

Article

Proactive Proctoring: A Critical Analysis of Machine Learning Architectures and Custom Temporal Data Sets for Moodle Fraud Detection

Andrei-Nicolae Vacariu , Marian Bucos , Marius Otesteanu  and Bogdan Dragulescu * 

Communications Department, Politehnica University Timisoara, 2 Vasile Parvan, 300223 Timisoara, Romania; andrei.vacariu@student.upt.ro (A.-N.V.); marian.bucos@upt.ro (M.B.); marius.otesteanu@upt.ro (M.O.)

* Correspondence: bogdan.dragulescu@upt.ro

Abstract

This paper examines the use of Machine Learning (ML) approaches in maintaining academic integrity using the information provided in the Moodle system logs. The paper focuses on data set construction, handling the issue of class imbalance, and the assessment of the performance of different ML models in uncovering academic fraud. Twelve different data sets were created by using the concept of temporal windows (e.g., one-day and three-day windows) during the feature extraction stage from the Moodle system logs. The manual labeling of the data sets was done based on a predefined set of rules that outline the fraudulent activities. The issue of class imbalance was treated using eleven different resampling approaches, such as SMOTE, ADASYN, Tomek Links, and NearMiss. We evaluated six classification algorithms, thus resulting in a total of 792 experiments based on the interactions between the data sets, resampling methods, and classification algorithms. The results from the experiment show that the Random Forest and AdaBoost models performed the best in the experiment. Furthermore, we observed a trade-off between fraud detection rates and model precision based on the temporal windows and resampling methods. The shortest temporal windows and hybrid undersampling approaches resulted in the maximum recall value in this study and could identify the greatest number of at-risk students. On the other hand, the longest temporal windows and hybrid oversampling approaches with data cleaning resulted in the best results in terms of F1-Score and Cohen's Kappa statistics. The results provide conclusive evidence that the models can identify fraud; however, they should be used as predictive models for the improvement of proctoring approaches, such as random selection for verification or seating arrangement strategies, instead of judgment models.

Keywords: academic fraud; data mining; educational data mining; fraud detection; machine learning in education; Moodle logs



Received: 27 January 2026

Revised: 21 February 2026

Accepted: 26 February 2026

Published: 28 February 2026

Copyright: © 2026 by the authors.

Licensee MDPI, Basel, Switzerland.

This article is an open access article distributed under the terms and conditions of the [Creative Commons Attribution \(CC BY\)](https://creativecommons.org/licenses/by/4.0/) license.

1. Introduction

Fraud can be defined in many ways. As a general definition, it can be viewed as deceptive actions performed to obtain an undeserved gain [1], but it lacks specific characteristics that can help detect fraud. A more specific definition is provided in [2] as “Fraud is an uncommon, well-considered, imperceptibly concealed, time-evolving, and often carefully organized crime that appears in many types of forms”. The latter provides some key characteristics of fraud that are relevant in detecting it. First, fraud is rare, which means that

fraudulent actions are more sparse than legitimate actions. This translates into imbalanced data sets. Imbalanced data sets present a challenge to current ML algorithms, as they struggle in this scenario, focusing on the majority class and ignoring the minority class [3]. Since most actions are legitimate, this offers the opportunity for fraudulent actions to hide among the legitimate ones. Another important aspect of fraud is that it changes over time. Fraudster behavior adapts over time to elude their identification, so the model must adjust to this change as well.

Academic fraud encompasses any form of unauthorized activities that violate rules and regulations, to gain an unfair advantage and to show a performance that misrepresents one's true ability [4,5]. A problem specific to the academic field is that the perception of students of what constitutes cheating can vary significantly based on their background [5,6]. To address this behavior, students must be informed about academic integrity during classes and reinforce it throughout the course. Furthermore, faculty should provide a clear guideline on what constitutes academic fraud [6]. Cheating affects the validity of the examination in higher education and jeopardizes a fair learning and assessment environment [4]. Studies have shown that teachers are not better at detecting academic fraud than chance [7]. To address this limitation, ML techniques can be applied to detect academic fraud either in real time or after the exam is completed, resulting in better results. However, legitimate privacy concerns arise when detecting academic fraud, given the necessity of handling sensitive data [8].

Detecting fraud has the potential to save companies important amounts of money, as the Association of Certified Fraud Examiners estimates that up to 5% of company revenue can be lost due to fraud [2] or taxpayers that should cover losses in the medical field or other areas of the public sector [9,10]. Detecting academic fraud provides equal opportunities for all students, ensures the quality of education, and maintains academic integrity.

ML techniques have been used in this scope for a long time. Credit card fraud is detected using Logistic Regression (LR), Naïve Bayes (NB), K-Nearest Neighbors (KNN) [11], or Random Forest (RF) and Multilayer Perceptron (MLP) [12]. Fraud detection in the American healthcare system, particularly in Medicare, can be achieved using RF and LR, which are employed in both [9,13]. Additionally, Gradient Tree Boosting (GTB) [9], NB, KNN, Support Vector Machine (SVM), and Decision Tree (DT) [13] are also utilized. RF has been shown to be effective in detecting tax fraud [1], while SVM is capable of detecting money laundering [14]. Clustering is performed to detect anomalies in time series independently of domain [15,16]. Academic fraud is detected using DT, LR, and KNN [17], as well as RF [18].

More recently, deep learning and graph-based techniques have been applied in this field. Graph Neural Networks (GNNs) that utilize both labeled and unlabeled data [19], along with Artificial Neural Networks (ANNs), Recurrent Neural Networks (RNNs), Long Short-Term Memory (LSTM) networks, and Gated Recurrent Units (GRUs) [20], are used for credit card fraud detection. Spam and fake review detection is achieved using Graph-Based methods [21,22] and GNNs [23,24]. Hierarchical Attention Networks [25] and Graph-Based methods [10] are used in detecting tax fraud. ANNs have been shown to be capable in detecting medical fraud [26]. Education fraud is detected using Convolutional Neural Networks (CNNs) [27,28], RNNs [28], and LSTM [28,29].

The hybrid techniques seek to enhance classification performance by stacking multiple classifiers. In credit card fraud, the techniques proposed by [30] involve calculating the similarity between a cardholder's transactions using KNN and making decisions based on Dynamic Random Forest. Another approach for detecting credit card fraud, outlined in [31], employs a hybrid method that combines multiple models, such as NB, DT, and RF, with statistical methods like Adaptive Boosting (AdaBoost) and Majority Voting. Additionally, a

hybrid approach presented in [32], which combines an ANN with RF, has been shown to be effective in detecting spam and fake reviews. Labeling based on K-means clustering is used to train an SVM for detecting academic fraud [6].

Data imbalance is a drawback in all domains where fraud detection occurs [3,9,14,33,34] due to the nature of the fraud. Most of the actions are legitimate in each data set; just a minority are fraudulent actions. Furthermore, in the educational field, collecting reliable data to train detection models is a challenge. Previous work to detect academic fraud has focused on using data from MOOC classes [5,6], using custom evaluation tools to detect illegal actions [8], or relying on synthetic data [29].

This paper shows a new approach that leverages Moodle logs as input data sets for supervised ML algorithms without employing a custom evaluation tool. We selected log data covering the COVID-19 pandemic years preceded and followed by standard academic years. While the standard years provide a baseline of normal behavior, the inclusion of the pandemic period was done to address class imbalance. During the pandemic, examinations were conducted online with limited proctoring mechanisms, which naturally resulted in a higher frequency of fraudulent activities compared to in-class proctored exams. By combining these years, we obtained a data set with a higher representation of the minority class, fraudulent activity, which is essential for training ML models. The primary goal is to develop a robust methodology for extracting, labeling, and processing Moodle platform logs to train ML models capable of detecting fraudulent activities before exams. Although the resulting models might not be applicable to other courses, an educator following the proposed method can train their own.

In line with the above objective, the research is framed by the following research questions (RQ):

RQ1. What is the impact of temporal window size on the discriminative power of features extracted from Moodle activity logs for academic fraud detection?

RQ2. Which combination of resampling strategy and classification algorithm yields the best trade-off between fraud detection rate and precision in a highly imbalanced educational data set?

RQ3. How does the inclusion of auxiliary student metadata (e.g., grade book information) influence the performance of machine learning models in detecting academic fraud compared to using log-derived features alone?

This paper is structured as follows: Section 2 discusses the work related to fraud detection in various domains with a focus on the educational field. In Section 3, the steps taken to create the data sets are explained, followed by an overview of the evaluation of the chosen ML algorithms. The results obtained, clustered according to the research questions, are presented in Section 4. In Section 5, we discuss the results obtained from the experiment. Finally, the conclusions of this study are presented in Section 6.

2. Related Work

2.1. Detecting Fraud in Different Domains

Fraud detection extends beyond the educational field and finds relevance across various domains. Detection of credit card fraud is an extensively studied topic. Supervised classifiers, including LR, NB, KNN, RF, and MLP have been evaluated on public and private data sets [11,12,19,33,35–37]. Multiple studies [30,33,35,38] concluded that LR can be used successfully to detect fraud in credit card transactions. Hybrid approaches that combine multiple classifiers with statistical methods, such as AdaBoost and majority voting, have improved detection performance [30,31]. In the medical domain, studies on Medicare fraud have examined the effect of class imbalance on classifiers' performance.

The experiments compared RUS, ROS, and hybrid resampling strategies across RF, GTB, LR, and ANN [9,13,26].

Several insights from these domains are used in our current study. From the field of financial fraud detection, we adapted the concept of temporal windows for capturing recent behavioral patterns [38] and hybrid approaches for improved performance [31]. Resampling strategies to handle class imbalance are motivated by medical fraud research [13,26]. Ensemble classifiers are included given their strong cross-domain performance [30,31].

2.2. Detecting Academic Fraud

Educational fraud has some unique characteristics that distinguish it from fraud in financial or medical domains. One of the characteristics of educational fraud is the perceived power asymmetry in the instructor–student relationship [39], as the instructor (detector) holds authority over the student (subject). There is also the significant impact of false accusations (false positives) on a student’s education and career path [40], a risk not present in financial fraud detection. Furthermore, there is the cultural background that complicates the ground truth. What is considered “cheating” can vary drastically based on a student’s background [5,6]. Educational fraud also shares some similarities with other domains. The class imbalance issue, as fraudulent actions are less frequent than legitimate ones, the scarcity of publicly available data sets due to privacy concerns, and the need for models to adapt to changing fraud patterns over time represent elements present in every domain.

Detecting fraud in the educational field can be accomplished using an ML approach. Academic fraud can be detected in real time, such as during an exam, through the use of online proctoring methods [17,27,28]. Another approach is to detect academic fraud after the exam has ended, which involves analyzing information that becomes available only once the exam ends [5,6,8,18,29]. Our study focuses on building models that can predict academic fraud before the exam has started, without implementing a real-time proctoring method. This approach to academic fraud detection is less resource intensive and does not raise privacy concerns compared to online proctoring [8].

Haytom et al. [17] use a data set of biometrics and non-biometrics data to train a DT, LR, KNN scheme, and NB classifier for online proctoring. In the paper by Haytom et al., the best results were obtained with DT and LR. Another online proctoring system able to predict fraud during exams is provided in [28]. The training data set consists of three classes: cheating, trying cheating, and not cheating. To detect fraud, the authors train a CNN, an RNN, and an LSTM over more than 7000 selected images of students during exams. The best results were obtained by CNN with an accuracy of 95%, while RNN and LSTM achieved accuracies of 72% and 76%, respectively. These studies show that real-time proctoring can achieve good accuracy results when using multimodal biometrics [17] or deep learning on camera feeds [28]. Concern with these techniques can be the privacy issue raised by the use of biometric data and the hardware requirements for real-time video processing. Our study focuses on predicting fraud before the exam. This eliminates the need for real-time video processing and the use of biometric data, thus addressing privacy concerns and hardware requirements.

Subash and Song [27] propose a Real-time Behavioral Biometric Information Security (RBBIS) architecture based on Convolutional Neural Networks to validate students’ identities during exams. The method learns the behavior of students to validate users and detect intrusion, identity theft, and assessment fraud. The authors use a publicly available data set, the CMU benchmark data set, that consists of keystroke timings of 51 participants over 400 iterations. The parameters included in the data set are Hold time, Keydown–Keydown Time, and Keyup–Keydown Time. The proposed CNN is evaluated against

three ML algorithms: j48, Naïve Bayes, and MLP. The metrics used to compare the results are accuracy, precision, recall, F-measure, kappa statistic, and false positive rate (FPR). To test the algorithms, the data set was split into 90% training and 10% testing. The results indicate that CNN performed the best across all tested metrics. CNN achieved an accuracy of 92.45%, while NB, j48, and MLP achieved accuracies of 68.87%, 73.38%, and 77.11%, respectively. This implementation relies on keystroke dynamics, which is not available by default in standard Moodle logs. Our approach focuses on using the information available in Moodle logs, which does not require additional hardware or software for data collection, thus making it more accessible to a wider range of educational institutions.

Sangalli et al. [6] perform K-means clustering-based labeling on a data set, which is then used to train an SVM classification model. They identify academic fraud, such as students who collaborate by sharing answers (collaborators) or those students who use fake accounts to obtain correct answers for exercises (harvesters). The data used in this experiment consists of log actions performed by students: interaction with the course material, exercise submission attempts, forum posts, etc. The log is processed to keep only exercise-related information: timestamp, student answer, and whether is correct or not. This information is then used to generate metrics that can be used to identify harvesters and collaborators. They define the metric *User Bias* that helps identify who copied from whom. The metric *Final Score Difference* is defined as the difference between the final score obtained by any pair of accounts. If a pair of users solves the exercises within a time window and has a high final score difference, it can suggest that this pair is composed of a real user (harvester) and a fake account. If the difference is low, it can indicate that two real students collaborate. Another metric is *Material Interaction Rate* that captures how much time a student spent trying to solve exercises versus the total amount of time spent taking the course. The last metric used is *Percentage of Shared IP Numbers* as the percentage of shared IP addresses between two users. The data set features are standardized by removing the mean and scaling to unit variance. The scaled data set is used to identify harvesters and collaborators using K-means, resulting in a labeled data set. Sangalli et al. [6] used the data set to train the SVM classification model in order to distinguish between the two classes using the previous metrics. An SVM model is used with Linear and Radial Basis Function kernels. They perform within-train grid-search to tune the values of C and gamma for RBF kernel and of C for Linear kernel. They use a 10-fold cross-validation to evaluate the model's performance. The results show that SVM is capable of distinguishing between collaborators and harvesters with an accuracy of 95% in all scenarios tested. Sangalli et al. [6] successfully identified "harvesters" and "collaborators" without the need for labeled training data. Nevertheless, this approach detects fraud that has already taken place. In our study, we use fixed rules to manually label the data set. To handle the class imbalance problem, we benchmark 11 resampling strategies. With this strategy, we focus on predicting fraud before it occurs, allowing for intervention rather than just punishment.

Cheaters in MOOC classes are found using anomaly detection and supervised ML in [5]. Similar to [6], they identify cheaters as learners who perform actions that violate edX's code of honor, such as copying using multiple accounts (CUMA) or sharing responses with peers (collaborators). The data set is based on information provided from two courses offered on edX, such as clickstream events and learner inputs. In line with the common approach from this field of applications, the authors used data to compute two types of features: video-based and assessment-based. Video-based features are *Mean video-time-percentage*—the mean value of watch time for the videos in the course, and *Standard deviation video-time-percentage*—the standard deviation of watch time for the videos in the course. Assessment-based features are *Guttman Error*—the number of pairs of items in which an easier item is answered incorrectly, and a more difficult item is answered correctly,

normalized by the total number of pairs, $ZU3$ —a nonparametric person-fit statistic, and Guttman Error on response-time—Guttman Error, but “difficulty” is interpreted based on the mean response time that is required to solve the item correctly. The authors evaluate the performance of classifiers in four scenarios using 100 five-fold cross-validation runs. This is a good validation of the generalization ability of the model across different courses, ranging from evaluating the model inside the same course, to evaluating the model on courses that were not part of training data. The performance ranges from a mean AUC value of 0.9 (inside the same course) to a mean AUC value of 0.7 (across different courses). The limitation of this study is that the feature engineering is based on the MOOC platform edX. It relies on video-consumption features. These features are less relevant in standard university courses, where video watching is not the primary learning activity. Our feature set is derived from usual learning interactions on a Moodle platform, such as quiz attempts, assignment responses, and resource access. Furthermore, we designed 12 data sets based on several temporal windows and auxiliary data. This allows us to detect specific behavioral shifts as the exam approaches, not just relying on semester-long patterns.

A different approach to detecting academic fraud is taken by Kamalov et al. [29] who treat the issue of detecting potential cheating cases as an outlier detection problem. They employed a different strategy than what was used in the previous papers, where student’s actions are used as input data set. The authors used students’ continuous assessment results throughout the semester to identify abnormal scores on the final exam. The data sets used in the experiments consist of three synthetic and one real-life data sets. All synthetic data sets consist of 100 normal-grade sequences and 10 cheating cases. Data set 1 consists of fraud cases with a 35-point difference between the average scores during the semester and the final exam score. Fraud is better masked in data set 2 with a 20-point difference between the average and the final exam score. Data set 3 contains honest grades that are increasing throughout the semester in such a way that the difference between the average score during the semester and the final exam is the same as the cheating cases. A two-step process is used to detect academic fraud. An LSTM neural network is trained to predict the final exam score for each student. The input features for LSTM included scores on quizzes, midterm exams, projects, and other pre-final exam assessments. The output of the model is the score on the final exam. To identify cheaters, the authors calculate the error between the LSTM predicted scores and the actual exam scores. They employ an outlier detection method based on kernel density estimation (KDE) on the set of errors to identify potential cheating cases. The results showed an average of 95% TPR and 2.5% FPR on synthetic data, with 100% TPR and 4% FPR on the real-life data set. A limitation of this study is that validation relies primarily on synthetic data, with three out of four data sets being synthetic. We validate our approach on real-world Moodle logs spanning multiple years and courses.

Another two-step process was applied in [18] to detect cheaters from a high school final exam. Similar to Kamalov et al. [29] they use ML to predict students’ results in the final exam. The authors identify cheaters as those who show a big difference between reality and prediction and collaborate with peers sitting nearby. A difference from [29] is that RF is used for predictions instead of LSTM. The data set consists of high school students enrolled in three classes. Students’ answers to questions throughout the semester are utilized as training data for RF, with the predicted value corresponding to the answer to an exam question. The questions are linked by knowledge points, meaning that the questions that cover the same subject share the same knowledge point. The first step of the process involves using RF to predict, for each question, whether the student will provide the correct answer. A divergence rate is computed as the percentage of questions where the student’s answer was not the same as the predicted answer, a similar approach to [29] where they computed the difference between the actual and predicted scores. The authors

mark as “Suspicious student” those students with a divergence rate greater than 50%. The suspicious student’s results are compared against the results of the other four students who were seated around them. If the answers between a pair of two students match in more than 50% of the cases, both students are marked as cheaters. This represents another difference compared to [29], where a kernel density estimation method was utilized. The results from [18] showed a mean precision value of 82.3% and a mean recall value of 72.4% over the three courses for the proposed solution. Their input data set included the score on quizzes, midterm exams, projects, and other pre-final exam assessments. The output of the model is the score on the final exam. A drawback of this method is that it requires seating charts, limiting the application of the model to in-person exams. Our approach can be applied both to in-person and online courses, as it does not rely on physical constraints.

A summary of the related work in the educational field and the differentiation of our study is provided in Table 1. The table compares the method, key strength, key limitation, and differentiation of our study for each of the related works discussed above.

Table 1. Comparison of related work and differentiation of the proposed study.

Reference	Method	Key Strength	Key Limitation	Differentiation in Our Study
Haytom et al. [17]	Biometric and Non-biometric (DT, LR, KNN)	Multimodal approach for online proctoring.	Requires specialized hardware; not scalable to standard LMS.	Scalability: Uses standard Moodle logs available in any installation; no extra hardware required.
Elkohli et al. [28]	Deep Learning (CNN, RNN, LSTM) on video/images.	High accuracy (95%) with CNNs for real-time monitoring.	Privacy invasive (camera required); high resource cost.	Privacy: Non-intrusive; analyzes log metadata rather than video feeds or student images.
Subash & Song [27]	Behavioral Biometrics (Keystroke Dynamics)	High precision in identity verification using CNNs.	Data availability: Keystroke timings are not captured by standard LMS logs.	Data Source: Relies on native Moodle log data, avoiding the need for custom client-side scripts.
Sangalli et al. [6]	Unsupervised Clustering (K-means) and SVM	Effective at labeling without ground truth (detects “harvesters”).	Label reliability: Relies on cluster density assumptions rather than confirmed fraud cases.	Imbalance Handling: We use rule-based ground truth and explicitly tackle class rarity using 11 resampling methods (SMOTE, ADASYN, etc.).
Alexandron et al. [5]	Anomaly Detection on MOOC data (edX)	Strong cross-course generalization testing.	Feature granularity: Relies on coarse, MOOC-centric aggregates (e.g., video watch time).	Temporal Features: We use granular temporal windows (1, 3, 5, 7 days pre-exam) to capture evolving behavior, generating 12 distinct data sets.
Kamalov et al. [29]	LSTM + Outlier Detection	Novel use of sequential data for grade prediction.	Validated primarily on synthetic data; lacks real-world noise.	Validity: Validated on real-world data from multiple years/courses; combines activity logs with grade book data.
Hu et al. [18]	RF Prediction + Physical Context	Combines grade prediction with seating proximity.	Physical constraint: Requires seating charts; not applicable to remote/digital exams.	Environment: Designed for digital/remote learning where physical proximity is irrelevant.

2.3. Data Availability and Class Imbalance

The key drawbacks of applying these techniques are represented by data availability and class imbalance. Of the eight studies on academic fraud detection included in this paper, only one utilized publicly available data sets in its experiments. Subash and Song [27] use the CMU benchmark data set [41], a publicly available data set that comprises password keystrokes of users.

Data imbalance is an intrinsic problem in fraud detection, as fraudulent actions are sparse compared to honest ones. A common approach to addressing this issue is to

resample the data set. Resampling is performed to create an equal number of instances across classes. If the number of instances in the majority class is reduced to match that of the minority class, the process is known as undersampling. An example of this method is RUS. The main limitation of Random Undersampling is that some information could be lost [11]. Alternatively, increasing the number of instances in the minority class is referred to as oversampling. ROS and Synthetic Minority Oversampling Technique (SMOTE) are common oversampling techniques. SMOTE is not just copying values from the minority class to balance the data set [42]; the algorithm takes each sample from the minority class and computes its k-nearest neighbors. Then it generates new data based on a combination of features from the sample and its neighbors. The number of elements being generated is dependent on the imbalanced characteristic of the data set.

SMOTE has been successfully used to balance a federated data set [43], a public data set for credit card fraud detection [2,12,38,42]. The same data set was also balanced using (RUS) [11]. RUS has been used to balance other data sets for credit fraud detection [30], a data set for medical fraud detection [13], and a data set for money laundering detection [14]. When evaluating the performance of Deep Neural Networks (DNNs) based on RUS, ROS, and a combination of ROS-RUS resampling techniques, it was concluded that the best results are obtained with ROS and RUS-ROS on a data set for medical fraud detection [26].

3. Materials and Methods

3.1. Data Set Construction

Politehnica University Timisoara operates an online learning environment based on the open-source Moodle system [44], known as Campus Virtual [45]. The data set utilized in this paper was derived from logs recording the actions of users on this platform, spanning three courses across three academic years. The specific procedure followed to construct this data set is outlined in Algorithm 1.

Algorithm 1 The procedural pipeline for constructing the ML data set, encompassing raw log filtering, feature extraction, and integration with auxiliary grade book data

Input: cv.upt.ro

Output: ML Data Set

```

1: Retrieve raw logs from cv.upt.ro
2: Encrypt user specific information
3: Get semester start and stop time
4: Determine list of active students
5: for all logs do
6:   if (log.time < semester_start OR log.time > semester_stop) then
7:     Remove log
8:   end if
9:   if (log.student ≠ active_student) then
10:    Remove log
11:   end if
12: end for
13: Map student to exam information
14: for every student – exam pair do
15:   Count internal elements access
16:   Count external elements access
17:   Check fixed rules violation
18:   Check flexible rules violation
19: end for
20: Merge current data with Grade Book Data Set
21: Filter and remove duplicates
22: Filter and remove students that withdraw from course

```

3.2. Logs Retrieval

We have collected logs from three different courses: LP2 (Programming Languages 2), POO (Object-Oriented Programming), and BD (Database Management Systems). These courses are part of the Electronics, Telecommunications and Information Technology (ETTI) specialization offered by University Politehnica Timisoara (UPT), a technical university from Romania. The students are undergraduate students in their second and third year of study. Due to their technical background, the students have a good digital literacy and technical proficiency. The students' profile is similar across the three courses, as they are all part of the same specialization. The courses are offered in the first and second semester of the academic year. The logs are generated by students' interactions with the Moodle platform, which include activities such as accessing course materials, submitting assignments, or responding to quizzes. Because of this students' specialization, it is important to note that these interaction patterns may differ from those observed in non-technical disciplines, such as the humanities.

We selected log data covering both standard academic years and the COVID-19 pandemic period to address class imbalance. It is important to note that the pedagogical structure and the actual activities on the Moodle platform remained the same across both formats. The difference between these two educational environments is represented by the physical location of the exam and the efficiency of the proctoring. While face-to-face exams took place inside the classroom with strict physical oversight, online exams were conducted remotely with less efficient proctoring constraints. The proctoring method used for online exams only required for cameras to be turned on during the exam. We assume that this environmental change from face-to-face to online exams yields a higher incidence of academic fraud due to the reduction in physical proctoring. Including the online format was intentional as it better represents the minority class (fraud) in the final data set.

The logs are obtained from the following academic years: 2019–2020, 2020–2021, 2021–2022. Of the three courses, one takes place in the first semester from September to February (POO), while the other two courses (LP2, BD) take place in the second semester from February to June. The COVID-19 pandemic forced classes to take place in an online environment during the second semester of 2019–2020, the entire academic year 2020–2021, and the first semester of 2021–2022. Table 2 illustrates the distribution of data regarding the number of students and the number of logs generated across the three courses and the three university years. Campus Virtual is an e-learning platform that can be used in face-to-face courses as well. Therefore, the available logs are generated during two semesters of face-to-face classes and four semesters of online classes.

The initial logs, as can be seen in Figure 1, consist of the time the action was performed, the user's name, the course's component interacted with, a description of the action, the source of the action (web, app, cli), and the IP address that was used. To protect the privacy of users, all user-specific information has been encrypted using a one-way cryptographic function, md5 with a custom salt string. Furthermore, one of the course instructors implemented data encryption. Appropriate measures were taken to safeguard sensitive data before preprocessing. The sensitive information consists of the user's ID, IP address, and name.

Besides the logs generated by the students on the Moodle platform, we also had access to the students' grade book for each course. Its content included the student's encrypted identifier, gender, and grades, as well as a flag indicating students who took only the final examination, their course activity having been previously finalized.

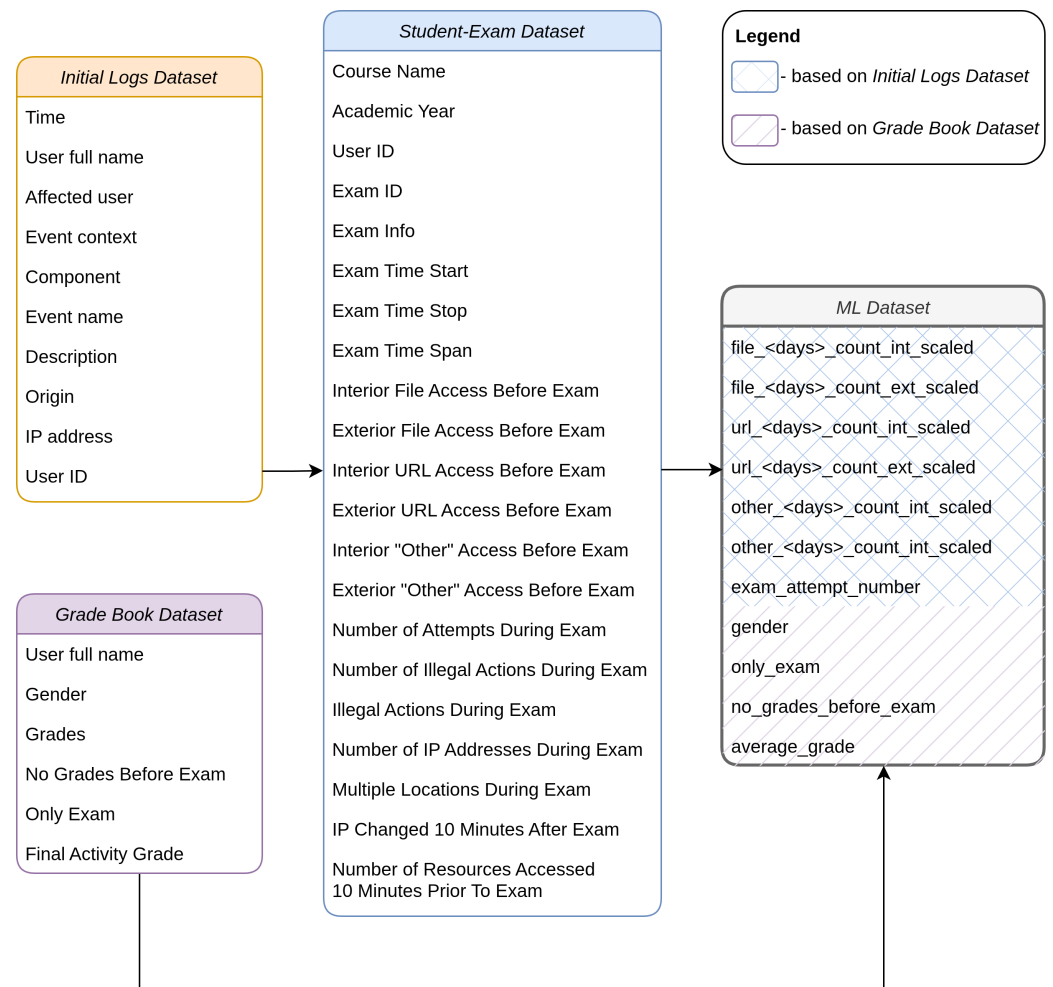


Figure 1. Feature engineering architecture. Visual overview of the attribute template used to construct the fraud detection data sets, highlighting the distinction between dynamic log metrics, subject to temporal windowing, and fixed student metadata.

Table 2. Distribution of active students and generated activity logs stratified by course and academic year.

Metric	LP2	POO	BD
2019–2020			
Active Students	112	118	143
Logs Generated	102,784	88,044	120,328
2020–2021			
Active Students	115	111	139
Logs Generated	75,612	63,806	119,185
2021–2022			
Active Students	127	128	128
Logs Generated	84,366	82,546	87,166

3.3. Data Preprocessing

The goal of data preprocessing is to map information about each student and the exams they attended during a course. The information obtained after this step is stored in the Student–Exam data set, Figure 1.

The first step taken was to filter out the logs that took place outside the semester’s timeframe, as maintenance work can reflect in the logs, but it is not relevant to our study.

Afterward, we extracted the ID of the resource the user interacted with and appended it to the information provided by logs. The resource ID can depict the type of resource like a file, a URL, an exam activity, or others.

Since a student can take part in some activities from a semester, but decide not to take any exams, we need to remove these students' actions from the logs. A list of the exams that took place in the respective semester can be retrieved from the logs. Based on the list of exams, we were able to extract the list of active students from that semester, the students who attended at least one exam. The list of active students is used to filter the logs once more and keep only the logs performed by those who were active. We use this filtering process to keep only information related to the active students. This approach is similar to that of Sangalli et al. [6], as they processed log actions to retain only exercise related information.

In the *Student–Exam data set*, each student is paired with the exams they took the corresponding semester. The origin of the data is maintained by features *Course Name* and *Academic Year*. Based on the logs, the start and stop time of the exam for the student–exam pair can be retrieved, and the duration of the exam can be computed. To have quantifiable metrics about a student's efforts to prepare for an exam, we counted on how many occasions the available resources for exam preparation were accessed by the student before the exam started from the beginning of the semester. With this approach we obtain a quantifiable measure of a student's involvement with the course directly from logs. This is an alternative to methodologies that rely on grades from quizzes or midterms to predict a final result [18,29]. The resources are split into three categories: File, URL, and Other. A second-level differentiation is performed based on the location of resource access: "Interior" if accessed on faculty premises during classes, or "Exterior" if accessed from external devices outside scheduled class hours. This can be achieved by looking at the IP address and comparing it with the list of IP addresses used in the laboratory. In the existing literature, shared IP addresses are used to identify collaborating students [6]. We use the IP data to establish the location of the students' preparation and exam environment. A parameter that conveys the number of times a student looked through their exam questions is saved as *Number of Attempts During Exam*. For later usage in the labeling process, we retrieve how many resources are accessed during the exam; since this is not allowed, these are saved as "Illegal Actions". We also check if the student opened multiple sessions of the exam from distinct locations (the IP changes quickly after the exam), and the student's behavior in the minutes preceding the exam. The detection of rapid IP changes to identify multiple sessions aligns with the identification of copying using multiple accounts in the edX platform [5]. We use this rule-based system to establish our ground truth. An alternative to our approach is presented by unsupervised clustering used for labeling [6].

3.4. Data Labeling

To apply supervised ML algorithms, the data set must have a target value. Since the logs only supply information about what actions have been performed, the target has to be manually created. We created the label based on two sets of rules: "Fixed" and "Flexible", described in Table 3. If at least one of the fixed rules is broken, we know for sure that the student defrauded the exam and set the label accordingly. Flexible rules cannot determine fraud, but rather highlight suspicious behavior.

Table 3. Taxonomy of examination rules and behavioral indicators. This table differentiates between fixed rules, representing strict institutional prohibitions where violations confirm fraud, and flexible rules, which function as heuristic indicators to flag potentially anomalous behaviors for further investigation.

Rule ID	Description
Fixed Rules	
Rule #1	A student is not allowed to access any kind of resources during an exam.
Rule #2	Exams are non-transferable and must be completed by the registered student themselves.
Flexible Rules (Fraud Indicators)	
Rule #1	Accessing resources in the minutes prior to an exam might indicate a fraudulent behavior.
Rule #2	Changing the IP address in the minutes after the exam might indicate a fraudulent behavior.

Fixed Rule #1 prohibits students from accessing course resources or any other resources during the exam. Failure to follow this rule, and thus accessing any resource, will result in the student being marked as a cheater (committing fraud).

Fixed Rule #2 prohibits impersonation by any person acting on behalf of the student. This can be verified by tallying the number of unique IP addresses used during the exam. If multiple IP addresses are detected, it indicates that the same user account has been logged in from different computers. This suggests that someone is impersonating the student and therefore violating this rule. As a result, the student is flagged as a cheater (committing fraud). This approach can detect identity theft based on the IP address change from the standard Moodle logs. It presents an alternative to systems requiring specialized behavioral biometrics, such as keystroke dynamics [17,27], which is not available by default on the Moodle platform. Detecting rapid IP changes to identify multiple sessions aligns with the identification of copying using multiple accounts in edX platform [5].

Two fixed rules were chosen because they detect fraudulent behaviors that can be definitively verified from Moodle activity logs alone. Fixed Rule #1 is verifiable because Moodle logs every resource access with a timestamp. Fixed Rule #2 is verifiable because Moodle records IP addresses for each user action. Conversely, copying from printed materials, looking at a neighbor's screen, or downloading resources before the exam for offline access leave no trace in the platform logs. To detect these actions requires physical proctoring or additional monitoring, such as real-time camera feeds [28]. By restricting labeling to log-verifiable violations, we ensure 100% labeling precision at the cost of uncertain labeling recall.

Flexible rules can be violated without necessarily labeling the student as committing fraud; they serve as indicators of suspicious behavior.

Fixed Rule #1 can be avoided by downloading exam resources before the start of the exam. In this manner, a student could utilize the resources during the exam without leaving any trace in the logs. *Flexible Rule #1* is designed to detect such behavior.

The students can avoid *Fixed Rule #2* by not logging in during the exam. The single login can be executed by the impersonator, thereby leading to the identification of a unique IP address during an exam. The actual student may choose to log in only after the exam has concluded. *Flexible Rule #2* is designed to identify this scenario.

3.5. ML Data Set

The framework presented in Figure 1 represents a template that was used to derive 12 distinct data sets by combining the available features. The description of the attributes found in this template is provided in Table 4. The distinct data sets were derived from the standard data set by adjusting the attribute computation interval and selectively including auxiliary information, as illustrated in Figure 2.

Table 4. Description of the feature set utilized for model training. The table defines the derived log metrics and the auxiliary student metadata used to construct the experimental data sets.

Attribute	Description
file_<days>_count_int_scaled	Number of file accesses made by students during classes, normalized.
file_<days>_count_ext_scaled	Number of file accesses made by students outside classes, normalized.
url_<days>_count_int_scaled	Number of URL accesses made by students during classes, normalized.
url_<days>_count_ext_scaled	Number of URL accesses made by students outside classes, normalized.
other_<days>_count_int_scaled	Number of other resource accesses made by students during classes, normalized.
other_<days>_count_ext_scaled	Number of other resource accesses made by students outside classes, normalized.
exam_attempt_number	The sequential number of the exam attempt made by the student.
gender	The gender of the student.
only_exam	Indicator if the student participated only in the exam without prior activity.
no_grades_before_exam	Indicator if the student had no recorded grades prior to the exam.
average_grade	The average grade obtained by the student.

Note: The placeholder <days> represents the observation window before the exam, where <days> $\in \{0, 1, 3, 5, 7\}$. The value 0 denotes the start of the semester.

ML Dataset Template	
Standard Data	file_<days>_count_int_scaled file_<days>_count_ext_scaled url_<days>_count_int_scaled url_<days>_count_ext_scaled other_<days>_count_int_scaled other_<days>_count_ext_scaled
Auxiliary Data	exam_attempt_number gender only_exam no_grades_before_exam average_grade

Figure 2. Structure of the ML data set template. The attributes are categorized into Standard Data (log-derived metrics where <days> represents the variable computation interval) and Auxiliary Data (grade book information).

The configuration of the experimental data sets is summarized in Table 5. In the standard data set, each metric is computed cumulatively from the beginning of the semester. In contrast, the alternative data sets utilize temporal windows of 1, 3, 5, or 7 days for attribute calculation. The granular approach to define computation intervals is adapted from financial fraud detection [38]. Auxiliary information encompasses the number of exam attempts and the metadata available in the student’s grade book. The final grade was intentionally excluded from the final data set, as the model is designed to detect academic fraud proactively, before the issuance of the final grade. This design choice distinguishes our framework from studies where grades received during a semester are used to identify fraud based on outlier detection after the final exams are completed [18,29].

Table 5. Summary of experimental data sets defined by temporal observation windows (0–7 days). Base features (6 per window) track core resource access metrics, while the “Extra” set incorporates auxiliary data, including student profile variables such as gender and academic history.

Data Set Name	Content Description	Feature Count
original	Cumulative metrics calculated from the start of the semester (0 d).	6
original + extra	Cumulative metrics (0 d) combined with student metadata (grades, gender, exam attempts).	11
1 d	Metrics aggregated over the last 1 day prior to the exam.	6
1 d + extra	Metrics from the last 1 day combined with student metadata.	11
3 d	Metrics aggregated over the last 3 days prior to the exam.	6
3 d + extra	Metrics from the last 3 days combined with student metadata.	11
5 d	Metrics aggregated over the last 5 days prior to the exam.	6
5 d + extra	Metrics from the last 5 days combined with student metadata.	11
7 d	Metrics aggregated over the last 7 days prior to the exam.	6
7 d + extra	Metrics from the last 7 days combined with student metadata.	11
all	Combination of all time-window metrics (0 d, 1 d, 3 d, 5 d, and 7 d) without metadata.	30
all + extra	Combination of all time-window metrics (0 d, 1 d, 3 d, 5 d, and 7 d) combined with student metadata.	35

Different courses or the same course in different academic years may contain a different number of resources with which a student interacts. To merge the data into the same data set, standardizing the features is necessary. Standardizing the metrics is important to achieve generalization across courses. This is confirmed by Alexandron et al. [5] in their multiple courses evaluations. For both approaches, we scaled the number of resources the student accessed before the exam. The mean and standard deviation were computed for each exam and the samples were scaled based on (1).

$$scaled_value = \frac{current_sample_value - mean}{standard_deviation}. \quad (1)$$

3.6. Data Balancing

An important aspect of the prediction’s performance is the distribution of entries among the classes. As can be seen in Figure 3, the original data set (indicated by the *None* label) is highly unbalanced, with the *Fraud* class representing only 5.83% of the total entities. Beyond this baseline, the figure also details the comparative class distribution of *Non-Fraud* vs. *Fraud* entities for every balancing method evaluated in this study. These values are calculated on the entire data set and serve strictly to aid in understanding the data and the balancing methods. In our experiment, we employed a Repeated Stratified K-Fold procedure to split the data into training and testing sets. The balancing techniques were applied exclusively to the training data, leaving the testing data unmodified.

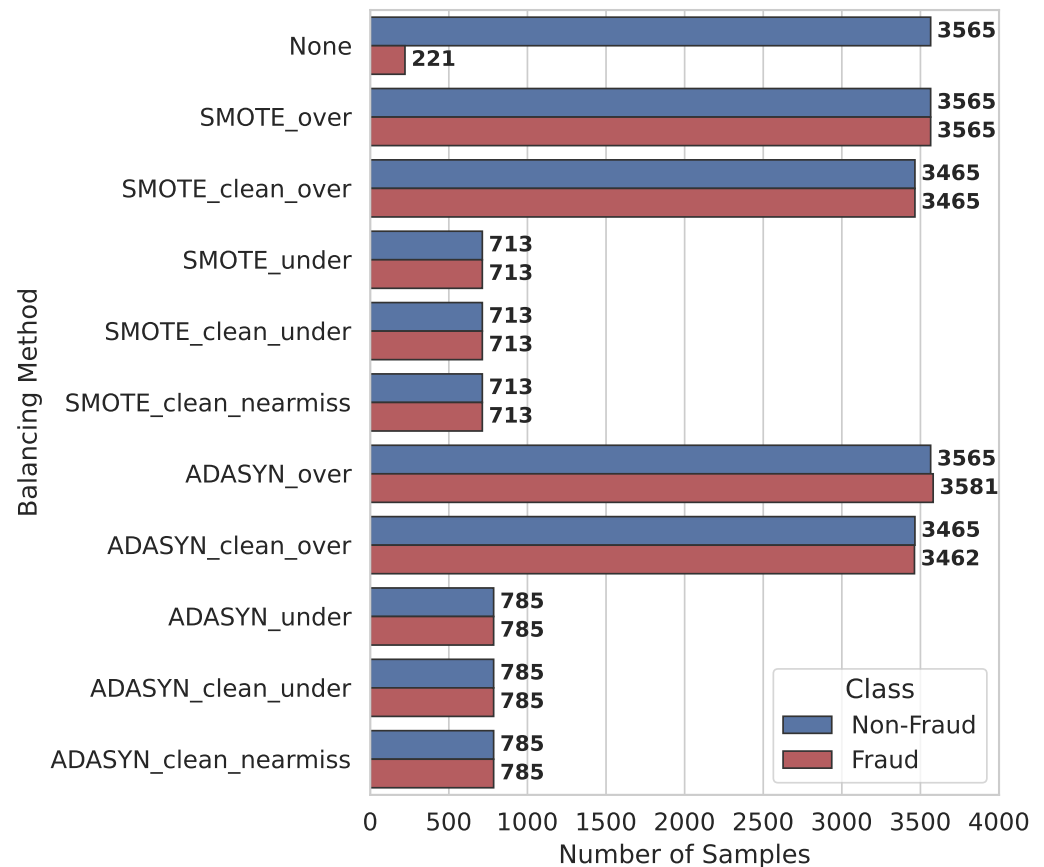


Figure 3. Class distribution across balancing methods. The chart illustrates the impact of the evaluated resampling protocols on the ratio between the minority (fraud) and majority classes, comparing the original imbalanced state against the outcomes of oversampling (SMOTE, ADASYN), undersampling (RUS, NearMiss), and hybrid pipelines.

To solve the unbalanced data set issue, we evaluated eleven resampling protocols, ranging from oversampling to hybrid pipelines. Table 6 describes the specific balancing techniques evaluated. We employ SMOTE and *Adaptive Synthetic Sampling* (ADASYN) for oversampling the minority class. ADASYN was selected to prioritize minority instances located near the decision boundary that are more difficult to learn [46]. These resampling methods were used both in isolation and within pipelines.

To address the potential for overfitting and noise amplification associated with oversampling, we introduced *Tomek Links* as a data cleaning step. A Tomek Link is defined as a pair of instances from different classes that are nearest neighbors to each other. These instances, which are close to the decision boundary, can be removed. Eliminating these pairs effectively widens the boundary between the two classes, thereby potentially enhancing the classifier's performance. Furthermore, we implemented hybrid sampling architectures that combine oversampling (targeting a minority ratio of 0.2) with subsequent undersampling using RUS or *NearMiss* to achieve a balanced data set. The efficiency of hybrid resampling strategies for preserving minority class structure in highly skewed data sets has been validated on fraud detection in the medical field [9,13,26]. This sequential approach aims to balance the data set while preserving the structure of the minority class.

Table 6. Summary of data balancing methods and techniques.

Balancing Method	Description	Sequence of Techniques
None	No balancing technique applied. The original imbalanced data set is used.	None
SMOTE_over	SMOTE is applied to increase the number of instances in the minority class.	1. Oversampling: SMOTE
SMOTE_clean_over	Tomek Links remove noisy instances, followed by SMOTE oversampling.	1. Cleaning: Tomek Links 2. Oversampling: SMOTE
SMOTE_under	Combination of SMOTE oversampling and Random Undersampling.	1. Oversampling: SMOTE (ratio 0.2) 2. Undersampling: Random (ratio 1)
SMOTE_clean_under	SMOTE oversampling, Tomek Links cleaning, and Random Undersampling.	1. Oversampling: SMOTE (ratio 0.2) 2. Cleaning: Tomek Links 3. Undersampling: Random (ratio 1)
SMOTE_clean_nearmiss	SMOTE oversampling, Tomek Links cleaning, and NearMiss undersampling.	1. Oversampling: SMOTE (ratio 0.2) 2. Cleaning: Tomek Links 3. Undersampling: NearMiss
ADASYN_over	ADASYN generates synthetic samples focusing on hard-to-learn examples.	1. Oversampling: ADASYN
ADASYN_clean_over	Tomek Links are first used to clean the data, followed by ADASYN oversampling.	1. Cleaning: Tomek Links 2. Oversampling: ADASYN
ADASYN_under	Combination of ADASYN oversampling and Random Undersampling.	1. Oversampling: ADASYN (ratio 0.2) 2. Undersampling: Random (ratio 1)
ADASYN_clean_under	ADASYN oversampling, Tomek Links cleaning, and Random Undersampling.	1. Oversampling: ADASYN (ratio 0.2) 2. Cleaning: Tomek Links 3. Undersampling: Random (ratio 1)
ADASYN_clean_nearmiss	ADASYN oversampling, Tomek Links cleaning, and NearMiss undersampling.	1. Oversampling: ADASYN (ratio 0.2) 2. Cleaning: Tomek Links 3. Undersampling: NearMiss

We used the Python implementation available in the *imbalanced-learn* library [47] for the oversampling, undersampling, and cleaning techniques. We used Python 3.12, imbalanced-learn version 0.14.0 and sklearn version 1.6.2.

3.7. ML Algorithms

Detecting fraud is solving a binary problem, whether the activity is fraudulent or not. To ensure that the same class balance is kept between train and test, we implemented a Repeated Stratified K-Fold cross-validation with five folds and three repetitions. Its Python implementation is available in the *sklearn.model_selection* library. We used sklearn version 1.6.1. By employing stratified sampling, we ensure that the class distribution remains consistent across all training and testing partitions. This is critical in fraud detection to prevent the formation of unrepresentative folds due to class imbalance. In this configuration, the data set is divided into five distinct folds; in each iteration, one fold serves as the testing set, while the remaining four folds constitute the training set. This entire cycle is repeated three times with different randomization to ensure the stability of the results.

In this study, we employ six classifiers: LR, DT, SVM, Gradient Boosting Machine (GBM), AdaBoost, and RF. Our selection of these classifiers is based on two criteria. Firstly, the classifiers must have demonstrated good results in detecting fraud. Secondly, they must be capable of handling small data sets.

LR has demonstrated good performance in detecting fraudulent credit card transactions [11,30,33,35,38] and medical fraud within the United States' healthcare system [9]. LR and DT have been successfully used to create an authentication solution based on facial recognition and keystroke dynamics for preventing academic fraud [17]. SVM has proven effective in detecting cheating students in Massive Open Online Courses (MOOCs) [6].

Regarding detection of credit card fraud, Randhawa et al. [31] demonstrated that AdaBoost is capable of detecting fraudulent transactions, while Dhankhad et al. [35] and Taha and Malebary [48] successfully used variants of GBM. Meanwhile, Varmedja et al. [12] concluded that although LR can achieve better recall for the fraudulent class, it is surpassed by RF, which shows better precision and overall accuracy. In detecting academic fraud, RF has demonstrated strong performance in predicting whether students provide correct or incorrect answers, thus identifying as cheaters those who consistently provide answers that do not align with their usual performance patterns. [18]. Moreover, RF has proven good results in detecting tax fraud [1], as well as medical fraud [13]. GBMs were also employed in the medical domain, where Herland et al. [9] utilized them for Medicare fraud detection.

LSTM has been used to detect academic fraud [29], credit card fraud [20], and tax fraud [28,29]. ANNs have been used to detect credit card fraud [20] and medical fraud [26]. Academic fraud has been detected using CNNs [27,28]. Deep learning approaches are capable of detecting fraud; however, our data set encompasses, before applying any resampling technique, a total of 3788 entries. Considering that deep networks generally perform better with large data sets [12], and given that our data set is small, we decided not to use any deep learning networks in our experiment.

In our study, we utilized the Python implementation of these classifiers available in the sklearn library.

3.8. Evaluation Metrics

The efficacy of ML classifiers is often assessed through a series of evaluation metrics. We use several key evaluation metrics that are commonly used to gauge classifier performance. To evaluate the performance of the classifiers, we employed the following metrics: precision, recall, F1-Score, balanced accuracy, G-Mean, and Cohen's Kappa. Our data set consists of two classes: Fraud and Non-Fraud. The data set is imbalanced, with the majority Non-Fraud class representing 94.17% of the total entries.

To better understand the evaluation metrics, we define the components of the confusion matrix for a binary classification problem. True Positives (TPs) are instances correctly classified as Fraud, True Negatives (TNs) are instances correctly classified as Non-Fraud, False Positives (FPs) are Non-Fraud instances incorrectly classified as Fraud, and False Negatives (FNs) are Fraud instances incorrectly classified as Non-Fraud. Based on these components, we define the True Positive Rate (TPR) based on (2) and False Positive Rate (FPR) based on (3).

$$TPR = \frac{TP}{TP + FN} \quad (2)$$

$$FPR = \frac{FP}{FP + TN} \quad (3)$$

3.8.1. Minority Class Centric Metrics

To overcome the limited number of entries in the minority class, we employ evaluation metrics focused on this class. These metrics are less sensitive to class imbalance, as their calculation does not rely on the majority class. Evaluation metrics suitable for this scenario are: precision, recall, and F1-Score.

Precision is the ability of the classifier not to label as positive a sample that is negative. It is calculated based on (4). Precision is a metric that is computed for each class of the output label. The value range is between 0 and 1, where 0 shows that none of the predictions for a certain class are correct, and 1 shows that all predicted instances for a certain class are correct.

$$Precision = \frac{TP}{TP + FP} \quad (4)$$

Recall represents a classifier's ability to identify a certain class. It is calculated based on (5). Recall is computed for each class of the output label. It ranges between 0 and 1, where 0 indicates that the classifier is unable to identify any instance of a certain class, and 1 indicates that the classifier can identify every instance of a certain class.

$$\text{Recall} = \text{TPR} \quad (5)$$

F1-Score is the harmonic mean of the precision and recall. It is calculated based on (6). The F1-Score ranges from 0 to 1, with the worst score at 0 and the best score at 1.

$$F1 = \frac{2 \times TP}{2 \times TP + FP + FN} \quad (6)$$

3.8.2. Balanced Aggregation Metrics

Another way to handle the imbalanced data set is to use balanced metrics, such as: balanced accuracy, G-mean, and Cohen's Kappa. These metrics normalize performance contribution across classes, treating the minority class as equally important to the majority class.

Balanced accuracy is an alternative to accuracy that takes into account the class imbalance in the data set. It is calculated as the average of the TPR and TNR, as shown in (7). Balanced accuracy ranges from 0 to 1, where 0.5 indicates random guessing, and 1 represents perfect classification.

$$\text{Balanced Accuracy} = \frac{1}{2}(\text{TPR} + \text{TNR}) \quad (7)$$

G-Mean represents the geometric mean of the True Positive Rate (TPR) and True Negative Rate (TNR). It indicates the balance between the classification performance on the majority and minority classes. G-Mean is calculated based on (8). The G-Mean ranges from 0 to 1, where 0 indicates that the classifier is unable to correctly classify any instance from either class, and 1 indicates that the classifier can perfectly classify all instances from both classes.

$$G - \text{Mean} = \sqrt{\text{TPR} \times \text{TNR}} \quad (8)$$

Cohen's Kappa is used to measure the level of agreement between two classifiers that identify the instances of mutually exclusive categories. It is calculated using (9). Cohen's Kappa ranges from 0 and 1, with 0 indicating random agreement between the two classifiers and 1 indicating perfect agreement.

$$k = \frac{p_o - p_e}{1 - p_e} \quad (9)$$

where:

k Cohen's Kappa;

p_o Relative observed agreement among model and test data;

p_e Hypothetical probability of chance agreement.

To evaluate the classifiers, we utilize the Python implementation of these metrics available in the sklearn.metrics library.

4. Results

In order to construct a data set to train ML models for fraud detection, we processed the logs available in the Campus Virtual Moodle platform, as depicted in Algorithm 1. Data privacy is ensured by encrypting all the user-specific information from the logs. The information found in the logs is mapped to describe each student's behavior in relation to

a certain exam. Data labeling is performed by identifying the cheaters. Table 7 presents the fraud ratios observed across two teaching methods during the study period: Face-to-Face (Face2Face) and Online. Table 8 provides the incidence of fraud for each course per academic year. Moreover, it illustrates the evolution in time of the fraud phenomenon among students. Further filtering of the data set was performed to eliminate duplicate records and students who withdrew from the course. Cheaters are represented by those who break the fixed rules.

Table 7. Fraud statistics per class format.

Class Format	Fraud Cases	Exam Attempts	Fraud (%)
Face2Face	40	1321	3.02
Online	181	2465	7.34
Total	221	3786	5.83

Table 8. Fraud statistics per course.

Course	Class Format	Academic Year	Fraud (%)
2019–2020			
POO	Face2Face	2019–2020	0.00
BD	Online	2019–2020	4.32
LP2	Online	2019–2020	7.20
2020–2021			
POO	Online	2020–2021	3.85
BD	Online	2020–2021	9.28
LP2	Online	2020–2021	9.63
2021–2022			
POO	Online	2021–2022	9.05
BD	Face2Face	2021–2022	7.48
LP2	Face2Face	2021–2022	1.83

After labeling, a notable difference was observed between the honest students and cheaters. Only 5.83% of students were identified as cheaters. Imbalance in the data set is a prevalent issue in fraud detection problems.

For handling the class imbalance problem, we tried out 11 resampling strategies, as described in Table 6, ranging from traditional oversampling methods to hybrid solutions. These methods include the use of the SMOTE algorithm followed by Tomek Links removal to avoid overfitting in the ADASYN algorithm, which is designed to handle challenging boundary samples. Moreover, hybrid solutions consisting of partial oversampling to obtain a ratio of 0.2 in the minority class, followed by undersampling using the Random Undersampling (RUS) method or the NearMiss method, were also implemented.

We proposed 12 approaches for the ML data set based on the template depicted in Figure 1. The 12 distinct data sets were derived by varying the attributes' computation interval, using either standard cumulative metrics or temporal windows of 1, 3, 5, or 7 days, as well as selectively incorporating auxiliary information such as exam attempts and grade book metadata, as seen in Figure 2. Comprehensive attribute descriptions and a summary of experimental data sets are provided in Table 4 and Table 5, respectively.

In all cases, data labeling is performed based on fixed rules. The final step in data set construction is scaling the attributes based on (1). The 12 data sets created serve as input for six ML classifiers.

We analyzed the performance of ML techniques in identifying academic fraud by conducting a comprehensive experimental study comprising 792 unique configurations. These experiments examined the performance of 12 derived data sets, 11 balancing protocols,

and six classification algorithms. Classifiers included in this study are LR, SVM, DT, GBM, AdaBoost, and RF.

The classifiers' performance was evaluated using a Repeated Stratified K-Fold cross-validation with five folds and three repetitions. For each fold, we split data into 80% for training and 20% for testing. This 5-fold process was repeated three times with different random data partitions, yielding a total of 15 independent evaluation runs. Given the highly imbalanced nature of the data set (where the minority fraud class represents 5.83% of entries), classifier performance was assessed using metrics capable of capturing minority class performance: precision, recall, and F1-Score, as well as balanced metrics such as: balanced accuracy, G-Mean, and Cohen's Kappa.

Table 9 summarizes the top-performing experiments across the following metrics: recall, F1-Score, G-Mean, and Cohen's Kappa. To compare performance, the result with the original data set is provided for each metric. The following section details our findings in response to our research questions, as introduced in Section 1.

Table 9. Performance summary of the top-five experimental configurations stratified by optimization metric. The table presents the mean values ($\pm$ standard deviation over 15 runs) for precision, recall, F1-Score, balanced accuracy, G-Mean, and Cohen's Kappa. Baseline results using the original unmodified data set are included for reference within each category.

Experiment	Metrics					
	Precision	Recall	F1-Score	Balanced Accuracy	G-Mean	Cohen's Kappa
Best Recall						
1 d_ADASYN_clean_nearmiss_RF	0.063 $\pm$ 0.004	0.757 $\pm$ 0.056	0.116 $\pm$ 0.008	0.528 $\pm$ 0.026	0.475 $\pm$ 0.018	0.009 $\pm$ 0.008
1 d_SMOTE_clean_nearmiss_GBC	0.064 $\pm$ 0.005	0.747 $\pm$ 0.056	0.118 $\pm$ 0.009	0.534 $\pm$ 0.029	0.489 $\pm$ 0.023	0.011 $\pm$ 0.010
1 d_ADASYN_clean_nearmiss_GBC	0.063 $\pm$ 0.006	0.745 $\pm$ 0.072	0.117 $\pm$ 0.011	0.530 $\pm$ 0.037	0.484 $\pm$ 0.028	0.010 $\pm$ 0.012
7 d_ADASYN_clean_nearmiss_RF	0.062 $\pm$ 0.005	0.745 $\pm$ 0.072	0.115 $\pm$ 0.010	0.526 $\pm$ 0.034	0.477 $\pm$ 0.025	0.008 $\pm$ 0.011
5 d_ADASYN_clean_nearmiss_RF	0.062 $\pm$ 0.005	0.744 $\pm$ 0.070	0.115 $\pm$ 0.009	0.525 $\pm$ 0.031	0.476 $\pm$ 0.020	0.008 $\pm$ 0.010
original_None_SVC	0.075 $\pm$ 0.008	0.617 $\pm$ 0.078	0.134 $\pm$ 0.014	0.573 $\pm$ 0.034	0.569 $\pm$ 0.034	0.033 $\pm$ 0.016
Best G-Mean						
1 d + extra_ADASYN_under_AdaB	0.09 $\pm$ 0.01	0.584 $\pm$ 0.068	0.156 $\pm$ 0.017	0.609 $\pm$ 0.034	0.607 $\pm$ 0.035	0.061 $\pm$ 0.019
1 d + extra_SMOTE_under_AdaB	0.091 $\pm$ 0.009	0.567 $\pm$ 0.052	0.157 $\pm$ 0.015	0.608 $\pm$ 0.028	0.606 $\pm$ 0.030	0.063 $\pm$ 0.017
1 d + extra_ADASYN_clean_under_AdaB	0.09 $\pm$ 0.012	0.576 $\pm$ 0.085	0.155 $\pm$ 0.021	0.607 $\pm$ 0.040	0.604 $\pm$ 0.043	0.06 $\pm$ 0.023
5 d + extra_ADASYN_under_AdaB	0.09 $\pm$ 0.011	0.567 $\pm$ 0.084	0.156 $\pm$ 0.019	0.606 $\pm$ 0.036	0.602 $\pm$ 0.040	0.061 $\pm$ 0.021
3 d + extra_SMOTE_clean_under_AdaB	0.089 $\pm$ 0.012	0.573 $\pm$ 0.091	0.153 $\pm$ 0.020	0.604 $\pm$ 0.041	0.601 $\pm$ 0.044	0.058 $\pm$ 0.022
original_None_SVC	0.075 $\pm$ 0.008	0.617 $\pm$ 0.078	0.134 $\pm$ 0.014	0.573 $\pm$ 0.034	0.569 $\pm$ 0.034	0.033 $\pm$ 0.016
Best F1-Score						
original + extra_ADASYN_clean_over_RF	0.218 $\pm$ 0.042	0.189 $\pm$ 0.037	0.202 $\pm$ 0.038	0.573 $\pm$ 0.019	0.423 $\pm$ 0.041	0.156 $\pm$ 0.040
original + extra_SMOTE_clean_over_RF	0.223 $\pm$ 0.041	0.184 $\pm$ 0.033	0.201 $\pm$ 0.033	0.572 $\pm$ 0.016	0.419 $\pm$ 0.037	0.156 $\pm$ 0.035
original + extra_ADASYN_over_RF	0.223 $\pm$ 0.036	0.177 $\pm$ 0.037	0.196 $\pm$ 0.035	0.569 $\pm$ 0.018	0.410 $\pm$ 0.041	0.153 $\pm$ 0.036
original + extra_SMOTE_over_RF	0.226 $\pm$ 0.053	0.169 $\pm$ 0.047	0.192 $\pm$ 0.048	0.566 $\pm$ 0.023	0.400 $\pm$ 0.056	0.150 $\pm$ 0.050
original_SMOTE_clean_over_RF	0.169 $\pm$ 0.045	0.196 $\pm$ 0.045	0.180 $\pm$ 0.043	0.567 $\pm$ 0.024	0.426 $\pm$ 0.053	0.125 $\pm$ 0.047
original_None_SVC	0.075 $\pm$ 0.008	0.617 $\pm$ 0.078	0.134 $\pm$ 0.014	0.573 $\pm$ 0.034	0.569 $\pm$ 0.034	0.033 $\pm$ 0.016
Best Kappa						
original + extra_ADASYN_clean_over_RF	0.218 $\pm$ 0.042	0.189 $\pm$ 0.037	0.202 $\pm$ 0.038	0.573 $\pm$ 0.019	0.423 $\pm$ 0.041	0.156 $\pm$ 0.040
original + extra_SMOTE_clean_over_RF	0.223 $\pm$ 0.041	0.184 $\pm$ 0.033	0.201 $\pm$ 0.033	0.572 $\pm$ 0.016	0.419 $\pm$ 0.037	0.156 $\pm$ 0.035
original + extra_ADASYN_over_RF	0.223 $\pm$ 0.036	0.177 $\pm$ 0.037	0.196 $\pm$ 0.035	0.569 $\pm$ 0.018	0.410 $\pm$ 0.041	0.153 $\pm$ 0.036
original + extra_SMOTE_over_RF	0.226 $\pm$ 0.053	0.169 $\pm$ 0.047	0.192 $\pm$ 0.048	0.566 $\pm$ 0.023	0.400 $\pm$ 0.056	0.150 $\pm$ 0.050
original_SMOTE_clean_over_RF	0.169 $\pm$ 0.045	0.196 $\pm$ 0.045	0.180 $\pm$ 0.043	0.567 $\pm$ 0.024	0.426 $\pm$ 0.053	0.125 $\pm$ 0.047
original_None_DTC	0.118 $\pm$ 0.049	0.122 $\pm$ 0.058	0.119 $\pm$ 0.052	0.533 $\pm$ 0.029	0.33 $\pm$ 0.081	0.064 $\pm$ 0.054

Bold values highlight the primary metric used to evaluate and rank the configurations within each respective section (e.g., Recall values are emphasized under the Best Recall category).

RQ1. What is the impact of temporal window size on the discriminative power of features extracted from Moodle activity logs for academic fraud detection?

Our experiment results indicate that narrowing the temporal window size improves the fraud detection rate. The highest recall was achieved by the RF classifier using the one-day temporal window data set (1 d) combined with a hybrid balancing strategy: ADASYN oversampling (ratio 0.2), Tomek Links cleaning, and NearMiss undersampling. This configuration yielded a recall of 0.757 ± 0.056 . Furthermore, the highest G-Mean (0.607 ± 0.035) was also achieved by AdaBoost classifier using the one-day window paired with auxiliary information and a hybrid pipeline of ADASYN oversampling and Random Undersampling. The best results for the baseline unmodified data set were obtained using SVC with a recall value of 0.617 ± 0.078 and a G-Mean value of 0.569 ± 0.034 .

RQ2. Which combination of resampling strategy and classification algorithm yields the best trade-off between fraud detection rate and precision in a highly imbalanced educational data set?

When prioritizing the detection rate, hybrid architectures yielded the best results. The top five results for recall were achieved using hybrid methods involving NearMiss undersampling, Tomek Links for eliminating outliers, and either ADASYN or SMOTE for oversampling. While these models achieved high recall ($\approx 75\%$), they exhibited very low precision ($\approx 6\%$), indicating a high rate of false positives. To optimize the trade-off between detection rate and precision, we evaluated the F1-Score and Kappa metrics. The RF classifier performed best in terms of concurrency of F1-Score and Kappa when working with the original+extra data set, which was balanced using ADASYN and Tomek Links. This resulted in the highest F1-Score of 0.202 ± 0.038 and Kappa of 0.156 ± 0.040 . The SMOTE equivalent, using Tomek Links and RF on the same data set, performed equally well, with an F1-Score of 0.201 ± 0.033 and Kappa of 0.156 ± 0.035 . The best performance in the baseline data set, without any modification, was less, with the SVC model having an F1-Score of 0.134 ± 0.014 and the DT classifier obtaining the maximum Kappa of 0.064 ± 0.054 .

RQ3. How does the inclusion of auxiliary student metadata (e.g., grade book information) influence the performance of machine learning models in detecting academic fraud compared to using log-derived features alone?

Integrating student information and grade book data alongside log-derived features improved the classifiers' overall performance and stability. When assessing the models using the G-Mean metric, the best result was given by the AdaBoost classifier. The best results for F1-Score and Kappa were achieved by the RF classifier. Notably, all five best-performing experiments when measuring G-Mean, F1-Score, and Kappa utilized data sets enriched with auxiliary information (+extra). The improvements over the baseline version are represented by an increase of 6.67% in G-Mean, 50.7% in F1-Score, and 143% in Kappa.

5. Discussions

The educational sector requires a fair and balanced environment for all the involved stakeholders, such as students and teachers. For the maintenance of equity in the evaluation process, there should be accurate and sound assessment techniques. As hypothesized in our methodology, the shift to online learning environments with limited proctoring resulted in a higher incidence of academic fraud, more than doubling from 3.02% in face-to-face settings to 7.34% in online classes, as shown in Table 7. This trend is confirmed by the analysis of fraud rates across courses and academic years in Table 8. The instances of academic fraud increased as a result of the start of online learning and continued to be high throughout the period of online learning, peaking at a rate of about 10% before declining as face-to-face learning resumed in late 2021. The results indicate that the lack of physical

surveillance during online exams contributed greatly to the occurrence of the misconduct. There thus exists a need for the development of an automated system that ensures academic integrity. The problem posed by the need for an automated system for academic integrity detection is tackled by the research, which investigates the following research questions: The impact of temporal window size on the discriminative power of Moodle log features (RQ1), the optimal combination of resampling strategy and classifier to balance detection rate and precision (RQ2), and the effect of adding auxiliary student metadata on detection performance (RQ3). The results show that while there may be some similarities between the detection of academic and financial fraud, academic fraud detection poses distinct challenges regarding feature engineering and class imbalance.

A contribution of this study is the creation of data sets derived from Moodle activity logs. Our results from the G-Mean, F1-Score, and Cohen's Kappa analysis showed that the inclusion of auxiliary data (students' grade book attributes) generated a better classification performance. This aligns with research in academic fraud detection. Kamalov et al. [29] demonstrated that detecting cheating requires analyzing differences between a student's performance (grades) during the course and their final exam score, treating fraud as an outlier in student's performance. Similarly, Hu et al. [18] utilized gaps between practice exercises and final exam scores to flag potential cheaters. Our findings support this, confirming that "grade book" attributes, specifically grades obtained prior to the exam, are relevant features for academic fraud detection.

Furthermore, our use of temporal windows to capture recent behavior is similar to the sliding window strategies used in financial fraud detection by Dornadula and Geetha [38]. Similarly, in the educational domain, Sangalli et al. [6] focused on granular interaction metrics, such as co-occurring submission times, to detect fraud. By translating the concept of the sliding window to the educational domain, where recent financial transactions are analogous to recent study activities, we demonstrated that temporal granularity (e.g., the 1 d data set) is more effective for fraud detection than semester-long aggregates when prioritizing recall or G-Mean. Notably, all top-five experiments yielding the best recall and G-Mean utilized temporal windows, achieving a maximum recall value of 0.757 ± 0.056 with the 1 d_ADASYN_clean_nearmiss_RF configuration and a maximum G-Mean value of 0.607 ± 0.0035 with the 1 d + extra_ADASYN_under_AdaB.

While temporal windows proved decisive for maximizing detection rates (recall and G-Mean), our analysis of F1-Score and Cohen's Kappa reveals a divergent trend. The highest stability, indicated by the top F1-Score (0.202 ± 0.038) and Cohen's Kappa (0.156 ± 0.040), was achieved using the cumulative (original + extra) data set rather than the segmented temporal windows. This distinction highlights the specific roles of different feature types: short-term temporal windows are sensitive to bursts of anomalous activity, making them suitable for flagging potential fraud (high recall). However, this sensitivity often comes at the cost of precision, as it generates more false positives. The semester-long cumulative metrics provide the necessary historical context to differentiate between a cheater and a hard-working student, effectively filtering out false alarms. Nevertheless, this stability induces a substantial trade-off: recall drops severely from 0.757 ± 0.056 to 0.189 ± 0.037 to achieve the best values for F1-Score and Cohen's Kappa.

Regarding the capability of ML models to detect fraud, our extensive experimental evaluation identified RF and AdaBoost as the superior classifiers. Moreover, RF accounted for 13 out of the 20 top performing results in our study. This finding is strongly supported by the existing literature, where RF has demonstrated efficacy in detecting financial fraud [1], medical fraud [13], and academic fraud [18]. However, a direct comparison of performance metrics reveals the distinct difficulty of this domain. While Sangalli et al. [6] reported accuracy exceeding 95% for specific collusion types, our optimal models achieved a recall

of approximately 75% and a Cohen's Kappa of 0.16. This disparity may stem from the definition of fraud; whereas Sangalli et al. targeted specific, detectable patterns, such as multiple account cheating, our study aimed to detect a broader range of anomalous behaviors, which are inherently more subtle. This reflects the reality of the evaluated data set, where students did not adhere to a specific cheating protocol, requiring a generalized anomaly detection strategy capable of identifying diverse forms of academic misconduct.

Our results confirm that standard, unmodified data sets are insufficient for detecting fraud, yielding a baseline Kappa of only 0.06. The use of hybrid resampling architectures, specifically combining oversampling (SMOTE/ADASYN) with cleaning (Tomek Links) or undersampling (NearMiss), were the balancing methods that showed the best results. We observed a distinct trade-off: hybrid undersampling using NearMiss maximized the recall (identifying more fraud instances), while oversampling with SMOTE or ADASYN paired with Tomek Links cleaning optimized the precision and F1-Score. These findings are similar to those of Garg and Goel [8], who utilized feature engineering and clustering to distinguish cheating patterns but noted the difficulty in establishing a clear boundary between “intense study” and “cheating”.

Considering the experiments that yielded high recall, approximately 75% at the expense of precision, these prediction results include a wide range of students who are possibly at-risk, including actual cheaters and many false alarms. Therefore, these results should not be considered proof of guilt but are more appropriately used for optimization in the proctoring phase. Instructors can use the risk assessment tool for proactive and non-intrusive intervention techniques. Such techniques may involve assigning the identified students high-visibility seating, giving the students priority for “random” checks, and improved monitoring through frequent walkthroughs and pinned video monitoring. Such measures will have the overall effect of increasing the chances of detecting actual cheating and improving the efficiency of human monitoring, while appearing as standard administrative procedures to innocent students.

A limitation of this study is the process of creating the data sets. The twelve data sets were created based on logs provided from three courses over several years from Campus Virtual. Although the extracted information is accurate in its current context, the algorithm for extraction is necessarily context-dependent. Therefore, when used for log data from a different course, there may be issues in obtaining a valid data set. Since there is no standardized structure that a Moodle course must follow, there exists the possibility that a custom algorithm may be required to extract information from the logs of another course. Future research can focus on making use of data that is retrieved from other courses and years. Given the absence of a standardized course structure, it is necessary to develop an extraction method that is not reliant on specific course structures. Alternatively, we need to improve the current method's capabilities to enable extraction from diverse course structures.

Another limitation may arise from the way we label the data set. We have labeled the data set with high precision, 100%, but the recall is uncertain. We label cheaters (fraud) as those students who break the fixed rules. This gives certainty that those labeled as cheaters have indeed cheated on the exam. Nonetheless, this method may leave out cheaters that performed fraudulent actions that are not covered by the fixed rules. For example, a student can download the course material and access it locally during an exam, a student can share answers with colleagues who took the exam earlier, or a student can look at a neighbor's screen. The fixed rules do not cover these actions that do not leave marks on the logs, thereby preventing the student from being labeled as a cheater. The moderate performance of the ML models may be partially explained by the label noise. The “honest” class contains an unknown proportion of undetected cheaters. A possible solution to this limitation that

can be explored in future work is to use unsupervised learning techniques for labeling, as done by Sangalli et al. [6].

While our hybrid models improved detection capabilities, the high false positive rate, indicated by low precision in the high recall experiments, remains a limitation. As noted by Carcillo et al. [49] in the financial domain, augmenting data sets with too many derived scores can lead to variance issues. Future work could address this by integrating unsupervised learning techniques, similar to the clustering methods used by Garg and Goel, to identify different types of legitimate study patterns (e.g., “cramming” vs. “consistent study”). By distinguishing these diverse normal behaviors from actual anomalies, we could reduce the noise that leads to the high number of false alarms.

Any intervention must be non-intrusive and framed as standard procedure to avoid creating a hostile environment for innocent students.

In summary, the contributions of this paper are:

1. The framework presented in Algorithm 1 is used for constructing data sets from the available Moodle activity logs. We applied this framework to create 12 data sets based on the logs of three courses over three academic years. The distinction between the data sets is based on the temporal windows used and the inclusion of auxiliary information from the students’ grade book. Other instructors can use this framework to create data sets based on their available logs, but they should treat it as a template and adapt it to their specific course structure.
2. The analysis of the temporal windows revealed that the usage of shorter temporal windows results in capturing a higher number of fraudulent activities. The highest recall of 0.757 ± 0.056 is achieved using the one-day window for the data set. The best G-Mean of 0.607 ± 0.035 was also achieved by a one-day window data set, but with the inclusion of auxiliary information. This suggests that short temporal windows are more efficient in detecting fraudulent activities, while the inclusion of auxiliary data improves the overall performance.
3. The evaluation of 11 resampling strategies over 12 data sets and six classifiers (792 experiments total) demonstrated that class imbalance can be better treated using hybrid methods. Oversampling followed by cleaning with Tomek Links and under-sampling with NearMiss showed the best recall values, approximately 75% for the top five results. Across all evaluated metrics, the best results were achieved with a hybrid resampling method.
4. Given the high false positive rate inherent in high-recall configurations, model outputs should not be treated as evidence of guilt. An instructor can best use these as screening tools. The students flagged by the model can be prioritized for “random checks”, they can be assigned to high-visibility seats, and they can be monitored more closely. These measures can increase the chances of detecting actual fraud and improving the efficiency of human proctoring, at the same time appearing as standard procedures to honest students.

6. Conclusions

Our study explored how to use activity logs to construct a data set that ML models can be trained on for academic fraud detection. We established a framework for creating data sets based on the Moodle logs and student grade books. This process produces 12 distinct data sets by varying the temporal windows used in feature extraction and the inclusion of student grade book information. We used 11 resampling techniques: oversampling with SMOTE and ADASYN; hybrid resampling based on oversampling followed by undersampling with RUS and NearMiss; hybrid resampling with cleaning based on Tomek Links; and no resampling to serve as a benchmark. We evaluated six ML techniques: LR, DT,

SVM, GBM, AdaBoost, and RF. In this study, we developed a total of 792 experiments by testing every combination of data set, resampling technique, and ML technique. Furthermore, we found that maximizing recall is more advantageous when features are computed using smaller temporal windows rather than standard semester-long windows (RQ1). The highest stability, indicated by the F1-Score and Cohen's Kappa was achieved using the auxiliary grade book information (RQ3). While RF and AdaBoost outperformed the other techniques using hybrid methods for resampling involving NearMiss, Tomek Links, and either ADASYN or SMOTE (RQ2), the overall performance of the evaluated models remains unsatisfactory as none of them provided acceptable values for F1-Score, Cohen's Kappa, or G-Mean. A possible explanation can be related to the ground truth. We defined rules that define with certainty fraudulent activities but cannot cover every possible fraud scenario. This labeling method ensures that every entity labeled as a "cheater" committed fraud, but does not guarantee that all "honest" labeled students did not cheat. Another aspect that we extracted based on the results is that hybrid undersampling maximized recall ($\approx 75\%$), while hybrid oversampling optimized metrics such as F1-Score and Cohen's Kappa.

Our findings suggest that the evaluated models cannot definitively prove guilt, but high-recall models can function as screening tools. Instructors can use these predictions to focus their attention on the sensitive cases. They can take actions such as moving flagged students to better visibility seats, prioritizing them for "random" checks, or closely observing them, while leaving honest students undisturbed. As future work, we will focus on reducing false positives by using unsupervised clustering to better distinguish diverse, legitimate study patterns from misconduct, and validating these approaches on data sets that originate from different institutions.

Author Contributions: Conceptualization, A.-N.V. and B.D.; methodology, M.O. and B.D.; software, A.-N.V., B.D. and M.B.; validation, A.-N.V., M.B. and B.D.; formal analysis, M.O.; investigation, A.-N.V.; resources, M.B.; data curation, B.D.; writing—original draft preparation, A.-N.V. and B.D.; writing—review and editing, M.O. and M.B.; visualization, B.D.; supervision, M.O.; project administration, M.B.; funding acquisition, A.-N.V. and B.D. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: Our experimental framework created 12 distinct data sets by changing the temporal windows and the inclusion of student grade book information. Despite the fact that we encoded students' personal information using MD5 hashing, the data sets cannot be made publicly available due to privacy concerns. The combination of course, academic year, and activities can be used to identify students, as there is a limited number of students in each course and academic year, thus creating the potential to re-identify students.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

ML	Machine Learning
SMOTE	Synthetic Minority Oversampling Technique
ADASYN	Adaptive Synthetic Sampling
RUS	Random Undersampling
DT	Decision Tree
LR	Logistic Regression
SVM	Support Vector Machine
GBM	Gradient Boosting Machine
RF	Random Forest
AdaBoost	Adaptive Boosting

F1	F1-Score
G-Mean	Geometric Mean
Kappa	Cohen's Kappa

References

1. Yao, J.; Zhang, J.; Wang, L. A financial statement fraud detection model based on hybrid data mining methods. In *Proceedings of the 2018 International Conference on Artificial Intelligence and Big Data, ICAIBD 2018*; IEEE: New York, NY, USA, 2018; pp. 57–61. <https://doi.org/10.1109/ICAIBD.2018.8396167>.
2. Baesens, B.; Höppner, S.; Verdonck, T. Data engineering for fraud detection. *Decis. Support Syst.* **2021**, *150*, 113492. <https://doi.org/10.1016/j.dss.2021.113492>.
3. Thabtah, F.; Hammoud, S.; Kamalov, F.; Gonsalves, A. Data imbalance in classification: Experimental evaluation. *Inf. Sci.* **2020**, *513*, 429–441. <https://doi.org/10.1016/j.ins.2019.11.004>.
4. Ranger, J.; Schmidt, N.; Wolgast, A. The Detection of Cheating on E-Exams in Higher Education—The Performance of Several Old and Some New Indicators. *Front. Psychol.* **2020**, *11*, 568825. <https://doi.org/10.3389/fpsyg.2020.568825>.
5. Alexandron, G.; Berg, A.; Ruipérez-Valiente, J.A. A General Purpose Anomaly-Based Method for Detecting Cheaters in Online Courses. *IEEE Trans. Learn. Technol.* **2024**, *17*, 1–11. <https://doi.org/10.1109/TLT.2023.3297132>.
6. Sangalli, V.A.; Martínez-Muñoz, G.; Cañabate, E.P. Identifying Cheating Users in Online Courses. In *Proceedings of the 2020 IEEE Global Engineering Education Conference (EDUCON)*; IEEE: New York, NY, USA, 2020; pp. 1168–1175. <https://doi.org/10.1109/EDUCON45650.2020.9125252>.
7. Marksteiner, T.; Reinhard, M.A.; Dickhäuser, O.; Sporer, S.L. How Do Teachers Perceive Cheating Students? Beliefs about Cues to Deception and Detection Accuracy in the Educational Field. *Eur. J. Psychol. Educ.* **2012**, *27*, 329–350. <https://doi.org/10.1007/s10212-011-0074-5>.
8. Garg, M.; Goel, A. Preserving Integrity in Online Assessment Using Feature Engineering and Machine Learning. *Expert Syst. Appl.* **2023**, *225*, 120111. <https://doi.org/10.1016/j.eswa.2023.120111>.
9. Herland, M.; Khoshgoftaar, T.; Bauder, R. Big Data fraud detection using multiple medicare data sources. *J. Big Data* **2018**, *5*, 29. <https://doi.org/10.1186/s40537-018-0138-3>.
10. Velasco, R.; Carpanese, I.; Interian, R.; Paulo Neto, O.; Ribeiro, C. A decision support system for fraud detection in public procurement. *Int. Trans. Oper. Res.* **2021**, *28*, 27–47. <https://doi.org/10.1111/itor.12811>.
11. Itoo, F.; Meenakshi, S.; Singh, S. Comparison and analysis of logistic regression, Naïve Bayes and KNN machine learning algorithms for credit card fraud detection. *Int. J. Inf. Technol.* **2021**, *13*, 1503–1511. <https://doi.org/10.1007/s41870-020-00430-y>.
12. Varmedja, D.; Karanovic, M.; Sladojevic, S.; Arsenovic, M.; Anderla, A. Credit Card Fraud Detection - Machine Learning methods. In *Proceedings of the 2019 18th International Symposium INFOTEH-JAHORINA, INFOTEH 2019—Proceedings*; IEEE: New York, NY, USA, 2019; pp. 1–5. <https://doi.org/10.1109/INFOTEH.2019.8717766>.
13. Bauder, R.; Khoshgoftaar, T. The effects of varying class distribution on learner behavior for medicare fraud detection with imbalanced big data. *Health Inf. Sci. Syst.* **2018**, *6*, 9. <https://doi.org/10.1007/s13755-018-0051-3>.
14. Pambudi, B.N.; Hidayah, I.; Fauziati, S. Improving Money Laundering Detection Using Optimized Support Vector Machine. In *Proceedings of the 2019 2nd International Seminar on Research of Information Technology and Intelligent Systems, ISRITI 2019*; IEEE: New York, NY, USA, 2019; pp. 273–278. <https://doi.org/10.1109/ISRITI48646.2019.9034655>.
15. Amarbayasgalan, T.; Pham, V.; Theera-Umporn, N.; Ryu, K. Unsupervised anomaly detection approach for time-series in multi-domains using deep reconstruction error. *Symmetry* **2020**, *12*, 1251. <https://doi.org/10.3390/SYM12081251>.
16. Amarbayasgalan, T.; Jargalsaikhan, B.; Ryu, K. Unsupervised novelty detection using deep autoencoders with density based clustering. *Appl. Sci.* **2018**, *8*, 1468. <https://doi.org/10.3390/app8091468>.
17. Haytom, M.; Rosenberger, C.; Charrier, C.; Zhu, C.; Regnier, C. Identity verification and fraud detection during online exams with a privacy compliant biometric system. In *Proceedings of the 17th International Joint Conference on E-Business and Telecommunications (SECURITY)—Volume 1*; SciTePress: Setúbal, Portugal, 2020; Volume 3, pp. 451–458. <https://doi.org/10.5220/0009874104510458>.
18. Hu, H.; Li, Z.; Wang, Z. Test Cheating Detection Method Based on Random Forest. In *Proceedings of the 2021 3rd International Conference on Computer Science and Technologies in Education (CSTE)*; IEEE: New York, NY, USA, 2021; pp. 47–52. <https://doi.org/10.1109/CSTE53634.2021.00017>.
19. Wang, D.; Lin, J.; Cui, P.; Jia, Q.; Wang, Z.; Fang, Y.; Yu, Q.; Zhou, J.; Yang, S.; Qi, Y. A Semi-Supervised Graph Attentive Network for Financial Fraud Detection. In *Proceedings of the 2019 19th IEEE International Conference on Data Mining (ICDM 2019)*; IEEE: New York, NY, USA, 2019; pp. 598–607. <https://doi.org/10.1109/ICDM.2019.00070>.
20. Roy, A.; Sun, J.; Mahoney, R.; Alonzi, L.; Adams, S.; Beling, P. Deep learning detecting fraud in credit card transactions. In *Proceedings of the 2018 Systems and Information Engineering Design Symposium, SIEDS 2018*; IEEE: New York, NY, USA, 2018; pp. 129–134. <https://doi.org/10.1109/SIEDS.2018.8374722>.

21. Ji, S.J.; Zhang, Q.; Li, J.; Chiu, D.; Xu, S.; Yi, L.; Gong, M. A burst-based unsupervised method for detecting review spammer groups. *Inf. Sci.* **2020**, *536*, 454–469. <https://doi.org/10.1016/j.ins.2020.05.084>.
22. Wang, Z.; Gu, S.; Zhao, X.; Xu, X. Graph-based review spammer group detection. *Knowl. Inf. Syst.* **2018**, *55*, 571–597. <https://doi.org/10.1007/s10115-017-1068-7>.
23. Liu, Z.; Dou, Y.; Yu, P.S.; Deng, Y.; Peng, H. Alleviating the inconsistency problem of applying graph neural network to fraud detection. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*; ACM: New York, NY, USA, 2020; pp. 1569–1572.
24. Dou, Y.; Liu, Z.; Sun, L.; Deng, Y.; Peng, H.; Yu, P.S. Enhancing graph neural network-based fraud detectors against camouflaged fraudsters. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*; ACM: New York, NY, USA, 2020; pp. 315–324.
25. Craja, P.; Kim, A.; Lessmann, S. Deep learning for detecting financial statement fraud. *Decis. Support Syst.* **2020**, *139*, 113421.
26. Johnson, J.; Khoshgoftaar, T. Medicare fraud detection using neural networks. *J. Big Data* **2019**, *6*, 63. <https://doi.org/10.1186/s40537-019-0225-0>.
27. Subash, A.; Song, I. Real-Time Behavioral Biometric Information Security System for Assessment Fraud Detection. In *Proceedings of the 2021 IEEE International Conference on Computing, ICOCO 2021*; IEEE: New York, NY, USA, 2021; pp. 186–191. <https://doi.org/10.1109/ICOCO53166.2021.9673568>.
28. El Kohli, S.; Jannaj, Y.; Maanan, M.; Rhinane, H. Deep Learning: New Approach for Detecting Scholar Exams Fraud. *Int. Arch. Photogramm. Remote Sens. Spat. Inf. Sci.* **2022**, *XLVI-4-W3-2021*, 103–107. <https://doi.org/10.5194/isprs-archives-XLVI-4-W3-2021-103-2022>.
29. Kamalov, F.; Sulieman, H.; Calonge, D.S. Machine Learning Based Approach to Exam Cheating Detection. *PLoS ONE* **2021**, *16*, e0254340. <https://doi.org/10.1371/journal.pone.0254340>.
30. Nami, S.; Shajari, M. Cost-sensitive payment card fraud detection based on dynamic random forest and k-nearest neighbors. *Expert Syst. Appl.* **2018**, *110*, 381–392. <https://doi.org/10.1016/j.eswa.2018.06.011>.
31. Randhawa, K.; Loo, C.K.; Seera, M.; Lim, C.P.; Nandi, A.K. Credit Card Fraud Detection Using AdaBoost and Majority Voting. *IEEE Access* **2018**, *6*, 14277–14284. <https://doi.org/10.1109/ACCESS.2018.2806420>.
32. Dong, M.; Yao, L.; Wang, X.; Benatallah, B.; Huang, C.; Ning, X. Opinion fraud detection via neural autoencoder decision forest. *Pattern Recognit. Lett.* **2020**, *132*, 21–29.
33. Makki, S.; Assaghir, Z.; Taher, Y.; Haque, R.; Hacid, M.S.; Zeineddine, H. An Experimental Study With Imbalanced Classification Approaches for Credit Card Fraud Detection. *IEEE Access* **2019**, *7*, 93010–93022. <https://doi.org/10.1109/ACCESS.2019.2927266>.
34. Sokout, H.; Purnama, F.; Mustafazada, A.N.; Usagawa, T. Identifying Potential Cheaters by Tracking Their Behaviors through Mouse Activities. In *Proceedings of the 2020 IEEE International Conference on Teaching, Assessment, and Learning for Engineering (TALE)*; IEEE: New York, NY, USA, 2020; pp. 143–149. <https://doi.org/10.1109/TALE48869.2020.9368400>.
35. Dhankhad, S.; Mohammed, E.; Far, B. Supervised Machine Learning Algorithms for Credit Card Fraudulent Transaction Detection: A Comparative Study. In *Proceedings of the 2018 IEEE 19th International Conference on Information Reuse and Integration for Data Science, IRI 2018*; IEEE: New York, NY, USA, 2018; pp. 122–125. <https://doi.org/10.1109/IRI.2018.00025>.
36. Hussein, A.; Khairy, R.; Mohamed Najeeb, S.; Salim ALRikabi, H. Credit Card Fraud Detection Using Fuzzy Rough Nearest Neighbor and Sequential Minimal Optimization with Logistic Regression. *Int. J. Interact. Mob. Technol.* **2021**, *15*, 24–42. <https://doi.org/10.3991/ijim.v15i05.17173>.
37. Credit Card Fraud Detection. kaggle.com. 2024. Available online: <https://www.kaggle.com/datasets/mlg-ulb/creditcardfraud/> (accessed on 15 January 2026).
38. Dornadula, V.; Geetha, S. Credit Card Fraud Detection using Machine Learning Algorithms. *Procedia Comput. Sci.* **2019**, *165*, 631–641. <https://doi.org/10.1016/j.procs.2020.01.057>.
39. Robinson, C.; Taylor, C. Student voice as a contested practice: Power and participation in two student voice projects. *Improv. Sch.* **2013**, *16*, 32–46.
40. Giray, L. The Problem with False Positives: AI Detection Unfairly Accuses Scholars of AI Plagiarism. *Ser. Libr.* **2024**, *85*, 181–189. <https://doi.org/10.1080/0361526X.2024.2433256>.
41. Killourhy, K.S.; Maxion, R.A. Comparing anomaly-detection algorithms for keystroke dynamics. In *Proceedings of the 2009 IEEE/IFIP International Conference on Dependable Systems & Networks*; IEEE: New York, NY, USA, 2009; pp. 125–134.
42. Rtayli, N.; Enneya, N. Enhanced credit card fraud detection based on SVM-recursive feature elimination and hyper-parameters optimization. *J. Inf. Secur. Appl.* **2020**, *55*, 102596. <https://doi.org/10.1016/j.jisa.2020.102596>.
43. Yang, W.; Zhang, Y.; Ye, K.; Li, L.; Xu, C.Z. FFD: A federated learning based method for credit card fraud detection. In *Big Data—BigData 2019; Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*; Springer: Cham, Switzerland, 2019; Volume 11514, pp. 18–32. https://doi.org/10.1007/978-3-030-23551-2_2.
44. Moodle—Open-Source Learning Platform | Moodle.org. 2024. Available online: <https://moodle.org/> (accessed on 5 December 2025).

45. Campus Virtual UPT. cv.upt.ro. Available online: <https://cv.upt.ro/> (accessed on 8 March 2024).
46. He, H.; Bai, Y.; Garcia, E.A.; Li, S. ADASYN: Adaptive synthetic sampling approach for imbalanced learning. In *Proceedings of the 2008 IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence)*; IEEE: New York, NY, USA, 2008; pp. 1322–1328. <https://doi.org/10.1109/IJCNN.2008.4633969>.
47. Lemaître, G.; Nogueira, F.; Aridas, C.K. Imbalanced-learn: A Python Toolbox to Tackle the Curse of Imbalanced Datasets in Machine Learning. *J. Mach. Learn. Res.* **2017**, *18*, 1–5.
48. Taha, A.A.; Malebary, S.J. An Intelligent Approach to Credit Card Fraud Detection Using an Optimized Light Gradient Boosting Machine. *IEEE Access* **2020**, *8*, 25579–25587. <https://doi.org/10.1109/ACCESS.2020.2971354>.
49. Carcillo, F.; Le Borgne, Y.A.; Caelen, O.; Kessaci, Y.; Oblé, F.; Bontempi, G. Combining unsupervised and supervised learning in credit card fraud detection. *Inf. Sci.* **2021**, *557*, 317–331. <https://doi.org/10.1016/j.ins.2019.05.042>.

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.