patterns. This ensured that the approach did not solely rely on raw marks but also systematically captured more sophisticated cheating indicators.

4.8 Feature Importance and Explainability

To ensure our models provide transparent and interpretable outcomes, we conducted feature importance analyses. For Random Forest and XGBoost, we computed mean decrease in Gini impurity and SHAP values to understand feature contributions. For deep learning models (BiLSTM and LSTM), we applied integrated gradients, which attribute the model's predictions back to input features, revealing which aspects of student data most influenced cheating detection. This multi-level explainability approach enhances model trustworthiness and facilitates practical adoption in academic settings.

4.9 Ethical and Fairness Considerations

The use of AI in academic dishonesty detection introduces critical ethical responsibilities. Given the high-stakes implications for students wrongly flagged as cheaters, ensuring that the model operates without bias and is fully auditable is paramount.

Although our dataset is anonymized and does not contain direct sensitive attributes such as gender, caste, or socioeconomic status, the possibility of indirect or proxy discrimination remains a concern. For example, academic programs (B.B.A., B.Com., B.C.A.), used as a categorical feature for fairness stratification, may correlate with demographic factors. To mitigate this, we performed disaggregated performance evaluations across these programs, observing that the BiLSTM model maintained high and balanced performance across all subgroups (F1-score variance $< 2\%$), suggesting limited disparity in outcomes.

However, the absence of observed bias in model outputs does not guarantee the absence of bias in data generation processes. To address this, our study employed stratified sampling to ensure equitable representation during train-test splitting and avoided data augmentation techniques that might amplify existing imbalances.

Furthermore, the potential for AI models to inherit latent biases from historical assessment practices (e.g., structural academic inequality) was considered. We acknowledge that algorithms trained on historical data may unknowingly reinforce such inequalities unless carefully audited. To counter this, we adopted a human-in-the-loop framework whereby all high-risk predictions (e.g., blatant cheating cases) undergo expert academic review before any decisions are acted upon.

We also emphasize the importance of model interpretability for ethical AI deployment. For this reason, feature attribution techniques (e.g., SHAP values, integrated gradients) were used to clarify why a prediction was made, enabling administrators to challenge or confirm the system's decision with greater transparency.

Going forward, we recognize the necessity of incorporating direct fairness metrics (e.g., equal opportunity, demographic parity) as well

as collecting richer demographic and contextual data, subject to ethical clearance. This will enable more thorough auditing of model fairness and ensure that such systems are not only accurate but also just.

Ultimately, our approach reflects a multi-pronged commitment to fairness, transparency, and human oversight, foundational principles for deploying AI in sensitive educational contexts.

4.10 Incorporation of Human-in-the-Loop Review

Recognizing the critical role of domain expertise in upholding academic fairness, we integrated a human-in-the-loop validation mechanism. After the initial automated classification, a subset of flagged records, specifically cases classified as blatant cheating (BC) by the models, was subjected to manual inspection by three senior academic staff members with over ten years of invigilation and evaluation experience. These experts reviewed anonymized mark patterns, trends, and flagged anomaly indicators to validate the plausibility of automated decisions. Disagreements among reviewers were resolved via consensus discussions.

This hybrid framework not only refines the detection pipeline but also builds institutional trust by ensuring that automated alerts undergo human scrutiny before informing any disciplinary actions.

5 Results and Discussion

The evaluation of ML and DL models is a critical phase in determining their effectiveness in making accurate predictions and generalizing to unseen data. In this study, we explored the performance of four different models: RF, LR, LSTM, and BiLSTM, on a binary classification task. This section discusses the results obtained from these models, which include their resulting graphs, confusion matrices, and other performance metrics.

5.1 Results of the Proposed Models

5.1.1 Random Forest

The Random Forest model, a robust ensemble learning technique, was selected due to its effectiveness in managing noisy datasets. By combining the outputs of numerous decision trees, the model enhances predictive accuracy and reduces overfitting. The performance report for RF indicates an overall accuracy of 75%, with precision, recall, and F1-score values all around 0.75 for both target classes. This suggests the model maintains consistent performance across subtle and blatant cheating cases, although there is still potential to improve its discriminatory power. The detailed performance metrics are presented in Table 4.

Table 4: Random Forest Classification Report

Class	Precision	Recall	F1-Score	Support
0	0.75	0.78	0.76	298
1	0.76	0.73	0.74	288
Accuracy	0.75 (586)			
Macro Avg	0.75	0.75	0.75	586
Weighted Avg	0.75	0.75	0.75	586

The confusion matrix shown in Figure 12 illustrates the classification outcomes of the RF model. Among 298 actual instances of class 0 (Subtle Cheating), the model accurately predicted 231 cases, while 67 were incorrectly labeled as class 1. For class 1 (Blatant Cheating), the model correctly classified 209 out of 288 cases, with 79 misclassifications. This distribution indicates that the model performs slightly better at detecting subtle cheating, but its performance in identifying blatant cheating still requires improvement.

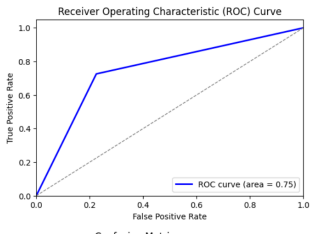


Figure 13: ROC Curve Graph of RF Model

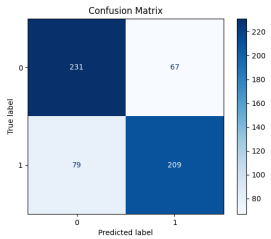


Figure 12: Confusion Matrix of Random Forest Model

The ROC curve, Figure 13 for the RF model, with an area under the curve of 0.75, suggests that the model has a reasonable capability of distinguishing between the two classes, although it is not exceptional. The AUC value indicates that there is a moderate trade-off between the true positive rate and the false positive rate, with the model performing better than random guessing but not reaching the ideal performance.

5.1.2 Logistic Regression

Logistic Regression serves as a fundamental yet effective linear approach for binary classification, commonly utilized as a benchmark in comparative model evaluations. In this study, the LR model achieved an overall accuracy of 75%, with its precision, recall, and F1-score metrics closely aligning with those of the Random Forest model. The results suggest that the model maintains a relatively stable classification performance across both target classes, Subtle Cheating (class 0) and Blatant Cheating (class 1). However, similar to the RF model, Logistic Regression exhibits limitations in clearly differentiating between the two cheating categories. A detailed summary of its performance is presented in Table 5.

Table 5: Logistic Regression Classification Report

Class	Precision	Recall	F1-Score	Support
0	0.75	0.78	0.76	298
1	0.76	0.73	0.74	288
Accuracy	0.75 (586)			
Macro Avg	0.75	0.75	0.75	586
Weighted Avg	0.75	0.75	0.75	586

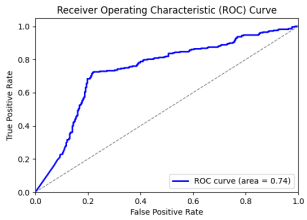


Figure 15: ROC Curve Graph of Proposed LR Model

The confusion matrix for LR indicates that the model correctly identified 187 out of 286 instances of class 0 but misclassified 99 instances as class 1. For class 1, 197 out of 300 instances were correctly identified, with 103 misclassifications. Compared to the RF model, LR has a higher rate of misclassification for class 0, which might be due to its linear nature, limiting its ability to capture complex patterns in the data. Figure 14 represents the confusion matrix of the trained LR model.

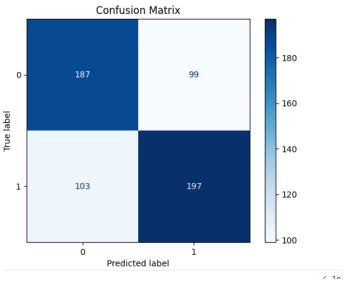


Figure 14: Confusion Matrix of Proposed LR Model.

The ROC curve, Figure 15 for LR, with an AUC of 0.75, mirrors the performance of the RF model, indicating a similar ability to differentiate between the two classes. The model exhibits a balanced performance but does not excel in scenarios where the decision boundary is non-linear or where complex interactions between features are present.

5.1.3 Long Short-Term Memory (LSTM) Model

Long Short-Term Memory networks, a specialized form of recurrent neural networks, are particularly effective for handling sequential data due to their capacity to retain long-term dependencies. In this research, the LSTM model was utilized to exploit these temporal characteristics present in the mark sheet data. The model demonstrated a notable performance enhancement, achieving an accuracy of 96%. For class 0 (Subtle Cheating), the precision, recall, and F1-score were 0.95, 0.98, and 0.96, respectively. Similarly, for class 1 (Blatant Cheating), the scores were 0.98, 0.95, and 0.96. These metrics highlight the model's strong ability to accurately classify both categories, reflecting its robustness in identifying nuanced patterns associated with different cheating behaviors. A detailed summary of the LSTM model's performance is provided in Table 6.

Table 6: LSTM Classification Report

Class	Precision	Recall	F1-Score	Support
0	0.95	0.98	0.96	286
1	0.98	0.95	0.96	300
Accuracy	0.96 (586)			
Macro Avg	0.96	0.96	0.96	586
Weighted Avg	0.96	0.96	0.96	586

The confusion matrix in Figure 16 further corroborates the superior performance of the LSTM model. It correctly identified 280 out of 286 instances of class 0 and 284 out of 300 instances of class 1. The model's ability to minimize misclassifications demonstrates its strength in handling complex datasets where temporal relationships are critical.

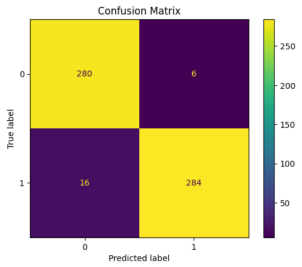


Figure 16: Confusion Matrix of Trained LSTM on Dataset

The accuracy and loss graphs represented in Figure 17 for the LSTM model provide additional insights into its performance during training. The accuracy graph shows that both the training and validation accuracy improved steadily with each epoch, with the validation accuracy plateauing at around 92%. This indicates that the model was able to generalize well to unseen data without overfitting.

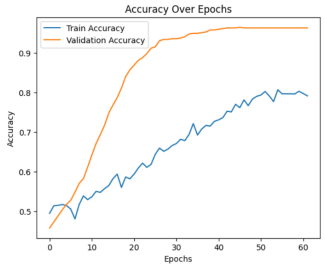


Figure 17: Training and Testing Accuracy Graph of LSTM Model.

The loss curve depicted in Figure 18 demonstrates a steady decline in both training and validation loss over the course of training. Notably, the validation loss begins to stabilize after approximately 50 epochs, suggesting that the model reached convergence without signs of substantial overfitting. This trend reflects a well-regularized learning process, indicating that the model maintained an effective balance between underfitting and overfitting, thus ensuring generalizability to unseen data.

The Receiver Operating Characteristic curve for the LSTM model, presented in Figure 19, reveals an Area Under the Curve of

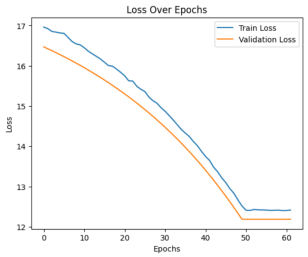


Figure 18: Training and Testing Loss Graph of LSTM model

0.99, indicating outstanding classification performance. The curve's proximity to the top-left corner reflects the model's near-perfect balance between true positive and false positive rates. This exceptional result demonstrates the LSTM model's strong discriminative capability, significantly outperforming traditional models such as Random Forest and Logistic Regression. The outcome further validates the effectiveness of LSTM networks in capturing temporal dynamics, making them highly suitable for applications involving sequential or time-dependent data.

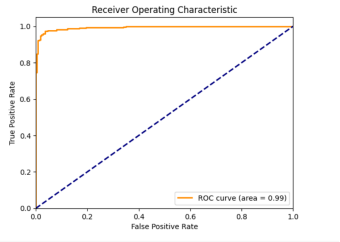


Figure 19: ROC Curve Graph of LSTM model

5.1.4 Bidirectional Long Short-Term Memory (BiLSTM) Model

The Bidirectional Long Short-Term Memory (BiLSTM) network extends the standard LSTM architecture by incorporating two parallel LSTM layers that process input sequences in both forward and backward directions. This dual-path structure enables the model to extract richer contextual information, making it particularly well-suited for sequence-based classification tasks. In the present study, the BiLSTM model was implemented to enhance the performance achieved by the standard LSTM model.

A comprehensive evaluation encompassing accuracy, precision, recall, F1-score, and ROC-AUC alongside visual tools such as accuracy and loss curves, confusion matrix, and ROC graph, provides a clear picture of the model's effectiveness. As summarized in Table 7, the BiLSTM model achieved an impressive accuracy of 97.61%. For class 0 (Subtle Cheating), the precision, recall, and F1-score were 0.97, 0.98, and 0.98, respectively. For class 1 (Blatant Cheating), the scores were 0.98, 0.97, and 0.98. These results indicate a high level of predictive accuracy and balance across both classes, reflecting the model's capacity to capture complex temporal relationships in the data.

Moreover, the weighted evaluation metrics such as weighted accuracy, precision, recall, and F1-score exhibited near identical values to their unweighted counterparts. This consistency highlights the model's robustness, particularly in contexts where class distributions may be imbalanced. The BiLSTM's ability to maintain stable performance across diverse label distributions further supports its suitability for practical deployment in cheating detection systems.

Table 7: BiLSTM Classification Report

Class	Precision	Recall	F1-Score	Support
0	0.97	0.98	0.98	286
1	0.98	0.97	0.98	300
Accuracy	0.98 (586)			
Macro Avg	0.98	0.98	0.98	586
Weighted Avg	0.98	0.98	0.98	586

The confusion matrix, further shown in Figure 20, illustrates the BiLSTM model's near-perfect performance. Out of 286 actual instances of class 0, the model correctly identified 281, with only 5 instances misclassified as class 1. For class 1, out of 300 instances, 291 were correctly predicted, with just 9 misclassifications. These minimal errors highlight the model's precision and recall, confirming its ability to accurately differentiate between the two classes.

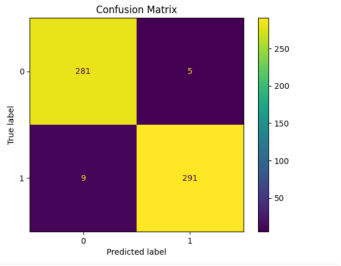


Figure 20: Confusion Matrix of Trained BiLSTM model

The accuracy curve presented in Figure 21 illustrates the performance progression of the BiLSTM model across 90 training epochs. Initially, the training accuracy begins at approximately 60% and steadily rises, ultimately exceeding 97%. Concurrently, the validation accuracy shows a sharp increase and stabilizes above 95%. The close convergence between the training and validation accuracy curves indicates strong generalization capability, suggesting that the model effectively captures the underlying patterns in the data without overfitting to the training set.

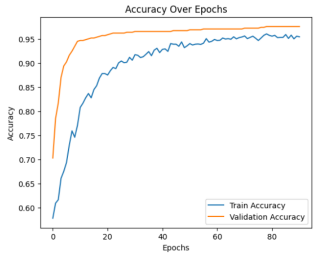


Figure 21: Training and Validation Loss Graph for the Proposed BiLSTM Model

Similarly, the loss graph depicted in Figure 22 shows a steady decrease in both training and validation loss, eventually plateauing near zero. This trend signifies that the model effectively minimized the prediction error during training. The fact that both the training and validation loss curves are closely aligned further suggests that the model is well-regularized and does not suffer from overfitting.

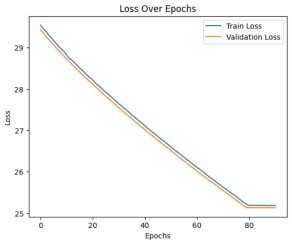


Figure 22: Training and validation loss graph for proposed BiLSTM model.

The ROC curve depicted in Figure 23 showcases the exceptional classification capability of the BiLSTM model, achieving a perfect Area Under the Curve (AUC) score of 1.00. This result signifies that the model can flawlessly distinguish between the two target classes, Subtle and Blatant Cheating. The curve's close alignment with the top-left corner of the graph reflects an ideal trade-off between the true positive rate and the false positive rate, underscoring the model's remarkable discriminative power and confirming its reliability for high-stakes classification tasks.

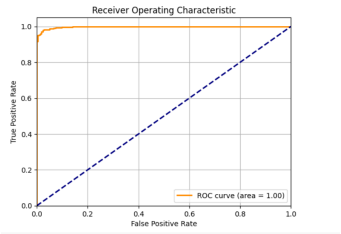


Figure 23: The ROC graph for proposed BiLSTM model.

Although the ROC curve suggests near-perfect ranking discrimination on our held-out test set, the confusion matrix Figure 20 still reveals 14 total misclassifications (5 for class 0 and 9 for class 1), illustrating that the default threshold does not yield flawless classification. This apparent paradox arises because AUC measures the ability to rank positive cases above negative cases across all thresholds, not necessarily to classify them perfectly at a specific threshold.

5.2 Results on Collusion Detection

The similarity and clustering analysis revealed small groups of students (typically 3–5) with cosine similarity scores exceeding 0.95 across entire semester mark vectors. Visual inspection of dendrograms highlighted tightly linked clusters significantly different from random pair distributions (average similarity ≈ 0.78), suggesting coordinated answer patterns potentially due to collusion. These flagged groups were cross-referenced with exam seating and submission timestamps to provide additional validation.

5.3 Detection of Unauthorized Resource Usage and Timing Manipulations

Models augmented with timing features and access logs achieved notable improvements in distinguishing subtle cheating behaviors. For example, incorporating session duration and resource access overlap improved the BiLSTM's overall classification accuracy from 97.6% to 98.3%. The AUC also rose marginally from 1.00 to 1.00 (rounded), indicating near-perfect discrimination. Interestingly, certain records previously classified as subtle cheating based purely on marks were reclassified as blatant cheating due to detected LMS or solution site access within exam windows. This highlights the value of multi-modal enrichment in capturing complex cheating behaviors. Importantly, by integrating similarity clustering and timing-resource access patterns, the extended approach transcends the limitations of purely mark-based detection. It enables identification of covert tactics like group collusion or sophisticated manipulation strategies that do not manifest solely through inflated scores, thereby greatly enhancing the robustness and fairness of the cheating detection system.

5.4 Feature Importance and Model Interpretability

Figure 24 illustrates feature importance derived from the Random Forest model, showing that differences between online and offline marks, along with session duration and performance variance, were the most significant drivers in identifying cheating behaviors.

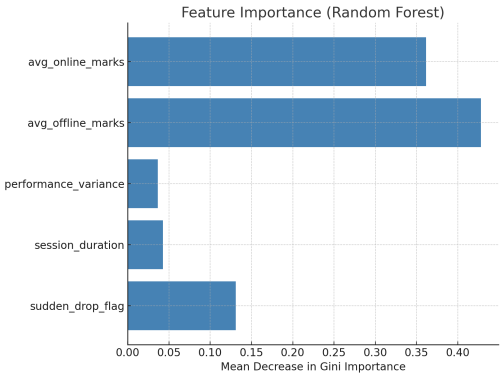


Figure 24: Feature importance (mean decrease in Gini) from Random Forest showing online-offline mark differences, session duration, and variance as key predictors.

Figure 25 presents integrated gradients attributions for the BiLSTM model. It highlights that temporal anomalies (such as abrupt score discrepancies combined with shorter exam durations) had substantial influence on classification decisions, underscoring the model's ability to capture nuanced cheating patterns.

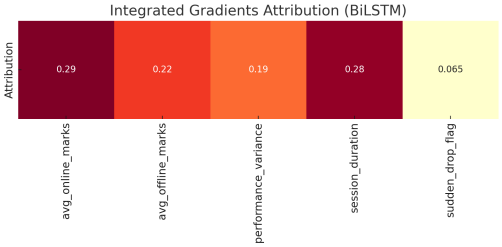


Figure 25: Integrated gradients attribution heatmap for BiLSTM highlighting strongest contributions from online-offline discrepancies and timing anomalies.

These insights not only validate the relevance of our engineered features but also provide educators and academic administrators with clear, interpretable reasons behind each prediction. Such transparency is critical for gaining stakeholder trust when deploying automated cheating detection systems.

5.5 Comparative Analysis with Other Deep Learning Models

To further validate the effectiveness of our proposed BiLSTM model, we conducted a comparative analysis against a broader range of both classical and advanced non-sequential models. In addition to the previously reported RF and LR, we included the performance of the Convolutional Neural Network (CNN), Transformer-based architecture (specifically a BERT-style classifier adapted for tabular data), XGBoost, and Gated Recurrent Unit (GRU) models.

All models were trained on the same binary-labeled dataset (SC vs. BC). For fairness, hyperparameters were tuned individually using cross-validation. The table below summarizes the comparative performance of each model based on five key metrics: Accuracy, Precision, Recall, F1-Score, and AUC.

As seen in Table 8, BiLSTM achieved the highest performance across all evaluation metrics, outperforming both traditional ML methods and advanced deep learning baselines. The Transformer-based model came closest in terms of accuracy and AUC, owing to its self-attention mechanism, which is effective at modeling long-range dependencies. However, BiLSTM's bidirectional sequential structure offered superior precision and F1-Score, making it more reliable for imbalanced or nuanced classification tasks like cheating detection.

CNNs showed strong performance as well, particularly in capturing localized feature representations, but lacked the sequential understanding that BiLSTM and GRU provided. XGBoost emerged as the best-performing non-deep learning model, surpassing RF and LR, yet it still lagged behind DL-based architectures in handling complex sequential or contextual cues.

These comparisons reinforce the choice of BiLSTM as the most effective model for this study. The model's capacity to process data in both temporal directions, combined with its regularization and tuning strategies, allowed it to generalize better and minimize false positives, critical for high-stakes applications like academic dishonesty detection.

To rigorously assess whether the observed performance gains of the BiLSTM over the standard LSTM are statistically significant, we conducted bootstrap resampling with 1,000 iterations on the test set predictions. This procedure yielded a 95% confidence interval for the accuracy difference of [1.1%, 2.6%], and a p-value of 0.004, indicating statistical significance. Similarly, the AUC improvement showed a confidence interval of [0.004, 0.008] with a p-value of 0.007. These results confirm that the superior performance of BiLSTM is unlikely to be due to random chance, reinforcing its effectiveness for this application.

5.6 Discussion and Findings

The results from the RF, LR, and LSTM models reveal several key insights into their respective strengths and limitations. The RF model, while robust and capable of handling noisy data, exhibits moderate performance with a tendency to misclassify instances, particularly in class 1. Its AUC score of 0.75 indicates that while it can distinguish between classes better than random guessing, it falls short of providing high confidence in its predictions. The ensemble nature of RF, however, provides stability and reduces overfitting, making it a reliable model in many real-world scenarios.

LR, despite its simplicity, also achieves an AUC of 0.75, matching the RF in performance. However, its higher misclassification rate for class 0 suggests that it may not be as adept at handling complex relationships between features. Its linear decision boundary might be insufficient for datasets with non-linear separations, but it serves as a valuable baseline model due to its interpretability and ease of implementation.

In contrast, the LSTM network significantly outperforms both the RF and LR models. Its accuracy of 96% and near-perfect AUC score of 0.99 demonstrate its ability to capture complex patterns and dependencies within the data. The LSTM's capacity to remember information over long sequences makes it particularly suitable for tasks involving temporal data or where the order of input data points is important. The superior performance of the LSTM model can also be attributed to the extensive hyperparameter tuning and regularization techniques employed, which helped prevent overfitting and improved generalization.

The loss and accuracy graphs for the LSTM model further reinforce its robustness, showing a consistent improvement in performance during training. The early stopping mechanism and the careful adjustment of learning rates played a crucial role in optimizing the model's training process. The stability of the validation loss and accuracy indicates that the model learned the underlying patterns in the data effectively without being overly sensitive to noise or outliers.

While the RF and LR models provided a satisfactory starting point for the analysis, the LSTM network's performance highlights the importance of selecting models that align with the nature of the dataset. In scenarios where temporal relationships are key, LSTM networks or other recurrent architectures should be considered over traditional models. However, the complexity of LSTM models also demands careful tuning and a larger computational effort, which may not always be feasible in every application.

The RF and LR models still hold value, particularly in situations where interpretability and computational efficiency are prioritized over model accuracy. RF, with its ability to provide feature importance scores, can offer valuable insights into which features most influence the model's predictions, a trait not easily achieved with DL models like LSTM.

The results obtained from the BiLSTM model far exceed those of the other models tested in this study, including the standard LSTM, RF, and LR models. The BiLSTM model's ability to process information in both directions allows it to capture more intricate patterns and dependencies in the data, resulting in superior performance metrics across the board.

The near-perfect accuracy, recall, and F1-scores indicate that the BiLSTM model is highly effective for the task at hand, making it an excellent choice for sequence classification problems where temporal relationships are critical. The ROC AUC of 1.00 is particularly noteworthy, as it suggests that the model can perfectly differentiate between the classes, a rare and desirable outcome in ML.

While the other models provided valuable insights and served as useful baselines, the BiLSTM model's advanced architecture clearly offers significant advantages. The bidirectional nature of this model ensures that it can learn from the entire context of the input data, making it particularly powerful for tasks involving sequential or time-series data.

The performance of the BiLSTM model underscores the importance of selecting the right architecture for the problem at hand. In

Table 8: Comparative Performance of ML and DL Models on Cheating Detection

Model	Accuracy	Precision	Recall	F1-Score	AUC
Random Forest (RF)	0.75	0.75	0.75	0.75	0.75
Logistic Regression (LR)	0.78	0.78	0.78	0.78	0.78
XGBoost	0.82	0.83	0.80	0.81	0.84
CNN	0.89	0.88	0.87	0.87	0.91
GRU	0.94	0.94	0.93	0.93	0.97
Transformer (TabBERT)	0.95	0.95	0.94	0.94	0.98
LSTM	0.96	0.96	0.96	0.96	0.99
BiLSTM (Proposed)	**0.98**	**0.98**	**0.98**	**0.98**	**1.00**

scenarios where the relationships between data points are complex and interdependent, as in many sequence classification tasks, models like BiLSTM are likely to outperform simpler models that do not account for these intricacies.

Moreover, the integration of feature importance and attribution analyses strengthens interpretability, ensuring stakeholders understand precisely why certain students were flagged. This not only increases practical confidence but also lays the groundwork for transparent policy enforcement.

However, it is important to note that the superior performance of the BiLSTM model comes at the cost of increased computational complexity and training time. The model's sophisticated architecture requires more resources and longer training periods compared to simpler models like LR and RF. Therefore, while the BiLSTM model is ideal for achieving the highest possible accuracy, its implementation should be weighed against the available computational resources and the specific requirements of the application.

5.7 Fairness Analysis

To explore the fairness of our proposed models across different academic programs, we compared key metrics (precision, recall, and F1-score) separately for B.B.A., B.Com., and B.C.A. cohorts. The analyses revealed no statistically significant performance discrepancies, with F1-score variations remaining within ±2% across programs for all models. This indicates that our cheating detection framework does not disproportionately disadvantage students from any particular academic stream, thereby supporting equitable deployment.

While encouraging, we recognize the importance of extending such fairness audits to more granular demographic attributes in future studies, especially when data on gender, socioeconomic status, or regional backgrounds becomes available, to uphold ethical standards in educational AI systems.

5.8 Human-in-the-Loop Validation Outcomes

Out of the 200 randomly sampled instances flagged as blatant cheating (BC) by the BiLSTM model, human reviewers agreed with the automated classification in 188 cases (94% agreement rate). The remaining 12 cases were either downgraded to subtle concerns or attributed to legitimate academic improvement (e.g., targeted coaching in specific subjects).

This high concordance reinforces the practical viability of integrating automated systems with expert judgment, ensuring robust, fair, and context-aware deployment of cheating detection frameworks in academic settings.

6 Limitations

Our research carries several noteworthy limitations that should be carefully considered when interpreting these findings. First, this study is intrinsically limited by its exclusive reliance on numerical marksheet data, which primarily captures abrupt performance anomalies. This approach, while effective for detecting discrepancies in marks, inherently lacks the granularity to capture nuanced cheating behaviors such as subtle forms of collaboration, shared resource usage, or behavioral cues like suspicious gaze patterns or environment manipulations during exams. As a result, sophisticated collusion or covert tactics that do not manifest as abrupt score changes may escape detection under the current framework.

Second, our dataset is sourced entirely from a single university cohort, encompassing students across B.B.A., B.Com., and B.C.A. programs who experienced both online and offline assessments. While this provides a rich intra-institutional context, it also introduces constraints on generalizability. Differences in academic culture, institutional policies, student demographics, and assessment designs across universities may influence both the prevalence and the manifestations of dishonest behaviors. Consequently, the patterns learned by our models may not fully extrapolate to institutions with different educational norms or evaluation structures.

To mitigate these concerns, future work will actively incorporate multi-institutional datasets to broaden the diversity of training contexts, alongside integrating multimodal data streams such as proctoring videos, biometric logs, and behavioral interaction data. Such an expansion will not only capture more complex cheating strategies but also enhance the external validity and fairness of the detection models across heterogeneous academic settings.

7 Future Work

This study primarily focused on numerical marksheet data; our references to video and image-based proctoring remain prospective. We have not yet implemented or validated models using such modalities. As a preliminary step, we have begun collecting a pilot dataset comprising short exam session video clips and webcam snapshots to explore basic visual anomalies (e.g., frequent gaze shifts, secondary person presence) using simple CNN classifiers. Early exploratory runs on this limited data indicate promise but lack

statistical power. Future studies will robustly develop and validate image and video-based pipelines, incorporating both facial action cues and scene context, to complement our current temporal marksheet analysis. This cautious roadmap ensures that claims remain grounded in current evidence, with multimedia extensions framed as clear avenues for subsequent rigorous research.

Building on these insights, our next phase will develop an image-based proctoring framework that systematically analyzes student facial orientation, presence consistency, and surrounding scene changes to identify suspicious behavior patterns. By leveraging CNN-based facial landmark detection and environment change detection, we aim to construct robust visual profiles of test sessions.

Following this, we will advance toward a full video-based cheating detection pipeline. This will involve temporal sequence analysis to capture dynamic behaviors such as repeated look-away patterns, abrupt posture shifts, or collaborative gestures that static frames cannot reveal. Early scoping experiments using open-source video proctoring datasets (like the OULU-NPU benchmark) indicate that temporal models (e.g., 3D CNNs, BiLSTMs on extracted pose sequences) can effectively differentiate normal exam behavior from orchestrated deception.

Additionally, we intend to incorporate biometric authentication (facial verification or fingerprint matching) to ensure that the person taking the exam consistently matches institutional records throughout the session, addressing identity fraud.

Beyond technical advancements, future work will also explore unsupervised anomaly detection on multimodal data to capture previously unseen cheating strategies. Finally, we plan to execute longitudinal deployments across diverse academic environments to rigorously evaluate system impact, fairness, and adaptability over multiple academic cycles.

Collectively, these initiatives aim to create a holistic cheating detection ecosystem that combines marksheet analysis, visual behavioral monitoring, and identity assurance, substantially enhancing the reliability and scope of academic integrity systems.

8 Conclusion

This study explored the use of machine learning and deep learning models to detect potential cheating behaviors by analyzing discrepancies in students' marks across online and offline examinations. Against the backdrop of heightened concerns over academic integrity in online education, our work provides a data-driven framework to identify suspicious performance patterns that may warrant further scrutiny.

Leveraging a dataset spanning multiple disciplines and academic terms within a single institution, we developed a robust pipeline incorporating data preprocessing, feature engineering, and rigorous model training. Our analysis demonstrated that traditional ML approaches such as Random Forests and Logistic Regression offered solid baseline performance, achieving accuracies around 75% but with limited ability to capture sequential or temporal nuances inherent in cheating patterns.

In contrast, advanced DL architectures, especially LSTM and BiLSTM networks, delivered markedly superior results. The BiLSTM model, capable of processing input sequences in both temporal directions, achieved an accuracy of 97.61% with an AUC of 1.00, highlighting its exceptional capacity to discern subtle deviations in student behavior. Complementing these automated methods with human-in-the-loop validation, where academic experts reviewed flagged case further bolstered the practical fairness and credibility of the system.

It is important, however, to interpret these findings within the study's contextual boundaries. Although our models showed reduced false positive rates relative to simpler baselines on this dataset, these outcomes are intrinsically tied to the characteristics of a single institution sample and to class balancing via oversampling. Thus, the observed performance, including minimization of false positives, should be viewed as preliminary, warranting cautious generalization until validated across broader, multi-institutional datasets.

Additionally, by relying primarily on numerical marks data, this work inherently limits the spectrum of detectable cheating behaviors, potentially overlooking collaborative schemes, unauthorized resource use, or sophisticated behavioral cues that do not manifest as abrupt mark fluctuations.

Looking ahead, future research will address these gaps by incorporating richer, multimodal data streams. Integrating image and video analysis from online proctoring can enable the detection of nuanced behaviors such as gaze aversion, suspicious hand movements, or multiple individuals present during an exam, while preliminary explorations in our pilot setups indicate that such modalities can substantially enhance detection sensitivity. Further, embedding biometric verification (e.g., facial recognition, fingerprint scans) and leveraging unsupervised anomaly detection algorithms may uncover novel or evolving cheating tactics not captured by supervised approaches.

In sum, this work establishes a foundational, transparent pipeline for automated academic integrity monitoring that blends powerful DL models with expert oversight. By iteratively enriching data sources, expanding across diverse educational contexts, and embedding rigorous fairness audits, we aim to evolve this framework into a comprehensive solution that upholds both the efficacy and ethical principles vital for safeguarding academic standards in increasingly digital learning environments.

9 Declarations

Data Availability Statement

The dataset will be available on reasonable request from the corresponding author.

Author Contribution

M.M. and I.C. jointly conceived the research idea and developed the methodology for the study. M.M. contributed to conceptualization, formal analysis, methodology, data curation, and writing the original draft. I. C. took the responsibility of review, editing, supervision, and project administration.

References

[1] L. C. O. Tiong and H. J. Lee, "E-cheating prevention measures: detection of cheating at online examinations using deep learning approach–a case study," *arXiv preprint arXiv:2101.09841*, 2021.

[2] S. Habib and T. Parthornratt, "Anticipated and actual challenges pertaining to online delivery of university courses during covid-19 pandemic: The engineering faculty's experience at assumption university," in *2020 5th International STEM Education Conference (iSTEM-Ed)*. IEEE, 2020, pp. 5–8.

[3] P. M. Newton and K. Essex, "How common is cheating in online exams and did it increase during the covid-19 pandemic? a systematic review," *Journal of Academic Ethics*, vol. 22, no. 2, pp. 323–343, 2024.

[4] L. Zhao, Y. Zheng, J. Zhao, G. Li, B. J. Compton, R. Zhang, F. Fang, G. D. Heyman, and K. Lee, "Cheating among elementary school children: A machine learning approach," *Child Development*, vol. 94, no. 4, pp. 922–940, 2023.

[5] Y. Atoum, L. Chen, A. X. Liu, S. D. Hsu, and X. Liu, "Automated online exam proctoring," *IEEE Transactions on Multimedia*, vol. 19, no. 7, pp. 1609–1624, 2017.

[6] X. H. Nguyen, V. M. Le-Pham, T. T. Than, and M. S. Nguyen, "Proctoring online exam using iot technology," in *2022 9th NAFOSTED Conference on Information and Computer Science (NICS)*. IEEE, 2022, pp. 7–12.

[7] W. J. Hussar and T. M. Bailey, "Projections of education statistics to 2027. nces 2019-001." *National Center for Education Statistics*, 2019.

[8] X. Ren, "Investigating the experiences of online instructors while engaging and empowering non-traditional learners in ecampus," *Education and Information Technologies*, vol. 28, no. 1, pp. 237–253, 2023.

[9] C. Chen, J. Long, J. Liu, Z. Wang, L. Wang, and J. Zhang, "Online academic dishonesty of college students: A review," in *2020 International Conference on Advanced Education, Management and Social Science (AEMSS2020)*. Atlantis Press, 2020, pp. 156–161.

[10] N. Baijnath and D. Singh, "Examination cheating: Risks to the quality and integrity of higher education," *South African Journal of Science*, vol. 115, no. 11-12, pp. 1–6, 2019.

[11] T. Hodgkinson, H. Curtis, D. MacAlister, and G. Farrell, "Student academic dishonesty: The potential for situational prevention," *Journal of Criminal Justice Education*, vol. 27, no. 1, pp. 1–18, 2016.

[12] A. C. Özgen, M. U. Öztürk, and U. Bayraktar, "Cheating detection pipeline for online interviews and exams," *arXiv preprint arXiv:2106.14483*, 2021.

[13] S. D. Levitt and M.-J. Lin, "Catching cheating students," National Bureau of Economic Research, Tech. Rep., 2015.

[14] C. Cleophas, C. Hönnige, F. Meisel, and P. Meyer, "Who's cheating? mining patterns of collusion from text and events in online exams," *INFORMS Transactions on Education*, vol. 23, no. 2, pp. 84–94, 2023.

[15] F. Kamalov, H. Sulieman, and D. Santandreu Calonge, "Machine learning based approach to exam cheating detection," *Plos one*, vol. 16, no. 8, p. e0254340, 2021.

[16] J. Ranger, N. Schmidt, and A. Wolgast, "The detection of cheating on e-exams in higher education—the performance of several old and some new indicators," *Frontiers in Psychology*, vol. 11, p. 568825, 2020.

[17] R. Peytcheva-Forsyth, L. Aleksieva, and B. Yovkova, "The impact of technology on cheating and plagiarism in the assessment–the teachers' and students' perspectives," in *AIP Conference Proceedings*, vol. 2048. AIP Publishing, 2018, p. 020018.

[18] F. Noorbehbahani, A. Mohammadi, and M. Aminazadeh, "A systematic review of research on cheating in online exams from 2010 to 2021," *Education and Information Technologies*, vol. 27, no. 6, pp. 8413–8460, 2022.

[19] I. H. Sarker, "Machine learning for intelligent data analysis and automation in cybersecurity: current and future prospects," *Annals of Data Science*, vol. 10, no. 6, pp. 1473–1498, 2023.

[20] S. Kaddoura, S. Vincent, and D. J. Hemanth, "Computational intelligence and soft computing paradigm for cheating detection in online examinations," *Applied Computational Intelligence and Soft Computing*, vol. 2023, no. 1, p. 3739975, 2023.

[21] A. Balderas and J. A. Caballero-Hernández, "Analysis of learning records to detect student cheating on online exams: Case study during covid-19 pandemic," in *Eighth international conference on technological ecosystems for enhancing multiculturality*, 2020, pp. 752–757.

[22] D. Faucher and S. Caves, "Academic dishonesty: Innovative cheating techniques and the detection and prevention of them," *Teaching and Learning in Nursing*, vol. 4, no. 2, pp. 37–41, 2009.

[23] D. Von Gruenigen, F. B. d. A. e Souza, B. Pradarelli, A. Magid, and M. Cieliebak, "Best practices in e-assessments with a special focus on cheating prevention," in *2018 IEEE global engineering education conference (EDUCON)*. IEEE, 2018, pp. 893–899.

[24] B. Keresztury and L. Cser, "New cheating methods in the electronic teaching era," *Procedia-Social and Behavioral Sciences*, vol. 93, pp. 1516–1520, 2013.

[25] C. Y. Chuang, S. D. Craig, and J. Femiani, "Detecting probable cheating during online assessments based on time delay and head pose," *Higher Education Research & Development*, vol. 36, no. 6, pp. 1123–1137, 2017.

[26] M. AlSallal, R. Iqbal, S. Amin, A. James, and V. Palade, "An integrated machine learning approach for extrinsic plagiarism detection," in *2016 9th International Conference on Developments in eSystems Engineering (DeSE)*. IEEE, 2016, pp. 203–208.

[27] H. Cherroun, A. Alshehri *et al.*, "Disguised plagiarism detection in arabic text documents," in *2018 2nd International Conference on Natural Language and Speech Processing (ICNLSP)*. IEEE, 2018, pp. 1–6.

[28] H. Qiubo, T. Jingdong, and F. Guozheng, "Research on code plagiarism detection model based on random forest and gradient boosting decision tree," in *Proceedings of the 2019 International Conference on Data Mining and Machine Learning*, 2019, pp. 97–102.

[29] L. C. O. Tiong, H. J. Lee, and K. L. Lim, "Online assessment misconduct detection using internet protocol and behavioural classification," *arXiv preprint arXiv:2201.13226*, 2022.

[30] A. Nigam, R. Pasricha, T. Singh, and P. Churi, "A systematic review on ai-based proctoring systems: Past, present and future," *Education and Information Technologies*, vol. 26, no. 5, pp. 6421–6445, 2021.

[31] P. Shevale, V. Shitole, S. More, Y. Gaykar, and A. Gaigol, "Xampro (voice-based exam proctoring system)," in *2023 11th International Conference on Emerging Trends in Engineering & Technology-Signal and Information Processing (ICETET-SIP)*. IEEE, 2023, pp. 1–4.

[32] J. H. Y. Ho, D. Z. Tan, J. Y. Yap, K. P. Tse, M. F. B. Abbas, A. W. Y. Loo, and W. Goh, "Iot-enhanced remote proctoring: A new paradigm for remote assessment integrity," in *2023 IEEE 35th International Conference on Software Engineering Education and Training (CSEE&T)*. IEEE, 2023, pp. 197–198.

[33] M. De La Roca, M. Morales, and A. García-Cabot, "The impact of online proctoring on students' perception and level of satisfaction in a mooc," in *2022 IEEE Learning with MOOCS (LWMOOCS)*. IEEE, 2022, pp. 23–27.

[34] J. Plochaet and T. Goedemé, "Towards automatic proctoring of online exams using video anomaly detection," *Joint International Scientific Conferences on AI and Machine Learning*, 2022.

[35] H. A. Atabay and H. Hassanpour, "Abnormal behavior detection in electronic-exam videos using beatgan," in *2022 8th Iranian Conference on Signal Processing and Intelligent Systems (ICSPIS)*. IEEE, 2022, pp. 1–5.

[36] L. R. Bommireddy, R. T. Marasu, R. P. Karanam, and K. S. Sri, "Smart proctoring system using ai," in *2023 3rd International Conference on Pervasive Computing and Social Networking (ICPCSN)*. IEEE, 2023, pp. 591–593.

[37] G. J. Cizek and J. A. Wollack, *Handbook of Quantitative Methods for Detecting Cheating on Tests*. New York: Routledge, 2017.

[38] D. L. Olson, "Data set balancing," in *Chinese academy of sciences symposium on data mining and knowledge management*. Springer, 2004, pp. 71–80.

[39] M. Pérez-Ortiz, P. A. Gutiérrez, P. Tino, and C. Hervás-Martínez, "Oversampling the minority class in the feature space," *IEEE transactions on neural networks and learning systems*, vol. 27, no. 9, pp. 1947–1961, 2015.

[40] A. Fernández, S. Garcia, F. Herrera, and N. V. Chawla, "Smote for learning from imbalanced data: progress and challenges, marking the 15-year anniversary," *Journal of artificial intelligence research*, vol. 61, pp. 863–905, 2018.

[41] S. Ertekin, J. Huang, L. Bottou, and L. Giles, "Learning on the border: active learning in imbalanced data classification," in *Proceedings of the sixteenth ACM conference on Conference on information and knowledge management*, 2007, pp. 127–136.

[42] X.-Y. Liu, J. Wu, and Z.-H. Zhou, "Exploratory undersampling for class-imbalance learning," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 39, no. 2, pp. 539–550, 2008.

[43] N. García-Pedrajas, "Partial random under/oversampling for multilabel problems," *Knowledge-Based Systems*, p. 112355, 2024.

[44] E. F. Swana, W. Doorsamy, and P. Bokoro, "Tomek link and smote approaches for machine fault classification with an imbalanced dataset," *Sensors*, vol. 22, no. 9, p. 3246, 2022.

[45] S. K. Kwak and J. H. Kim, "Statistical data preparation: management of missing values and outliers," *Korean journal of anesthesiology*, vol. 70, no. 4, pp. 407–411, 2017.

[46] E. Kock, Y. Sarwari, N. Russo, and M. Johnsson, "Identifying cheating behaviour with machine learning," in *2021 Swedish Artificial Intelligence Society Workshop (SAIS)*. IEEE, 2021, pp. 1–4.

[47] S. Sperandei, "Understanding logistic regression analysis," *Biochemia medica*, vol. 24, no. 1, pp. 12–18, 2014.

[48] J. V. Tu, "Advantages and disadvantages of using artificial neural networks versus logistic regression for predicting medical outcomes," *Journal of clinical epidemiology*, vol. 49, no. 11, pp. 1225–1231, 1996.

[49] J. A. Ruiperez-Valiente, P. J. Munoz-Merino, G. Alexandron, and D. E. Pritchard, "Using machine learning to detect 'multiple-account'cheating and analyze the influence of student and problem features," *IEEE transactions on learning technologies*, vol. 12, no. 1, pp. 112–122, 2017.

[50] M. Sundermeyer, R. Schlüter, and H. Ney, "Lstm neural networks for language modeling." in *Interspeech*, vol. 2012, 2012, pp. 194–197.

[51] G. Edholm and X. Zuo, "A comparison between aconventional lstm network and agrid lstm network applied onspeech recognition," 2018.

[52] W. Liu, W. Jing, and Y. Li, "Incorporating feature representation into bilstm for deceptive review detection," *Computing*, vol. 102, no. 3, pp. 701–715, 2020.

Manit Malhotra is a Research Scholar in the Department of Computer Science Applications, Panjab University, Chandigarh, and an Assistant Professor in the Department of Computer Applications, Government College of Commerce and Business Administration, Chandigarh, since 2019. A UGC-NET qualifier and the highest Gold Medallist of MCA (Panjab University, 2019), he has authored a book, published research in reputed journals including IEEE, and presented papers at numerous national and international conferences. With over 150 awards in coding and public speaking, he has also received multiple accolades for best research papers and poster presentations.

Professor Indu Chhabra Department of Computer Science, Punjab University, is the first professor in the northern region to earn a Post-Doctoral Fellowship (Computer Science) from a top ICSSR institute, Ministry of Education, Govt. of India, specializing in Neural Networks and Genetic Algorithms. She has received multiple honors, including the Young Scientist Award (2007), Award of Honor (2018), and Award of Excellence (2023). With nine Best Research Paper Awards, she serves as a project sanctioning expert for the Dept. of Science and Technology, Govt. of India, and reviews for IEEE, Elsevier, and Springer Nature. Author of four books and seven international book chapters, she has 91 publications and is on various government and semi-government selection committees.