International Journal of Mathematical, Engineering and Management Sciences
Vol. 11, No. 2, 850-895, 2026
https://doi.org/10.33889/IJMEMS.2026.11.2.036

SMOTE-AAE-RF: A Novel Approach to Identifying Financial Fraud in Imbalanced Data Circumstances

Satyendra Singh Rawat
Computer Science and Engineering Department,
Amity University, Gwalior, Madhya Pradesh, India.
Corresponding author: satyandra.rawat@s.amity.edu, satyandra2005@gmail.com

Vikas Thada
Computer Science and Engineering Department,
Amity University, Gwalior, Madhya Pradesh, India.
E-mail: vthada@gwa.amity.edu

Amit Kumar Mishra
Computer Science and Engineering Department,
Sagar Institute of Science and Technology, Bhopal, Madhya Pradesh, India.
E-mail: amitkumarmishra@sistec.ac.in

(Received on June 1, 2025; Revised on July 15, 2025 & September 19, 2025 & October 31, 2025 & December 15, 2025; Accepted on December 25, 2025)

Abstract
Nowadays, artificial intelligence (AI) and deep learning-based fraud detection methods are more powerful than conventional machine learning methods. However, the financial sector, including banks, insurance companies, the share market, and non-banking financial companies (NBFCs), also widely adopted these techniques in India. But these AI and deep learning techniques still face several challenges, including class-imbalance issues, class overlapping, noise, and high computational costs. In this paper, we have analyzed these challenges in data sets in terms of data complexity. Reducing the data complexity of the dataset will undoubtedly enhance the performance of the model. We also suggest a novel SMOTE-AAE-RF technique to detect fraud in the financial domain. This technique combines the Synthetic Minority Oversampling Technique (SMOTE), the Adversarial Autoencoder (AAE), and the Random Forest (RF). When the data set is skewed, we use the SMOTE technique to resample it. An adversarial autoencoder is used as a filter to filter out the samples that were generated by the SMOTE technique, and the RF classifier is used to detect whether the transactions are genuine or fraudulent. We analyzed the superiority of the suggested method on various performance measures, and the results indicate that it outperformed its competitor methods.

Keywords- Autoencoder, Data complexity, Fraud detection, Machine learning, SMOTE.

1. Introduction

The class imbalance problem, commonly observed in fraud datasets, is typified by the underrepresentation of a more substantial class (the fraudulent class) in comparison to genuine ones. Due to the potential for significant financial consequences, machine learning (ML) is crucial in detecting fraudulent situations. However, it faces several difficulties due to the wide variance of class sizes and the absence of accurate class labels (Kennedy et al., 2023). Therefore, learning from unevenly distributed data may lead to a trained model that is less effective and reliable. We refer to this characteristic as an imbalance or unbalance (de Vargas et al., 2023). Imbalanced data sets lead to overfitting of the model, which worsens performance (Mujahid et al., 2024). The vast majority of newly generated data in current data fields such as insurance claim fraud (Maiano et al., 2023), credit card fraud (Jemai et al., 2024), Medicare fraud (Bounab et al., 2024), etc., is skewed, and these types of data provide unique challenges for ML and classification due to their extreme class imbalance (Kennedy et al., 2024). The issue of class imbalance has long existed in credit card transactions because there are substantially fewer fraudulent transactions than non-fraudulent ones (Alharbi et al., 2024). This challenge directed the progress of several strategies to address class imbalance issues (Cemernek et al., 2024). Preprocessing methods like oversampling the minority class are frequently employed to address this problem, but they often produce samples that are unrealistic or overly generic (Du et al., 2024). Class overlap, or similar instances of data across classes, is one of the major issues. It is a more critical issue than the class imbalance (Adiputra & Wanchai, 2024). The paper provides a detailed theoretical study on the problem of class imbalance, showing how it increases pre-existing data difficulty characteristics like noise, small disjuncts, and class overlap, biasing the model in favor of the majority class. This study recognized the significance of understanding and investigating the related consequences of data difficulty and class imbalance issues (Carvalho et al., 2025). Overlapping areas affect most classifiers, making it difficult to distinguish between classes. The overlapped samples in the problem-data space have similar characteristics, creating an intricate border that makes it difficult to distinguish between class samples and degrades performance. How much the classifier is impacted is yet unknown, although the research community acknowledged that class overlapping concerns have an impact on the classifier's performance (Mahmood et al., 2025). Researchers have developed several oversampling methods to generate synthetic samples for the minority class by linear interpolation between a chosen minority sample and its k-nearest neighbors (k-NN). Nevertheless, *k-NN*-based techniques may encounter difficulties due to the curse of dimensionality (increasing the number of features makes data sparse), particularly when text oversampling is included (Nouas et al., 2025). Oversampling methods based on Generative Adversarial Networks (GANs) have increased recently due to deep learning advancements. By generating high-quality new samples and picking the distributed characteristics of minority samples during training, these methods have confirmed exceptional performance in addressing the class imbalance problem. However, when generating new samples, oversampling techniques based on GANs may have noise and class overlapping problems, as well as gradient vanishing, which leads to method failure (Zhou et al., 2025). The learning system may perform poorly as a result of this data imbalance and misclassify minority labels, which may result in inaccurate predictions and additional costs. Former oversampling methods, in contrast to the other classes, commonly ignore the distribution and total density of minority samples. Furthermore, the majority of oversampling techniques are less explanations (Kumbhar et al., 2026).

The data complexity measures for classification can evaluate the complexity in dividing the samples into their predicted classes (Lorena et al., 2019). The goal of complexity metrics is to explain the basic challenges of supervised data and tackle problems like overlap, density, and linearity that can reduce how well machine learning (ML) classifiers work. Recently, the sample perspective of complexity has also been addressed, while the advanced has mostly concentrated on the dataset perspective of difficulty, i.e., providing an estimate of the complexity of the entire dataset (Lancho et al., 2023). Difficulty metrics (data-level/global measures) and instance hardness metrics (instance-level/local measures) are multi-features that show how complicated the real data are for the high-class overlapping issue (Al Hosni & Starkey, 2023). Researchers have created and evaluated numerous imputation techniques over the years, primarily in optimal conditions. Imputation is a method in machine learning that substitutes predicted values for missing data. Surprisingly, no detailed research has been conducted on how effective imputation techniques are when the ideal conditions for data distribution and model assumptions are not fully met (Templ & Ulmer, 2024). This paper provides a survey of the literature on ML methods for detecting financial fraud (Aros et al., 2024). While ML and data mining approaches can provide various methods and algorithms for fraud detection and mitigation, they still need to enhance their ability to identify unidentified fraud trends and integrate large data processing mechanisms. This study forecasts fraud in financial transactions by using diverse deep learning (DL) methods, including LSTM, recurrent neural networks, and autoencoders (AEs). This paper gives a thorough look at better ways to find credit card fraud (CCF) using AI, ML, DL, and meta-heuristic optimization techniques. This study's primary conclusion calls for more research on AI techniques capable of detecting the most recent forms of fraud. This research examines classifier performance from the perspective of data difficulty. Furthermore, classifiers are classified into several realms of competence and incompetence (Eberlein et al., 2025).

This study has the following research objectives:
(i) Address the above-mentioned issues with large and complex data sets in terms of data complexity;
(ii) Perform a comparative study on the most popular data resampling techniques;
(iii) Propose a novel method for identifying financial fraud that blends ML and DL methods. This combination is more effective than ML-based strategies alone, as many researchers have shown;
(iv) Compare the experimental outcomes with advanced approaches;
(v) Outline limitations and future directions.

The paper is structured as follows.
Section 1: This section gives an introduction.
Section 2: This section reviews the relevant work.
Section 3: The methods and materials are discussed in this section.
Section 4: This section discusses performance measures.
Section 5: The suggested work is covered in this section.
Section 6: This section discusses the results of the trials, and
Section 7: The conclusion and future directions are given in this section.

2. Related Work

Graph-based anomaly detection methods are some of the most widely used to examine patterns of connectivity in the communication network and detect questionable activity. The motto of this study is to analyze current trends and identify the main issues that need serious investigation to surge the legitimacy of the approaches (Pourhabibi et al., 2020). It is imperative to have organizations that promise the integrity and security of credit card transactions. Using real-world skewed datasets created from European credit cardholders, this study has applied an ML-based system for CCF detection. The trial results confirmed that applying AdaBoost improves the effectiveness of the introduced method. The limitation of this method is that it was only tested and verified on a small number of datasets (Ileberi et al., 2021). Developing efficient fraud detection algorithms is essential for reducing these losses; however, it is challenging because most credit card datasets are extremely imbalanced, and traditional techniques cannot adapt to the changing spending habits of credit card users. The suggested LSTM-ensemble approach did better than the other approaches, attaining a recall and specificity of 0.996 and 0.998, respectively (Esenogho et al., 2022). The substantial growth in fraud requires high-transactional processing costs in banking. Consider using class weight-setting hyperparameters to regulate the relative importance of genuine and fraudulent cases. However, using Bayesian optimization and DL to fine-tune the hyperparameters, we get scores of 80% precision, 82% recall, 81% F1-score, 81% MCC, and 95% ROC-AUC (Hashemi et al., 2023). Researchers developed a similar method that uses a DL-based stacking ensemble with data resampling to successfully detecting CCF. Its sensitivity, specificity, and AUC values were 100%, 99.70%, and 100%, respectively (Mienye & Sun, 2023). Machine learning is essential for detecting fraud because it can have significant impacts on financial wealth. However, due to the disparity between classes and the lack of reliable class labeling, ML encounters several challenges. This study offers a successful unsupervised fraud detection method that makes use of a repeated cleaning process. In the CCF and Medicare fraud domains, this method finds a higher PR-AUC with relatively few iterations (Kennedy et al., 2023). This study has several benefits for future collaboration among financial professionals in the detection of fraud. Additionally, it can uncover fascinating details about the potential applications of metaverse technology in corporate financing. Fraud detection (FD) is only one of the ways that the metaverse, a quickly developing technology, could change our lives (Xu et al., 2024).

To maximize the classification of both fraudulent and non-fraudulent transactions, a sequential model based on DL is proposed, achieving a 0.995 accuracy, 0.972 precision, 0.985 recall, and 0.989 F1-score (Palivela et al., 2024). The standard graph neural networks may ignore or hide important fraud information because of the small number of fraudsters. In other words, it doesn't provide a clear picture of fraud characteristics. To solve these issues, this work suggests the tran-smote on graph (GTS) approach for fraud detection (Wen et al., 2024). The researchers (Alamri & Ykhlef, 2024) recommend a hybrid Tomeklinks BIRCH Clustering Borderline SMOTE (BCBSMOTE) sampling method to balance an extremely imbalanced dataset of credit card transactions, and it achieved better performance than the baseline sampling methods, with an F1 score of 0.852. Because of the skewed distribution of classes, it is difficult to identify fraudulent transactions. Preprocessing strategies, such as oversampling minority groups, frequently address class imbalance issues by producing incorrect or distorted samples. Based on the SMOTE and CGAN models, this study proposes an AE with probabilistic XGBoost to identify CCF. The approach is promising due to its higher accuracy, TPR (true-positive rate), TNR (true-negative rate), and Matthew's correlation coefficient (Du et al., 2024). To address concerns about data security and privacy, this paper proposes a federated learning approach for detecting CCF. Furthermore, the study proposes hybrid resampling techniques as a result to imbalanced class problems and to enhance classification performance. TensorFlow Federated (92.15%, 91.97%, 92.93%) is less accurate than the federated model on PyTorch-pysyft (93%, 92%, 90%) for Adam (Salam et al., 2024). With a PSO-optimized stacking ensemble model, SMOTE-ENN, AE, and TOPSIS-based feature selection, this study provides a scalable and dependable framework for CCF recognition that achieves 100% AUC, 99.95% accuracy, 99.93% precision, 99.97% recall, and a 99.95% F1 score (Gupta et al., 2024).

3. Materials and Methods

3.1 Data Complexity Measures

Several preprocessing methods have been used to improve the classifiers' performance to solve problems with imbalance, overlap, dimensionality, and separability. Therefore, we can calculate data complexity metrics to characterize data and provide an indication of problem difficulty. It identifies the data complexity circumstances that enable each classifier to perform better or worse. Missing values, concept drift, duplicate examples, and outliers in the data sets are additional problems that are not specifically examined by the data complexity metrics (Ho & Basu, 2002). Every machine learning model has distinct benefits and drawbacks based on the characteristics of the underlying data, and the data has a significant impact on the models' efficacy (Cavalcanti et al., 2012). In addition to identifying different areas of the classifiers' competency and incompetence, this study looks at the classifiers' performance from the perspective of data complexity. It can classify the following complexity measures into six major categories: network (Density, ClsCoef, Hubs), linearity (L1, L2, L3), dimensionality (T2, T3, T4), overlap (F1, F1.v, F2, F3, F4), neighborhood (N1, N2, N3, N4, T1, LSC), and balance (C1, C2) (Eberlein et al., 2025).

3.2 Data Preprocessing Techniques

Imbalanced learning frequently uses preprocessing and data sampling as common methods to address data science issues. They work in tandem with any learning method and are straightforward and easily adaptable. Data-level approaches involve changing the training examples to create a more balanced class distribution, allowing classifiers to work like traditional classification methods (Singh et al., 2022). Algorithmic-level processes aim to modify standard learning algorithms to better address the problems of class imbalance (Harliman & Uchida, 2018). Cost-sensitive approaches combine methods from both data and algorithm levels, taking into account that misclassifying samples from the smaller class is more expensive than from the larger class, and they aim to reduce those costly errors (Araf et al., 2024). There are some resampling approach publications in the specialist literature that examine the impact of altering the class distribution to address imbalanced datasets. The primary benefit of these methods is their independence from the underlying classifier. Three classes can be used to classify resampling methods (Batista et al., 2004):

- *Undersampling methods*, which remove samples of the majority class from the initial dataset to produce a subset. For example, ENN (Wilson, 1972), Tomek links (Tomek, 1976), OSS (Kubat, 1997), and the NCR (Laurikkala, 2001), etc.

- *Oversampling techniques*, which replicate some samples of the original dataset or generate new instances from preexisting ones, produce a superset of the original dataset. For example, SMOTE (Chawla et al., 2002), Borderline-SMOTE (Han et al., 2005), ADASYN (He, 2008), DBSMOTE (Bunkhumpornpat et al., 2012), FLEX-SMOTE (Bunkhumpornpat et al., 2024), ASRDO (Tao et al., 2024), etc.

- *Hybrid methods*, which blend the two methods of sampling. For example, SMOTE-ENN (Batista et al., 2004), SMOTE-Tomek (Batista et al., 2004), SMOTE using edited displacement-based k- nearest neighbor (SMOTE-CDNN) (Wang et al., 2023), etc.

Table 1 lists the common sampling techniques along with their advantages and disadvantages.

Table 1. The standard resampling techniques have their pros and cons (Wang et al., 2023).

Technique category	Resampling techniques	Pros	Cons
Oversampling	ROS (Batista et al., 2004)	Easier to implement	Copying minority sample data can lead to overfitting
	SMOTE (Chawla et al., 2002)	Generates artificial examples instead of duplicating existing data	May create noise close to class boundaries
	Borderline-SMOTE (Han et al., 2005)	Emphasizes border samples	Risk of noisy regions being oversampled
	ADASYN (He, 2008)	Produces samples based on local class density	Risks oversampling in low-density areas
	KmeansSMOTE (Fonseca et al., 2021)	Uses K-Means clustering to minimize noise	Requires clustering phase, may overfit extreme samples
	FLEX-SMOTE (Bunkhumpornpat et al., 2024)	Handles unbalanced data	Uses minority class region density for synthetic instances
Undersampling	RUS (Batista et al., 2004)	Enhances effectiveness	Risk of losing insightful majority samples
	Tomeklinks (Tomek, 1976)	Eliminates ambiguous instances near borders	Too cautious removal retains noise
	ENN (Wilson, 1972)	Enhances class separation	Aggressively removes informative samples
Hybrid	SMOTE-Tomek (Batista et al., 2004)	Combines oversampling and undersampling benefits	May retain noisy instances
	SMOTE-ENN (Batista et al., 2004)	Balances classes efficiently	Aggressive removal may delete informative data
	SMOTE-CDNN (Wang et al., 2023)	Addresses class imbalance	Not ideal for heterogeneous classification
Ensemble	SMOTEBoost (Chawla et al., 2003)	Mitigates class imbalance	Performs poorly with noisy data.
	RUSBoost (Seiffert et al., 2010)	Simple, fast, and less complex	Information loss
	CSBBoost (Salehi & Khedmati, 2024)	Reduces redundancy and loss during resampling	Difficult adaptation for large datasets

3.3 SMOTE

By interpolating between numerous minority classes' imbalances within a particular neighborhood, the SMOTE algorithm rebalances the initial training set using an oversampling technique. The formal process works by introducing synthetic examples instead of just replicating observations of the minority class (Chawla et al., 2002). To accomplish this goal, we can either use a wrapper process or configure it to achieve a nearly 1:1 class distribution. First, we calculate the number N, the total oversampling quantity of the minority samples. The first step involves randomly selecting one instance of the minority class from the X_{min} subset. The next step involves carrying out a multi-phase iterative process. Finding K's closest neighbors is the next stage. N of these K randomly chosen examples are then interpolated to determine the new instances. We do this by contrasting the characteristic vector (sample) under consideration with each of the identified neighbors. We multiply the previous feature vector by an arbitrarily chosen value between 0 and 1 and then add this difference to it (Fernández et al., 2018). The patterns produced by SMOTE may not always adhere to the initial distribution of the minority class. This study offers a new theoretic look at the SMOTE approach by figuring out the probability distribution of the samples created by SMOTE (Elreedy et al., 2024). Algorithm 1 provides an outline of the entire procedure.

Algorithm 1. SMOTE (Chawla et al., 2002)

```
Input:   $X_{min:}$ Minority class dataset, $X_N$: Quantity of synthetic samples, $k$: Quantity of nearest
         Neighbors.
Output: Augmented $D_B$ dataset.
Method:
1) Compute the $k$-$NN$ for each sample in $X_{min}$ using Euclidean distance.
2) For each minority class sample $x_i$ in $X_{min}$:
   (a) Arbitrarily select one of its $k$-$NN$ $x_{nn}$.
   (b) Calculate the difference vector: diff = $x_{nn} - x_i$.
   (c) Generate an arbitrary number r between 0 and 1.
   (d) Make a synthetic minority class sample: $x_{synth} = x_i + r * \text{diff}$.
3) The process continue until $X_N$ synthetic samples are generated.
4) Add the synthetic samples $X_N$ with the original minority class samples $X_{min}$ to create an augmented $D_B$
   dataset.
```

3.4 Autoencoder

The study of autoencoders (AEs) has been prominent in unsupervised learning due to their capacity to learn data features and serve as a feature reduction strategy (Bank et al., 2020). To better understand AEs, we must examine their conventional design, as shown in **Figure 1**.

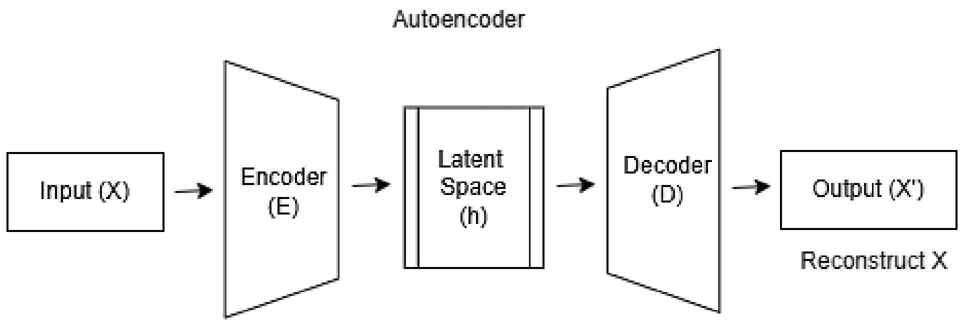


Figure 1. Autoencoder.

The term "latent feature representation" refers to the fact that the encoder and decoder are only functions. Generally, we expect the AE to efficiently reconstruct the input. At the same period, it should produce a meaningful and useful latent feature space (the encoder part's output in **Figure 1**). During the learning process, we extract the key information needed to solve an issue (e.g., data resampling). The latent feature space can be quite helpful for various tasks, including feature extraction, or for understanding the key characteristics of a dataset. The encoder and decoder in the majority of common architectures are neural networks, which are easily trained using pre-existing software libraries like *PyTorch* or *TensorFlow* with backpropagation (Michelucci, 2022). In common, the encoder can be shown as a function *f* that will rest on some parameters $h_i = f(x_i)$, where $h_i \in R^q$ (space of the latent feature) is the outcome of the encoder section in **Figure 1** when we assess it on the input x_i. It should be remembered that we will have $f: R^n \to R^q$. The decoder (and the network output that we will specify by x_i') can be specify then as a second basic function g of the latent features:

$$x_i' = g(h_i) = g(f(x_i)) \tag{1}$$

where, $x_i' \in R^n$. Training an AE simply means verdict the functions $f(\cdot)$ and $g(\cdot)$ that satisfy the Equation (2) as

$$argmin_{f,g}\mathbb{E}[\Delta(x_i, g(f(x_i)))] \tag{2}$$

where, $\mathbb{E}$ denotes the average of all instances, and Δ denotes a measure of the variance between the input and output of the AE (essentially, the loss function Equation (2) will penalize the variance between input and output). Finding f and g might be possible based on how the AE is set up, allowing it to learn to recreate the input x_i' in Equation (1) perfectly as its output (Michelucci, 2022). This paper begins with a thorough analysis of AEs, explaining the basic idea of a traditional AE and its main development process (Berahmand et al., 2024).

3.5 Hyperparameters in AE

AEs have several hyperparameters that need to be set before training, and the model's effectiveness can be greatly impacted by their settings. The following is a list of the most popular AE hyperparameters (Berahmand et al., 2024):

- *Hidden Layers*: The number of hidden layers in the AE determines its network depth and its capacity to capture complex input patterns. Increasing the number of hidden layers in a model can improve its representational power, but it also increases the possibility of overfitting and brings optimization difficulties. This parameter is set up before training.

- *The Quantity of Neurons in Every Layer*: Increasing neurons in each layer boosts the network's strength, but it also complicates optimization and increases overfitting. Typically, the set up occurs prior to training.

- *Latent Space Size*: The performance-to- model complexity ratio can be changed by adjusting the size of the bottleneck layer. This parameter is set before the start of the training procedure.

- *The Selection of Activation Functions*: These functions determine the network's nonlinearity and capacity to recognize complex data patterns. Bottleneck layers frequently use Sigmoid, ReLU, tanh, and SELU activation functions.

- *The selection of Objective Functions*: The objective function, also referred to as the loss function, is a key component of an AE that helps train the network by minimizing the variance between input and output data. The objective functions listed below are commonly used by autoencoders:
    - Mean Squared Error (MSE): The autoencoder's main goal is to calculate the average squared difference between the input and output data. The definition of MSE is

$$Loss(X, X') = \min(\|X - X'\|_F^2) \tag{3}$$

where, $\| \quad \|_F^2$ signifies the squared Frobenius norms.

- Binary Cross-Entropy (BCE): When the input data is binary (0 or 1), the BCE function determines the binary cross-entropy loss difference between the expected and actual output. It is defined as:

$$Loss\ (X, X') = -\sum_{i=1}^{n}(x_i \log(x_i')) + (1 - x_i)\log\,(1 - x_i') \tag{4}$$

With its strength against outliers and recall to data scaling, MSE is well-suited for regression tasks. BCE can be statistically unstable when it comes to 0 or 1 probability, but it is for binary classification. The most widely used AE loss function, MSE, measures input-output differences in the latent space.

- *Learning Rate*: In the optimization process, it determines the step size. The objective function's rate of convergence is impacted, as are changes to weight and bias. Overshooting may result from high values, whereas local minima trapping may occur from low ones.

- *The Selection of Optimization Techniques*: In the training process, autoencoders use optimization techniques to lower the objective function. These methods change the weights and biases of the network to properly train the encoder. The hyperparameters may need to be changed during training, even though we select the optimization approach beforehand. Several optimization techniques can be used to train the autoencoder, but the most popular ones are Adagrad, Adam, and Stochastic Gradient Descent (SGD). We explain each of these techniques in more detail below:
    - SGD: This well-known technique updates the network parameters after analyzing small batches of data. Despite being computationally efficient, complex models and datasets may take a while to converge. It is frequently necessary to carefully modify early learning rates.
    - Adam: The approach accelerates convergence and reduces the likelihood of a local minimum by utilizing momentum, adjustable learning rates, and SGD characteristics. The system works well on targets that are noisy and non-stationary, although the hyperparameters β_1 and β_2 must be adjusted.
    - Adagrad: The algorithm is adaptive, regularly updating parameters to modify learning rates. It can converge fast and performs well with sparse data, but it struggles with non-convex optimization and can converge prematurely.

The type of loss function, model complexity, dataset size, and computational resources all influence the optimization algorithm selection.

- *Number of Epochs*: Training iterations, or epochs, stand for complete dataset passes. Although they run a chance of overfitting, more epochs can improve model accuracy. The size of the dataset and the difficulty of the task determine the optimal count.

- *Batch Size*: In every optimization iteration, batch size impacts both optimization efficiency and gradient noise. Smaller sizes result in faster, memory-efficient optimization but noisier gradients. While larger sizes provide steady gradients, optimization becomes slower and requires more resources.

3.6 Adversarial Autoencoder

An Adversarial Autoencoder (AAE) with a deep encoder (*E*) and decoder (*D*) can be defined using *x* as the input and *z* as the latent code vector (hidden units). Let us consider $q(z \mid x)$ as an encoding distribution, $p(x \mid z)$ as the decoding distribution, and $p(z)$ as the previous distribution that we wish to apply to its codes. Additionally, consider the data distribution to be $p_d(x)$ and the model distribution to be $p(x)$. The following is the accumulated posterior distribution of $q(z)$ on the AE's hidden code vector defined by the encoding function $q(z \mid x)$:

$$q(x) = \int q(z|x)\, p_d\,(x) \tag{5}$$

Comparing the accumulated posterior, $q(z)$, to an arbitrary prior, $p(z)$, regularizes the AAE. The adversarial network (AN) guides $q(z)$ to match $p(z)$. In the meantime, the AE aims to reduce the reconstruction error. The encoder for the AE $q(z \mid x)$ is also the generator of the AN. By using the combined posterior distribution, the encoder ensures that the discriminative AN can be tricked into believing that the hidden code $q(z)$ is from the actual prior distribution $p(z)$. We train the AE and the AN together on small data sets using SGD in two steps: the regularization phase and the reconstruction phase. By updating both the encoder and the decoder, the AE lowers the input reconstruction error during the reconstruction phase. The adversarial network first expands its ability to tell the difference between the examples it creates and the hidden codes set by the AE during the regularization phase. Next, by changing its generator, which also acts as the encoder for the AE, the AN confuses the discriminative network. Once training is finished, the AE's decoder will build a generative model that converts the levied prior of $p(z)$ to the data distribution (Makhzani et al., 2015). The adversarial loss evaluates how well AAE can produce data that closely look like the initial distribution of input data. The decoder wants to reduce the discriminator's accuracy in identifying the real and produced data portion, while the discriminator wants to increase its accuracy. The expression for AAE's total loss function is:

$$Loss\,(X, X') = \min(\|X - X'\|_F^2 + \log D(X) + \log(1 - D(G(Z)))) \tag{6}$$

Figure 2 illustrates this process by adhering an adversarial network to the AE's concealed code vector. The decoder function *G*(*Z*) in Equation (6) resets the latent space to the initial input data, while *D*(*X*) denotes the discriminator's output for the initial input data. The term $log\,(1 - D\,(G\,(Z))$ represents the discriminator output for the data produced by the *D* (Berahmand et al., 2024). In appendix A.1, the AAE's pseudo-code is provided in Algorithm 2.

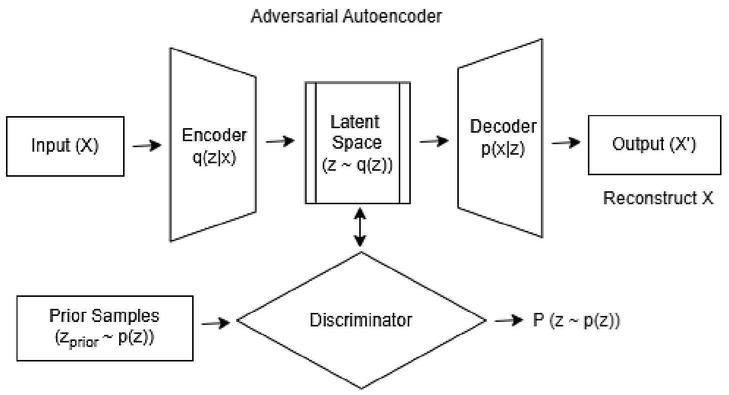


Figure 2. The design represents an adversarial autoencoder (AAE). The upper layer, a typical autoencoder, reconstructs an input *x*′ from a latent code z. The second network at the bottom is trained to tell the difference between samples that come from a sampled distribution chosen by the user and samples that come from the autoencoder's hidden code.

3.7 Random Forest

Definition: A RF classifier is made up of several tree structured predictors

$$\{h\,(x,\Theta_k),\ k = 1, 2, \ldots\} \tag{7}$$

where, each Θk *in* Equation (7) is an independently sampled random vector that is an independent random vector that is identically distributed, and each tree votes for the most favored class at input x with a unit vote (Breiman, 2001). An ensemble classifier known as a random forest is created by joining tree predictors so that each tree depends on the values of randomly selected inputs that are equally and separately sampled for every tree in the forest. As the quantity of trees in a forest surges, the generalization error settles to a certain amount. The depth and correlation of each individual tree determine the generalization error for a forest of tree classifiers. Using an arbitrary feature selection, we divide each node, producing error rates that are resilient to noise and compare well. As more features are added, internal estimations that track correlation, inaccuracy, and strength shed light on the splitting process. The significance of different variables is frequently evaluated using internal estimates (Breiman, 2001; Li, 2022; Distefano et al., 2024).

4. Performance Measures

More and more applications are using ML models to classify data into distinct groups. However, to guarantee the accuracy and dependability of these models, performance evaluation is required. In this assessment process, the confusion matrix is a crucial tool (Mahmudah et al., 2021). The confusion matrix is given in **Table 2**.

The definitions of the terms TP, FP, FN, and TN are as follows.

- *Truly Fraud Transaction* (*TP*): The prediction of fraud examples is accurate.
- *False Fraud Transaction* (*FP*): Real examples are falsely predicted as fraud examples.
- *False Legitimate Transaction* (*FN*): Fraud samples are incorrectly predicted as real examples.
- *Truly Legitimate Transaction* (*TN*): The prediction of real examples is accurate.

Table 2. Confusion matrix.

Actual Class ↓ \ Predicted class →	**Predicted fraud transaction**	**Predicted legitimate transaction**
Actual Fraud Transaction	Truly Fraud Transaction (TP)	False Legitimate Transaction (FN)
Actual Legitimate Transaction	False Fraud Transaction (FP)	Truly Legitimate Transaction (TN)

Here is an explanation of the performance metrics:

- *Precision*: It is the fraction of true positive examples to the total number of predicted positive examples, i.e.,

$$Precision = \frac{TP}{TP+FP} \tag{8}$$

- *Sensitivity or Recall*: It is the fraction of true positive examples to the total number of actual positive examples, i.e.

$$Recall = \frac{TP}{TP+FN} \tag{9}$$

- *Accuracy*: It is the fraction of all true examples to the total examples, i.e.,

$$Accuracy = \frac{TP+TN}{(TP+FP+FN+TN)} \tag{10}$$

- *F*1-*score*: It is the harmonic mean of precision and recall and is defined as follows:

$$F1 - score = \frac{2\times Precision \times Recall}{(Precision+Recall)} \tag{11}$$

- *Matthews Correlation Coefficient* (*MCC*) (Chicco & Jurman, 2020): It is used in machine learning to assess how well binary classifiers work. It can be defined as follows:

$$MCC = \frac{(TN \times TP - FN \times FP)}{\sqrt{(TP + FP)(TP+FN)(TN+FP)(TN+FN)}} \tag{12}$$

- *ROC-AUC*: The ROC-AUC (Receiver Operating Characteristic- Area under the Curve) gauges how well a classifier can discriminate between classes. A higher value, nearer to 1, indicates better performance. Perfect categorization is indicated by an AUC of 1, while arbitrary guessing is implied by an AUC of 0.5 (Powers, 2020).

- *PR-AUC*: The PR-AUC (Precision Recall-Area under the Curve) is especially helpful for datasets that are imbalanced, since it evaluates a model's effectiveness in terms of precision and sensitivity rather than overall accuracy. A higher PRAUC indicates an improved model's capability to classify positive cases (Saito & Rehmsmeier, 2015).

- *Cohen's Kappa*: Machine learning frequently evaluates the efficacy of classification models, particularly when there is an uneven distribution of classes. A more reliable indicator of model reliability than mere accuracy is Cohen's Kappa, which normalizes observed agreement versus expected random agreement. Mathematically, Cohen's Kappa (κ) is defined as $k = \frac{\mathcal{P}_o - \mathcal{P}_e}{1-\mathcal{P}_e}$, where $\mathcal{P}_o$ is the observed agreement between the raters, and $\mathcal{P}_e$ is it the expected agreement by chance. Values of κ range from -1 to 1, where 1 specifies faultless agreement, 0 means consensus equivalent to chance, and negative values suggest less than chance agreement (Cohen, 1960).

5. Proposed Work

5.1 Datasets

Table 3. Summary of datasets used.

S. No.	Dataset	Instances	Features	Feature type	Imbalance ratio	Repository
1.	Credit Card Fraud Detection (CCFD)	284807	30	Integer	577.87	Kaggle
2.	Vehicle Insurance Claim Fraud (VICF)	15420	33	Integer, Categorical	15.70	Kaggle
3.	Fastag Fraud Detection (FFD)	5000	13	Integer, Categorical	4.08	Kaggle
4.	Default of Credit Card Clients (DCCC)	30000	23	Integer	3.52	UCI
5.	Phishing Websites (PW)	11055	31	Integer	1.25	UCI
6.	Credit Approval (CA)	690	15	Integer, Categorical	1.24	UCI

We used the following datasets in our experiments: credit card fraud detection (Pozzolo et al., 2015), vehicle insurance claim fraud (Bansal, 2021), fastage fraud detection, default of credit card clients (Yeh, 2009), phishing websites (Mohammad et al., 2015), and credit approval (Quinlan, 1987). **Table 3** provides the dataset details needed for our studies. We list three UCI datasets and three Kaggle datasets. We arrange all the datasets from the financial and computer science sectors in decreasing order based on the imbalance ratio (IR). All datasets contain two classes of samples: either legitimate or fraudulent.

5.2 Experimental Setup

The Google Colab platform was used for the classification trials (Research, 2024). The available services and libraries as follow:

- Python 3 Google Compute Engine backend (TPU)
- System RAM: 334.6 GB
- Disk: 225.3 GB
- Scikit-Learn: ML in Python (Pedregosa et al., 2011)
- Problexity package (Komorniczak & Ksieniewicz, 2022)

5.3 Parameter-tuning

Tuning the parameters is a crucial process in the strategy design. It creates harmony among the components or procedures worked on in a collaborative manner. **Table 4** presents the parameters utilized in our proposed method design and evaluation, along with their consistent tuned parameters, for experimental examination.

Table 4. The parameter-tuning for the SMOTE-AAE-RF proposed method.

Technique	**Parameter value**
AAE	Random Seed = 42, input_dim, hidden_dim = 64, latent_dim = 16, discriminator_hidden = 32, batch_size = 64, lr = 1e-3, epochs = 100, Encoder (128, 64, 32; Activation = ReLU), Decoder (32, 64, 128; Activation = ReLU, Sigmoid), Optimizer = Adam, Loss = 'mse'
RF	n_estimators = 100, max_depth = None, n_jobs = -1, class_weight = "balanced", random_state = 42
SMOTE	n neighbors = 5, random state = 42
StratifiedKFold CV	n_splits = 10, shuffle = True, random_state = 42

5.4 Filtering using AAE

Let $x \in \mathbb{R}^d$ be an input sample, and let $z = E(x) \in \mathbb{R}^d$ be its corresponding latent code obtained through the encoder E. The decoder D reconstructs the input from the latent representation, giving $\hat{x} = D(z)$. During training, the AAE enforces that the aggregated posterior distribution $q(z)$ matches a chosen prior $p(z)$, typically a standard normal distribution $\mathcal{N}\,(0,1)$ (Bishop, 2006; Bengio et al., 2013). This is achieved via adversarial regularization, using a discriminator $D_z(z)$ trained to distinguish between samples drawn from $q(z)$ and $p(z)$. We define four indicators to identify undesirable samples:

1) *Reconstruction Error*: It is defined as

$$\mathcal{L}_{re}\,(x) = \|x - D(E(x))\|^2 \tag{13}$$

A high reconstruction loss suggests that the sample is either noisy or does not conform to the learned manifold.

2) *Prior Log-Likelihood*: Assuming a Gaussian prior, the log-likelihood of a latent code z under the prior is given by:

$$\log p(z) = \left(\frac{1}{2}\right) z^T z + d\log(2\pi)) \tag{14}$$

The encoded sample may be indicating noise or class overlap if the log-likelihood is low, indicating that it is located distant from the high-density zone of the prior distribution.

3) *Discriminator Confidence*: The likelihood that $z \sim p(z)$ is approximated by the discriminator output $D_z\,(z) \in [0,1]$. A low confidence level indicates a potentially troublesome sample since it suggests that the latent representation of the sample differs from the prior.

4) *Latent Space Density*: In latent space, we define a kernel density estimator as follows:

$$\rho\,(z) = \frac{1}{N}\sum_{i=1}^{N} \exp\left(-\frac{\|z - z_i\|^2}{2\sigma^2}\right) \tag{15}$$

Filtering Score**:** To quantify the likelihood that a sample is noisy, overlapping, or redundant, we define a composite score:

$$S\,(x) = \alpha.\mathcal{L}_{re}\,(x) - \beta.\log p\big(E(x)\big) - \gamma.logD_z\,\big(E(x)\big) + \delta.\rho(E(x)) \tag{16}$$

Here, $\alpha,\ \beta,\ \gamma,\ \delta \in \mathbb{R} > 0$ are weights controlling the contribution of each term. Samples are marked for filtering when $S(x)$ surpasses a predetermined threshold τ: Filter if $S(x) > \tau$. This mathematical formulation leverages the geometric properties and probabilistic nature of the adversarial regularized latent space in AAEs. By merging reconstruction error, prior mismatch, discriminator judgement, and latent density, this approach provides a rational and effective means of removing noisy, class-overlapped, or high-density samples, increasing data quality and model performance (Zhou & Paffenroth, 2017; Zong et al., 2018; Ruff et al., 2021).

5.5 Proposed Method

The methodology for the suggested SMOTE-AAE-RF approach is shown in **Figure 3**. The RF, Adversarial Autoencoder, and the SMOTE constitute the SMOTE-AAE-RF method. It helps detect fraudulent transactions in datasets that are imbalanced. SMOTE corrects class imbalance and enhances model learning by creating synthetic minority class examples. In relatively large and complex data sets, the AAE's generative nature enables it to identify strong hidden representations of both legitimate and fraudulent transactions. The AAE performs its task using both autoencoding and adversarial training. The RF classifier, a powerful ensemble learning method known for its resilience and interpretability, then uses these representations. This method makes use of AAE for feature learning, SMOTE for data balance, and RF for classification. Increasing the accuracy of fraud detection and reducing the effects of data imbalance makes finding fraudulent transactions in financial systems easier. Algorithm 2, found in Appendix A.1, provides the pseudocode for the suggested approach. AAE refines the samples generated by the SMOTE technique using the ideas presented in Section 5.4. The first step is to do some simple work on the dataset. These assist us in handling typical problems such as "Not a Number" (NaN), missing or duplicate samples, and feature value translation, which is the process of converting categorical data into numerical values. A few scaling operations, including normal scaling, were also carried out. The Python version of StandardScaler was obtained from Scikit-learn. Through the removal of the mean and scaling to unit variance, preprocessing normalizes the characteristics. This technique ensures that each feature's mean is zero and its standard deviation is one. It supports machine learning models that rely on distance-based computations, including SVMs, k-NN, and neural networks. Thereafter, we performed SMOTE on the data set to balance the class proportions. Algorithm 1 outlines the process of the SMOTE technique. Several important parameters are used in this method, such as the selection of k-nearest neighbors and the random state used in the generation of synthetic minority samples. The process of creating synthetic data continues until the balance of majority and minority samples is nearly equal. The SMOTE is simple to implement and serves as a standard for addressing class imbalance issues with data sets, making it more popular. However, the SMOTE-generated samples have some problems, including noise, outlier samples, misclassification risk, overlapping classes, and overfitting. All minority samples are considered equally important by SMOTE. These issues raise dou- bts about the effectiveness of SMOTE in addressing imbalances in large, complex data sets that are highly imbalanced. The application of AAE on synthetic samples generated by the SMOTE easily resolves these issues. The suggested method uses the AAE as a filter to look at the traits of the SMOTE-made synthetic minority samples. There are two parts that compete with each other during the adversarial training phase. The discriminator attempts to express the difference between encoded representations z and real samples from the prior distribution $p(z)$, i.e., $P = (z - p(z))$. The encoder function $q(z \mid x)$ takes inputs x and turns them into latent vectors z. The latent space is essentially regularized as the encoder learns to produce latent representations z that are similar to the prior z. The decoder network makes sure that the embedding $p(x \mid z)$ are meaningful and structured by putting together the original input x from the latent representation z after training. This adversarial regularization makes it easier for the model to produce more varied and better samples. It also improves generalization in later tasks like finding anomalies or fraud. In short, an AAE operated as a filter to check whether the synthetic samples created by the SMOTE were real or fake. Next, we recursively divide the balanced data sets into 10 folds for the training and test sets, respectively, using stratified 10-fold cross-validation to train and evaluate the RF classifier, and its classification performance is evaluated using the performance metrics outlined in Section 4.

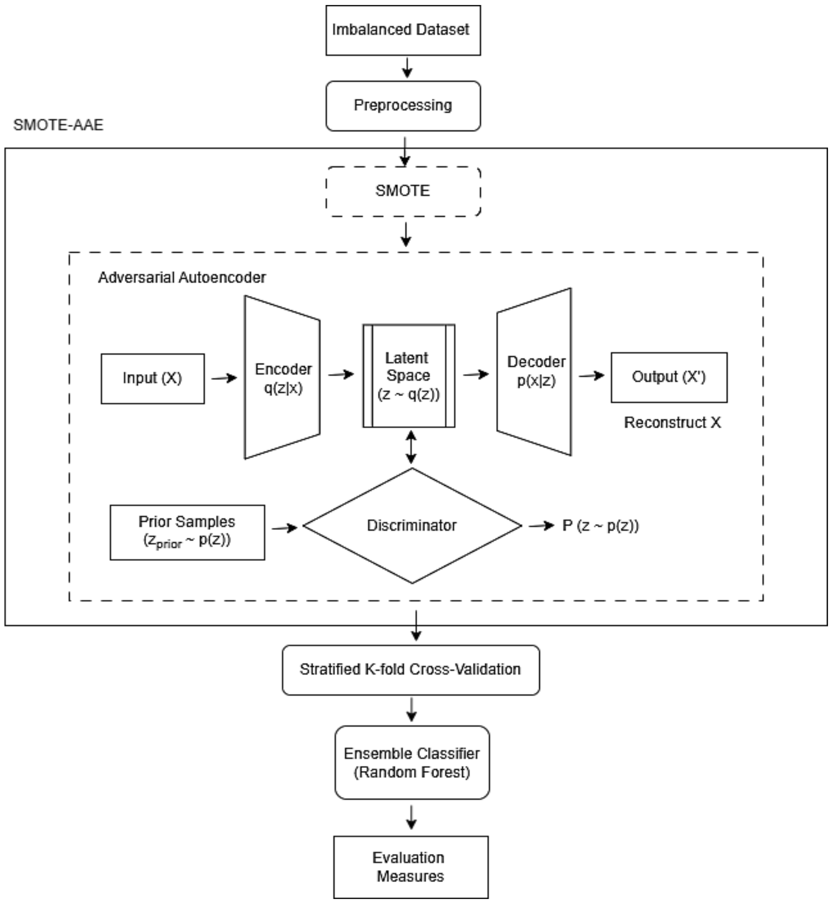


Figure 3. The methodology of the suggested method.

The SMOTE does some fundamental preprocessing before resampling the dataset. The k is set to 5 for SMOTE by default in the Python scikit-learn library; however, the dataset will determine the exact value of k. In our tests, we find the right value of k for SMOTE in the CCFD dataset using the silhouette method (Rousseeuw, 1987), as shown in **Figure 4**. **Figure 4** shows that $k = 4$, which yields the highest silhouette score for the CCFD dataset. The result indicates that $K = 4$, which is the real value utilized for SMOTE's nearest neighbors and CCFD's classification. For the other datasets, we have used $k = 5$. Then, we used the SMOTE to create synthetic minority class instances to address the class imbalance problem. To maintain consistency in the experimental outcomes, the SMOTE random state was likewise set to 42. By limiting the number of nearest neighbors (n-neighbors) to five in SMOTE, the interpolation process is impacted, and the structure of the minority class is maintained while excessive oversampling noise is avoided. The number of trees in the forest (n-estimators) has to be adjusted to 100 to tune the RF classifier's parameters, which balances processing power and model performance. We select 42 as the random state to ensure reproducibility. These hyperparameters were selected to maximize the model's effectiveness on the imbalanced dataset, stability, and generalization.

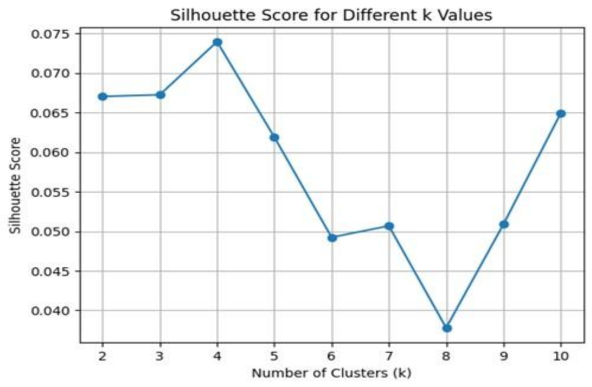
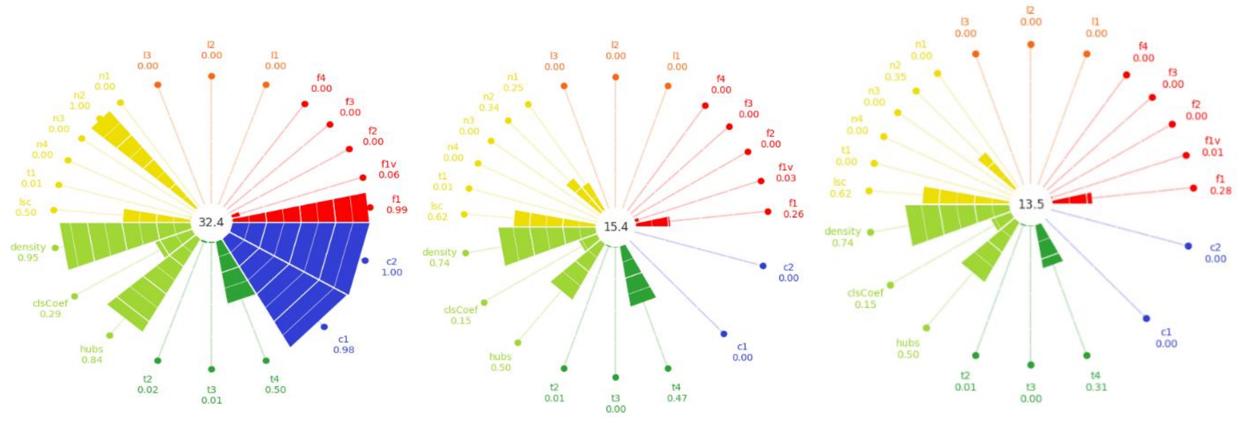


Figure 4. The silhouette method is used to select the actual value of k for the CCF detection.

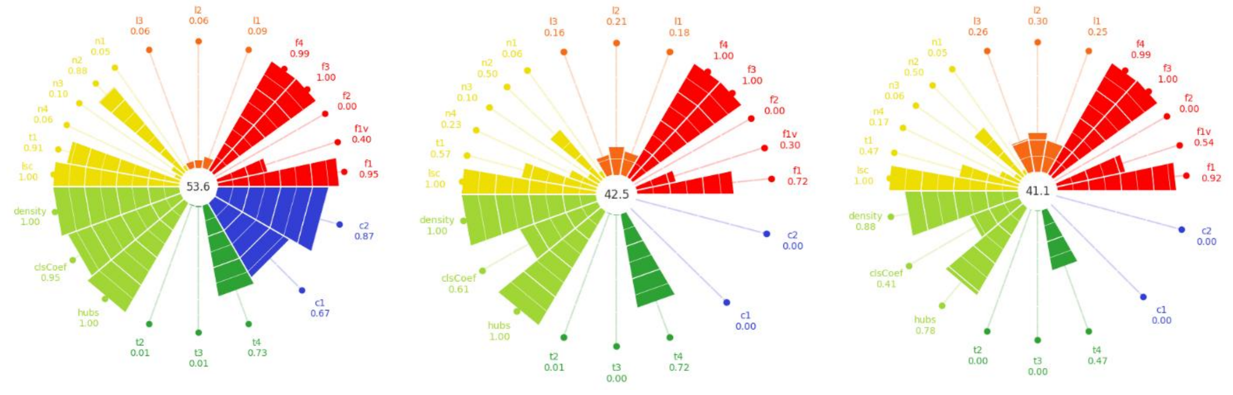
6. Results and Discussion

We carried out an evaluation of data complexity for the datasets given in **Table 3**. The data complexity of each dataset was evaluated in three different forms: original, SMOTE's resampled, and SMOTE-AAE encoded. To achieve this, we utilize the Problexity package (Komorniczak & Ksieniewicz, 2022), which is easy to install. We can find the complexity module and import it using the regular Python procedure. The problem difficulty scores for the CCFD and VICF datasets (shown in the middle of each graph in **Figure 5** and **Figure 6**) are the average of all the complexity measures mentioned in subsection 3.1. **Figure 5** displays the experiment results and data complexity scores for the CCFD data sets. The problem difficulty scores are 32.4, 15.4, and 13.5 for the CCFD datasets in the original, resampled, and encoded formats, respectively. We need to repeat the same exercise for the VICF dataset. **Figure 6** displays the results, revealing problem difficulty scores of 53.6, 42.5, and 41.1 for the original, resampled, and encoded data sets, respectively. The encoded datasets created using the SMOTE-AAE technique had the smallest problem difficulty score compared to both the original and the resampled SMOTE datasets. **Table 5** presents the problem difficulty (data complexity) scores for all datasets, including original, resampled, and encoded versions. It also indicates the difference in problem difficulty scores between the original datasets and the resampled datasets, the original datasets and the encoded datasets, and the resampled datasets and the encoded datasets, which are referred to as Pd_1, Pd_2, and Pd_3, respectively. It is clear from columns Pd_2 and Pd_3 of **Table 5** that our proposed method is more effective in reducing the problem difficulty scores for all datasets except the Fastag dataset for fraud detection. In this case, the problem difficulty score has increased 4.71% as compared to the resampled dataset. Moreover, the results indicate that the suggested SMOTE-AAE method is more effective in reducing the problem difficulty (data complexity) of the data set. Thereafter, we split the encoded datasets in an 80:20 ratio across training and testing sets. The SHAP feature selection process based on XGBoost identifies the most important features that influence model predictions. An XGBoost classifier is trained once the data has been preprocessed and standardized, and the SHAP value of each feature is calculated to determine its contribution. Features are sorted by mean absolute SHAP values, and the top *k* most significant features (in our case, *k* = 10) are selected. These best features are then used to retrain the model, which improves interpretability, often boosts performance, and removes redundant features (Lamane et al., 2024). Using SHAP-based feature selection with XGBoost, the features chosen for the datasets are displayed in **Figures 7(a)-(f)**, respectively. This procedure improves the model's efficiency and readability while reducing the likelihood of overfitting. Keeping only the most informative variables enhances the performance of ML models, lowering computing costs and increasing fraud detection accuracy (Salekshahrezaee et al., 2023).



(a) (b) (c)

Figure 5. Data complexity analysis in three different CCFD formats: (a) original dataset; (b) resampled dataset of SMOTE; and (c) encoded dataset of SMOTE-AAE. The problem difficulty scores are in the centers of each figure.



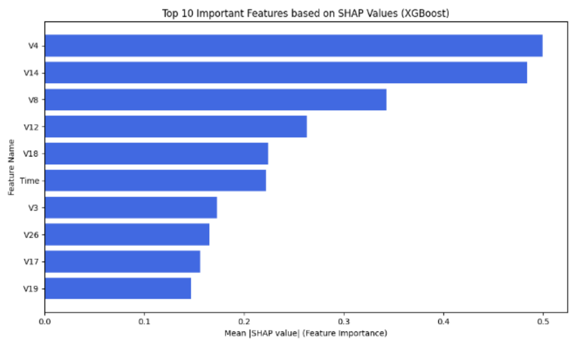
(a) (b) (c)

Figure 6. Data complexity analysis in three different dataset formats of VICF detection: (a) original dataset; (b) resampled dataset of SMOTE; and (c) encoded dataset of SMOTE-AAE. The problem difficulty scores are in the centers of each figure.

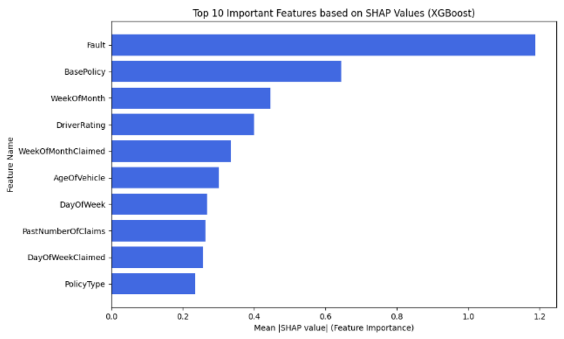
Table 5. The data complexity scores of all the datasets (each in different forms: original, resampled, and encoded).

Datasets	Problem difficulty scores					
	Original (O)	Resampled (R)	Encoded (E)	Pd_1 (%)	Pd_2 (%)	Pd_3 (%)
CCFD	32.4	15.4	13.5	110.38	140	14.05
VICF	53.6	42.5	41.1	26.12	30.41	3.40
FFD	41.6	36.4	38.2	14.28	8.90	- 4.71
PW	32.8	32.8	30.3	0.00	8.25	8.25
CA	38.8	38.3	35.1	1.30	10.54	9.11

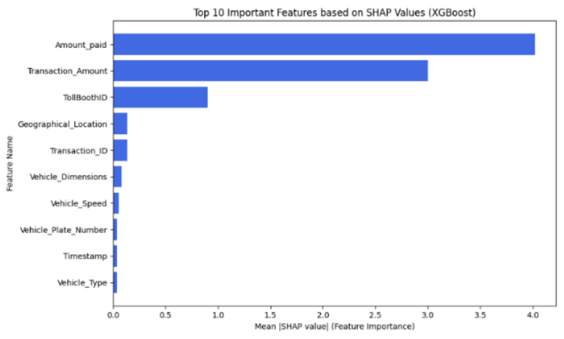
Here, $Pd_1 = O - R$, $Pd_2 = O - E$, and $Pd_3 = R - E$ are reduced problem difficulty scores.



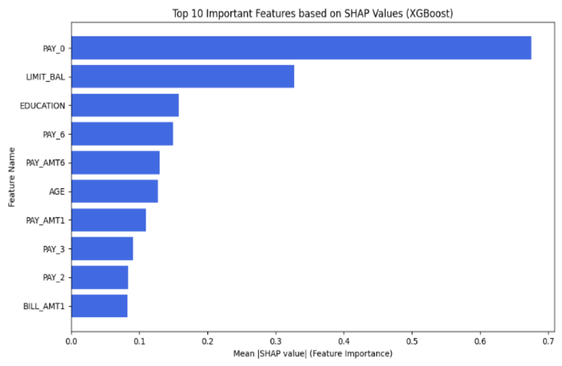
(a)



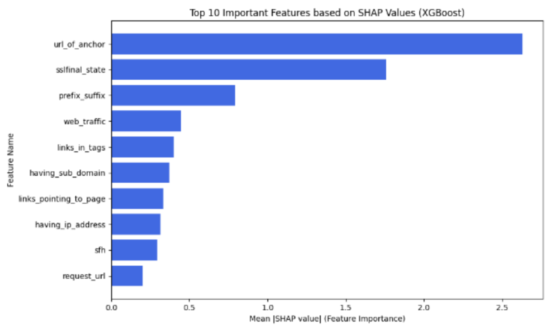
(b)



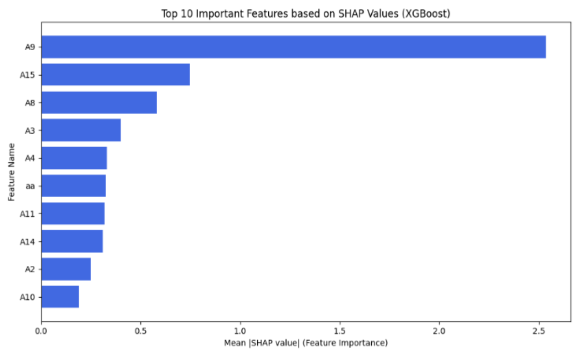
(c)



(d)



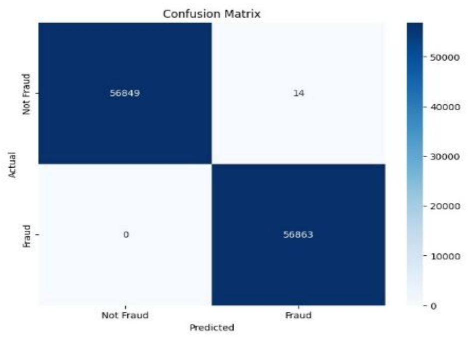
(e)



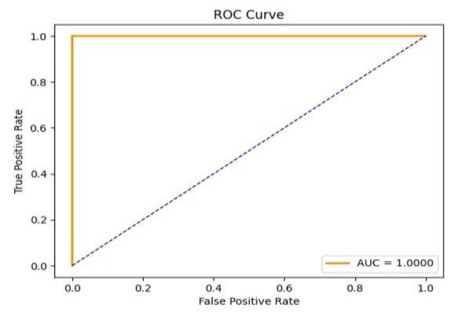
(f)

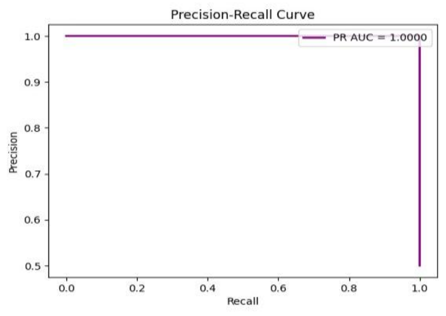
Figure 7. SHAP-based feature selection using XGBoost for the datasets (a) CCFD, (b) VICF, (c) FFD, (d) DCCC, (e) PW, and (f) CA.

Figures 8(a)-(c) display the confusion matrix, ROC-AUC, and PR-AUC for detecting CCF, respectively. The area under the curve for both PR-AUC and ROC-AUC was 1.0000. We repeated a similar exercise using the VICF dataset to assess the performance of the SMOTE-AAE-RF method. Again, we used the confusion matrix, ROC-AUC, and PR-AUC to show the results of our proposed method for the VICF dataset. **Figure 9 (a)-(c)** presents the results as the confusion matrix; the ROC-AUC was 0.9961, and the PR-AUC was 0.9967, respectively.



(a)



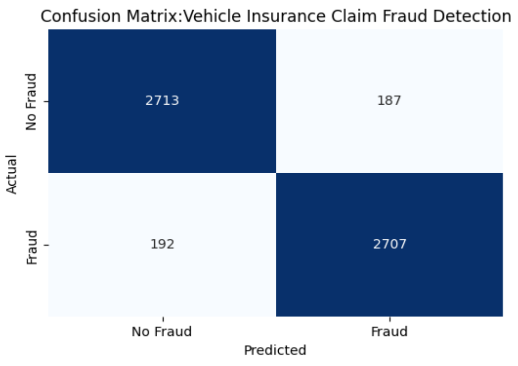


(b) (c)

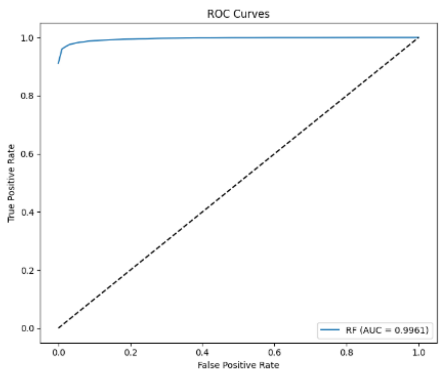
Figure 8. The results of the suggested method are shown for CCFD using (a) confusion matrix. (b) AUC-ROC, and (c) PR AUC, which both reached a highest value of 1.

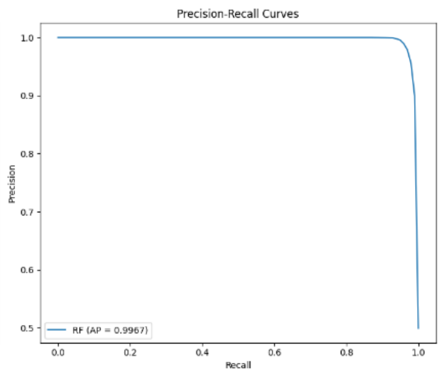
The results (in average) in **Table 6** of Appendix A.2 show that the proposed method works very well on all six datasets, proving it is effective for both imbalanced and moderately balanced classification tasks. For detecting credit card fraud, the model achieves near-perfect scores: accuracy of 0.9999, precision of 0.9998, recall of 1.0000, and an $F1$-score of 0.9999. The ROC-AUC and PR-AUC achieve the optimal value of 1.0000, demonstrating exceptional discrimination abilities, while the MCC and Cohen's Kappa are both 0.9996. The model also performs well in the situation of fraud pertaining to the VICF dataset, with an accuracy of 0.9773 and a precision of 0.9992. Recall is marginally lower at 0.9559, but the $F1$-score is still excellent at 0.9762, with ROC-AUC and PR-AUC at 0.9961 and 0.9967, respectively, and MCC and Kappa at 0.9374 and 0.9356. The FFD dataset yields an $F1$-score of 0.9955, MCC and Kappa of 0.9899 and 0.9898, and AUC-ROC/PR-AUC at 1.0000 when perfect precision (1.0000) and high recall of 0.9903 are maintained. The model achieves a balanced performance on the DCCC dataset, with an $F1$-score of 0.8961, accuracy of 0.8786, precision of 0.8816, and recall of 0.8934. Under more balanced class distributions, the

MCC and Kappa are 0.7727 and 0.7664, respectively, while the ROC-AUC and PR-AUC are higher than 0.9540, indicating dependable classification. The model yields an almost perfect ROC-AUC (0.9965) and PR-AUC (0.9956) for the PW dataset, with good recall (0.9823) and precision (0.9734), resulting in an $F1$-score of 0.9760 and MCC/Kappa of 0.9513. The model also obtains an MCC of 0.8926, strong ROC-AUC/PR-AUC values (0.9512 and 0.9472, respectively), an accuracy of 0.8961, a precision of 0.8998, a recall of 0.8858, and an $F1$-score of 0.8926 for the CA dataset. All of these findings demonstrate that the suggested approach is reliable, accurate, and applicable to various fraud and decision-making data sets. We also test the superiority of the proposed methods to the other advanced methods used for detecting CCF. In our comparison, we have included eleven advanced methods. However, **Table 7** in Appendix A.2 shows how well these methods performed on key measures like accuracy, precision, recall, $F1$-score, MCC, Kappa, and AUC for detecting CCF when the data was imbalanced. The suggested method has outperformed others on all performance metrics. We also compared the performance of the SMOTE-AAE-RF method with the advanced approaches for detecting fraud in vehicle insurance claims mentioned in Appendix A.2's **Table 8**. The results indicate that our methods outperformed the others in terms of PR-AUC and ROC-AUC metrics. On other metrics, our method lagged behind the competing method (Gheysarbeigi et al., 2025).



(a)





(b) (c)

Figure 9. The results of our suggested method for the vehicle insurance claim fraud detection dataset using (a) the confusion matrix, (b) the AUC-ROC, and (c) the PR AUC.

6.1 Ablation Study

This ablation study examines how using SMOTE, Adversarial Autoencoders (AAE), and their combination (SMOTE-AAE) impacts the performance of RF classifiers on six datasets with imbalanced classes: CCFD, VICF, FFD, DCCC, PW, and CA. To determine how each component aids in identifying the minority class, tests were conducted on each of the following setups: RAW+RF, SMOTE+RF, AAE+RF, and SMOTE-AAE+RF. The results included accuracy, precision, recall, F1-score, ROC-AUC, and PR-AUC. **Table 4** displays the parameters that we used in our tests. We examine the following information based on **Tables 9** and **10**.

- *CCFD*: The RAW+RF baseline on the CCFD dataset exhibits a low recall (0.7662) and extremely high accuracy (0.9995), suggesting that the classifier tends to favor the majority class. Due to SMOTE's ability to balance the dataset, SMOTE+RF achieves near-perfect recall and precision, which greatly improves performance. Although AAE+RF is not as effective as RAW+RF, it is still inferior to SMOTE-based methods. The combination of SMOTE and AAE (SMOTE-AAE+RF), however, yields flawless scores in every category, demonstrating that AAE enhances SMOTE by eliminating subpar synthetic samples.

- *VICF*: RAW+RF's poor ability to recognize the minority class in this highly imbalanced dataset, with a very low recall (0.0087) and an average accuracy (0.9404), highlights its inability to identify false claims. SMOTE+RF illustrates the advantages of synthetic oversampling by considerably inc reasing recall to 0.9488 with an almost perfect $F1$-score (0.9732). AAE+RF can't fix the imbalance alone, as it only slightly beats RAW+RF. The best results, with slightly better recall and $F1$-score than SMOTE+RF, are obtained with the hybrid SMOTE-AAE+RF configuration, demonstrating that AAE efficiently filters noisy synthetic samples from SMOTE.

- *FFD*: For the FFD dataset, RAW+RF achieves great precision (1.0000) but low recall (0.7783), suggesting that it is not detecting fraudulent behavior. The significant improvements in recollection (0.9515) and $F1$-score (0.9751) demonstrated by SMOTE+RF are consistent with earlier research on its efficacy. As with other datasets, the recall of AAE+RF alone is lower (0.7196). But SMOTE-AAE+RF keeps precision, ROC-AUC, and PR-AUC at 1.0000 while improving recall (0.9903) even more. This result shows that AAE improves SMOTE outputs' robustness against noisy or borderline synthetic features.

- *DCCC*: RAW+RF has a moderate accuracy (0.8170) for the DCCC dataset, but its recall (0.3379) and $F1$-score (0.4495) are poor. SMOTE+RF significantly enhances performance in all parameters, particularly PR-AUC (0.9353), ROC-AUC (0.9354), and recall (0.8462). Although it surpasses RAW once more, AAE+RF is not as successful as SMOTE. The SMOTE-AAE+RF setup achieves the best overall performance, marginally outperforming SMOTE+RF. This suggests that AAE adds value by minimizing noisy or overlapping synthetic data.

- *Phishing Websites*: The RAW+RF baseline also obtains satisfactory results (accuracy: 0.9722, $F1$-score: 0.9752) since the PW dataset is fairly balanced. AAE+RF and SMOTE+RF both continue to perform well with very slight enhancements. SMOTE-AAE+RF performs significantly better than other models. This indicates that while AAE provides robustness against possible overfitting from oversampling, neither SMOTE nor AAE substantially changes the classifier performance in balanced or nearly balanced situations.

- *Credit Approval*: RAW+RF performance on the CA dataset is already rather excellent (accuracy: 0.8739), but there is potential for improvement based on recall (0.8372) and $F1$-score (0.8554). AAE+RF yields results comparable to the baseline, while SMOTE+RF slightly increases recall (0.8159). In terms of $F1$-score (0.8926), ROC-AUC (0.9512), and PR-AUC (0.9472), the SMOTE-AAE+RF combination performs better than any other approach. This shows how the complementing qualities of SMOTE (balancing) and AAE (noise filtering) work well together on datasets that are mildly imbalanced.

Table 9. An evaluation of the relative performance (mean ± standard deviation) of RAW+RF, SMOTE+RF, AAE+RF and SMOTE-AEE+RF on six imbalanced datasets.

Datasets	Method	Accuracy	Precision	Recall	F1-score	ROC-AUC	PR-AUC
CCFD	RAW+RF	0.9995 ± 0.0001	0.9604 ± 0.0432	0.7662 ± 0.0293	0.8520 ± 0.0300	0.9520 ± 0.0126	0.8523 ± 0.0387
	SMOTE+RF	0.9999 ± 0.0000	0.9998 ± 0.0001	1.0000 ± 0.0000	0.9999 ± 0.0000	1.0000 ± 0.0000	1.0000 ± 0.0000
	AAE+RF	0.9995 ± 0.0001	0.9480 ± 0.0186	0.7524 ± 0.0340	0.8384 ± 0.0219	0.9431 ± 0.0121	0.8365 ± 0.0184
	SMOTE-AAE+RF	**0.9999 ± 0.0000**	**0.9998 ± 0.0001**	**1.0000 ± 0.0000**	**0.9999 ± 0.0000**	**1.0000 ± 0.0000**	**1.0000 ± 0.0000**
VICF	RAW+RF	0.9404 ± 0.0007	0.5833 ± 0.4425	0.0087 ± 0.0065	0.0170 ± 0.0128	0.8511 ± 0.0100	0.2935 ± 0.0291
	SMOTE+RF	0.9739 ± 0.0008	0.9989 ± 0.0007	0.9488 ± 0.0012	0.9732 ± 0.0008	**0.9963 ± 0.0004**	**0.9969 ± 0.0003**
	AAE+RF	0.9432 ± 0.0004	0.6500 ± 0.4500	0.0070 ± 0.0053	0.0138 ± 0.0104	0.8424 ± 0.0130	0.2541 ± 0.0235
	SMOTE-AAE+RF	**0.9773 ± 0.0008**	**0.9992 ± 0.0006**	**0.9559 ± 0.0063**	**0.9762 ± 0.0008**	0.9961 ± 0.0008	0.9967 ± 0.0006
FFD	RAW+RF	0.9564 ± 0.0068	1.0000 ± 0.0000	0.7783 ± 0.0343	0.8749 ± 0.0217	0.9968 ± 0.0035	0.9968 ± 0.0035
	SMOTE+RF	**0.9757 ± 0.0034**	1.0000 ± 0.0000	0.9515 ± 0.0068	0.9751 ± 0.0036	0.9997 ± 0.0002	0.9997 ± 0.0002
	AAE+RF	0.9471 ± 0.0061	1.0000 ± 0.0000	0.7196 ± 0.0325	0.8366 ± 0.0220	0.9969 ± 0.0014	0.9904 ± 0.0035
	SMOTE-AAE+RF	0.9707 ± 0.0036	1.0000 ± 0.0000	**0.9903 ± 0.0074**	**0.9955 ± 0.0028**	**1.0000 ± 0.0000**	**1.0000 ± 0.0000**
DCCC	RAW+RF	0.8170 ± 0.0039	0.6720 ± 0.0171	0.3379 ± 0.0155	0.4495 ± 0.0156	0.7694 ± 0.0085	0.5411 ± 0.0154
	SMOTE+RF	0.8659 ± 0.0044	0.8810 ± 0.0047	**0.8462 ± 0.0057**	0.8632 ± 0.0046	0.9354 ± 0.0028	0.9353 ± 0.0031
	AAE+RF	0.8228 ± 0.0020	0.6638 ± 0.0088	0.3367 ± 0.0126	0.4467 ± 0.0114	0.7739 ± 0.0090	0.5336 ± 0.0115
	SMOTE-AAE+RF	**0.8786 ± 0.0040**	**0.8816 ± 0.0036**	**0.8934 ± 0.0069**	**0.8961 ± 0.0045**	**0.9575 ± 0.0027**	**0.9543 ± 0.0033**
PW	RAW+RF	0.9722 ± 0.0033	0.9700 ± 0.0055	0.9805 ± 0.0055	0.9752 ± 0.0030	0.9960 ± 0.0014	**0.9961 ± 0.0016**
	SMOTE+RF	0.9753 ± 0.0024	0.9718 ± 0.0045	0.9790 ± 0.0040	0.9754 ± 0.0024	**0.9966 ± 0.0009**	0.9959 ± 0.0015
	AAE+RF	0.9707 ± 0.0033	0.9692 ± 0.0036	0.9773 ± 0.0036	0.9732 ± 0.0031	0.9954 ± 0.0010	0.9955 ± 0.0011
	SMOTE-AAE+RF	**0.9756 ± 0.0051**	**0.9734 ± 0.0030**	**0.9823 ± 0.0062**	**0.9760 ± 0.0050**	0.9965 ± 0.0006	0.9956 ± 0.0008
CA	RAW+RF	0.8739 ± 0.0591	0.8786 ± 0.0784	0.8372 ± 0.0767	0.8554 ± 0.0669	0.9266 ± 0.0491	0.9147 ± 0.0582
	SMOTE+RF	0.8630 ± 0.0218	**0.9014 ± 0.0235**	0.8159 ± 0.0420	0.8558 ± 0.0252	0.9271 ± 0.0114	0.9301 ± 0.0161
	AAE+RF	0.8705 ± 0.0243	0.8851 ± 0.0353	0.8075 ± 0.0385	0.8440 ± 0.0308	0.9275 ± 0.0203	0.9054 ± 0.0291
	SMOTE-AAE+RF	**0.8961 ± 0.0227**	0.8998 ± 0.0255	**0.8858 ± 0.0250**	**0.8926 ± 0.0233**	**0.9512 ± 0.0167**	**0.9472 ± 0.0192**

Table 10. Compare SMOTE+RF and SMOTE-AAE+RF's training time performance (mean ± standard deviation).

S. No.	Dataset	SMOTE+RF Training time (Mean$\pm SD$) seconds	SMOTE-AAE+RF Training time (Mean$\pm SD$) seconds	Extra overhead Training time (Mean$\pm SD$) Seconds
1.	CCFD	506.40 ± 18.84	497.66 ± 7.45	-8.74 ± 0.45
2.	VICF	2.46 ± 0.23	3.13 ± 0.22	0.67±0.01
3.	FFD	23.10 ± 2.10	25.73 ± 0.90	2.63±1.2
4.	DCCC	7.53 ± 0.47	8.14 ± 0.32	0.61±0.15
5.	PW	0.57 ± 0.11	0.75 ± 0.06	0.18±0.05
6.	CA	0.50 ± 0.14	0.32 ± 0.03	-0.18±0.11

Training Time Analysis: SMOTE+RF vs. SMOTE-AAE+RF

We examined the average training duration for the SMOTE+RF and SMOTE-AAE+RF processes to see whether it is feasible to use an AAE as an additional step following SMOTE. The datasets provide a comprehensive picture of the computational trade-offs involved, spanning multiple domains:

- Using the CCFD dataset, a surprising but useful finding showed that the SMOTE-AAE+RF pipeline took less time to train than SMOTE+RF (497.66 ± 7.45 vs. 506.40 ± 18.84 seconds). A negative overhead of -8.74 ± 0.45 seconds was the outcome. To speed up RF training, this unexpected finding suggests that the AAE filtering step may reduce or simplify the synthetic dataset for the classifier.

- A slight positive overhead of 0.67 ± 0.01 seconds was incurred by SMOTE-AAE+RF for the VICF dataset (3.13 ± 0.22 vs. 2.46 ± 0.23 seconds). The additional AAE stage doesn't add any additional time because the dataset is small and the initial training is minimal, which makes it ideal for smaller tasks.

- In the FFD dataset, SMOTE-AAE+RF introduced an overhead of 2.63 ± 1.2 seconds, taking somewhat more time (25.73 ± 0.90 vs. 23.10 ± 2.10 seconds). Although the AAE takes slightly more time, it is still appropriate because prior research has indicated that it strengthens the system and helps lower false positives (Makhzani et al., 2015; Wu et al., 2020).

- The overhead for the DCCC dataset increased from 7.53 ± 0.47 to 8.14 ± 0.32 seconds; however, it was still very small at only 0.61 ± 0.15 seconds. This confirms the AAE stage's computational justifiability and lower overhead for mid-sized datasets.

- The SMOTE-AAE+RF pipeline took just 0.75 ± 0.06 seconds, whereas SMOTE+RF took 0.57 ± 0.11 seconds for the PW dataset, where both configurations generally train fast. Real-time or batch processing settings would not be affected by the negligible overhead that results (0.18 ± 0.05 seconds).

- The CA dataset revealed yet another intriguing result: SMOTE-AAE+RF actually outperformed SMOTE+RF in terms of training time (0.32 ± 0.03 vs. 0.50 ± 0.14 seconds), resulting in a negative overhead of -0.18 ± 0.11 seconds. Similar to the credit card fraud case, the outcome suggests that AAE may improve the efficiency of the training process by eliminating noisy or overlapping samples, thereby reducing the number or complexity of the final synthetic instances.

Overall, the ablation investigation on all datasets demonstrates that RAW+RF continuously performs worse in terms of recall and $F1$-score, highlighting its drawbacks when handling imbalanced data. SMOTE+RF greatly improves recall and general performance, confirming its significance in resolving class imbalance. While AAE+RF somewhat increases resilience, it is ineffective on its own because it cannot rebalance data. The combined strategy, SMOTE-AAE+RF, continuously meets or exceeds the performance of SMOTE+RF by itself, demonstrating the usefulness of AAE in removing synthetic occurrences and improving the capability of minority samples. Datasets with significant imbalance or class overlap are particularly affected by this tiered architecture. It also shows that adding an AAE to SMOTE with the RF requires very minimal additional training time on most datasets. In four out of six datasets, the additional computation time remains below 1 second, while two datasets even show a slight reduction in training time, possibly due to AAE's filtering effect that simplifies learning for the classifier. The FFD dataset shows only a moderate increase (2.63 seconds), which is still acceptable considering the improved classification performance. Overall, the computational efficiency and scalability of the SMOTE-AAE+RF approach make it a practically viable enhancement for imbalanced learning scenarios.

McNemar's test and Wilcoxon signed-rank test:
To determine statistical significance, we conducted McNemar's test (Smith et al., 2020) (for pairwise model comparison on classification outcomes) and the Wilcoxon signed-rank test (Woolson, 2007) (for performance metric comparisons such as $F1$ score across folds) at a significance level of $\alpha = 0.05$. The McNemar's test was employed to assess the statistical significance of performance differences between RF, CatBoost (Prokhorenkova et al., 2018), and LightGBM (LGBM) (Ke et al., 2017) across six fraud-related and credit-based datasets. All models were tuned using the same hyperparameter search budget, ensuring a fair comparison across methods. For consistency, we used a stratified K-folds cross-validation search with 10 evaluation folds and the $F1$-score performance metric. The results of the McNemar test (see **Table 11**) show both dataset-specific and consistent variations among classifiers. While there was no significant difference between RF and CatBoost for detecting credit card fraud, there were statistically significant differences between RF and LGBM as well as between CatBoost and LGBM, with LGBM being favored. While there was no difference between RF and CatBoost, as well as between CatBoost and LGBM, for the VICF dataset, a substantial difference existed between RF and LGBM, suggesting that RF and CatBoost performed more similarly. The three classifier pairs (RF vs. CatBoost, RF vs. LGBM, and CatBoost vs. LGBM) in FFD all displayed very significant differences, indicating more robust model-level differences. CatBoost and LGBM were the only two pairs that exhibited a significant difference for the DCCC dataset; the other pairs did not, indicating that RF and LGBM performed similarly.

Again, the evidence suggests that CatBoost and RF match more closely. In the PW dataset, RF vs. LGBM and CatBoost vs. LGBM were statistically different, but RF vs. CatBoost was not. Finally, notable changes in credit approval were observed for both RF vs. CatBoost and CatBoost vs. LGBM, while no such changes were found for RF vs. LGBM, indicating that CatBoost was more distinct from the other two classifiers. All things considered, the McNemar test shows that classifier performance varies depending on the dataset: LGBM frequently exhibits notable differences from RF and CatBoost, whereas RF and CatBoost usually behave similarly, with the exception of situations like Fastag Fraud and Credit Approval, where all classifiers diverge more dramatically.

Table 12 demonstrates the dataset-specific relevance of model differences through the Wilcoxon signed-rank test on $F1$ scores across datasets. Here, we clarified the paired folds in **Figure 10** (see Appendix A.3), including effect sizes (such as paired Cohen's d and Cliff's delta), and provided bootstrap confidence intervals (95% CI) in **Table 13**. The model performance varies significantly depending on the dataset, according to the Wilcoxon signed-rank test with bootstrap confidence intervals, effect size estimation, and paired cross-validation folds. Statistically significant p-values ($p < 0.05$) confirm robust superiority across most datasets, especially for VICF, FFD, and DCCC, where RF consistently outperformed CatBoost and LGBM with narrow bootstrap confidence intervals and strong effect sizes (Cohen's $d > 0.8$; Cliff's $\delta \approx 1.0$). The CA and PW datasets had with strong effect sizes (Cohen's $d > 0.8$; Cliff's $\delta \approx 1.0$) and narrow bootstrap confidence intervals, confirming robust superiority. In contrast, for CCFD, significance was only observed between RF and LGBM ($p = 0.0273$), while the CA and PW datasets exhibited mostly nonsignificant differences, implying mostly nonsignificant differences, which suggested that the models behaved similarly.

However, for CCFD, significance was only found between RF and LGBM ($p = 0.0273$). In contrast to CA and PW, which appear to be less susceptible to model selection, RF's discriminative power in complicated imbalanced datasets is further supported by its constant large effect sizes and positive mean differences across significant comparisons. Overall, RF performed better than expected and held steady throughout most benchmarks, closely followed by LGBM, whereas CatBoost did not perform as well as it should have. As shown in **Table 14**, the thresholding analysis conducted during 10-fold cross-validation demonstrates differences in decision boundary calibration between datasets and models. For most datasets (e.g., DCCC: mean = 0.443, SD ≈ 0.01, and VICF: mean = 0.484, SD ≈ 0.02), Random Forest (RF) demonstrated stable and moderate thresholds and little fold-wise variability, suggesting robust calibration across folds and consistent class discrimination. CatBoost typically produced higher thresholds (e.g., CCFD = 0.729; FFD = 0.930), reflecting a more conservative decision strategy that prioritizes precision over recall, whereas LGBM displayed greater variability across folds, particularly in CA and CCFD, where thresholds fluctuated widely (e.g., CA: [0.04–0.84]), indicating sensitivity to shifts in data distribution. Threshold stability was most apparent across datasets in DCCC and VICF, indicating balanced and well-calibrated performance, whereas more variability in CA and CCFD reveals dataset complexity and model adaptation concerns. Overall, RF maintained the most constant thresholding behavior, CatBoost provided steady but higher thresholds, and LGBM displayed dynamic adjustment patterns that could need further calibration to guarantee consistent decision-making.

Table 11. Results of McNemar's Test (Pairwise) on datasets resampled using SMOTE-AAE at $\alpha = 0.05$.

Datasets	Classifiers	Contingency table	Statistic	p-value	Difference b/w classifiers
CCFD	RF vs CatBoost	[[568477 24] [21 108]]	0.0889	0.7656	No significant difference
	RF vs LGBM	[[568435 66] [24 105]]	18.6778	0.0000	Significant difference
	CatBoost vs LGBM	[[568433 65] [26 106]]	15.8681	0.0001	Significant difference
VICF	RF vs CatBoost	[[28045 13] [21 915]]	1.4412	0.2299	No significant difference
	RF vs LGBM	[[28051 7] [19 917]]	4.6538	0.0310	Significant difference
	CatBoost vs LGBM	[[28055 11] [15 913]]	0.3462	0.5563	No significant difference
FFD	RF vs CatBoost	[[7993 0] [37 4]]	35.0270	0.0000	Significant difference
	RF vs LGBM	[[7991 2] [28 13]]	20.8333	0.0000	Significant difference
	CatBoost vs LGBM	[[8017 13] [2 2]]	6.6667	0.0098	Significant difference
DCCC	RF vs CatBoost	[[40725 593] [575 4835]]	0.2474	0.6189	No significant difference
	RF vs LGBM	[[40753 565] [610 4800]]	1.6477	0.1993	No significant difference
	CatBoost vs LGBM	[[40953 347] [410 5018]]	5.0779	0.0242	Significant difference
PW	RF vs CatBoost	[[11939 72] [85 218]]	0.9172	0.3382	No significant difference
	RF vs LGBM	[[11871 140] [94 209]]	8.6538	0.0033	Significant difference
	CatBoost vs LGBM	[[11917 107] [48 242]]	21.7032	0.0000	Significant difference
CA	RF vs CatBoost	[[597 9] [28 132]]	8.7568	0.0031	Significant difference
	RF vs LGBM	[[572 34] [34 126]]	0.0147	0.9035	No significant difference
	CatBoost vs LGBM	[[582 43] [24 117]]	4.8358	0.0279	Significant difference

Comparing the calibration curves for RF, CatBoost, and LGBM across datasets (a) CCFD, (b) VICF, (c) FFD, (d) DCCC, (e) PW, and (f) CA (**Figures 11(a)-(f)**, see Appendix A.3) reveals notable variations in the models' probabilistic reliability. CatBoost demonstrated better probability calibration and precisely calibrated confidence estimations, as seen by its consistent closer alignment to the diagonal reference line.

Table 12. Results of the Wilcoxon Signed-Rank Test ($F1$-score) on SMOTE-AAE-resampled datasets at $\alpha = 0.05$.

Datasets	**Models**	**Statistic**	**p-value**	**Difference b/w Classifiers**	**Overall F1-Scores (across all 10-folds)**
CCFD	RF vs. CatBoost	16.0	0.4961	No significant difference in F1 scores (fail to reject H0)	0.9999 ± 0.0000 (RF) 0.9998 ± 0.0001 (CatBoost) 0.9996 ± 0.0002 (LightGBM)
	RF vs. LGBM	6.0	0.0273	Significant difference in F1 scores (reject H0)	
	CatBoost vs. LGBM	9.0	0.1289	No Significant difference in F1 scores (reject H0)	
VICF	RF vs. CatBoost	2.0	0.0058	Significant difference in F1 scores (reject H0)	0.9762 ± 0.0000 (RF) 0.9569 ± 0.0031 (CatBoost) 0.9139 ± 0.0029 (LightGBM)
	RF vs. LGBM	1.0	0.0039	Significant difference in F1 scores (reject H0)	
	CatBoost vs. LGBM	16.0	0.0020	Significant difference in F1 scores (reject H0)	
FFD	RF vs. CatBoost	7.0	0.0019	Significant difference in F1 scores (reject H0)	0.9955 ± 0.0028 (RF) 0.9367 ± 0.0008 (CatBoost) 0.4790 ± 0.0012 (LightGBM)
	RF vs. LGBM	11.0	0.0019	Significant difference in F1 scores (reject H0)	
	CatBoost vs. LGBM	5.0	0.0312	Significant difference in F1 scores (reject H0)	
DCCC	RF vs. CatBoost	9.0	0.0020	Significant difference in F1 scores (reject H0)	0.8594 ± 0.0045 (RF) 0.8026 ± 0.0056 (CatBoost) 0.7997 ± 0.0065 (LightGBM)
	RF vs. LGBM	4.0	0.01367	Significant difference in F1 scores (reject H0)	
	CatBoost vs. LGBM	19.0	0.2754	No significant difference in F1 scores (fail to reject H0)	
PW	RF vs. CatBoost	21.0	0.5566	No significant difference in F1 scores (fail to reject H0)	0.9760 ± 0.0030 (RF) 0.9604 ± 0.0050 (CatBoost) 0.9477 ± 0.0062 (LightBoost)
	RF vs. LGBM	9.0	0.0644	No significant difference in F1 scores (fail to reject H0)	
	CatBoost vs. LGBM	4.0	0.0136	Significant difference in F1 scores (reject H0)	
CA	RF vs. CatBoost	12.0	0.2500	No significant difference in F1 scores (fail to reject H0)	0.8926 ± 0.0233 (RF) 0.8660 ± 0.0434 (CatBoost) 0.8251 ± 0.0538 (LightGBM)
	RF vs. LGBM	21.0	0.9101	No significant difference in F1 scores (fail to reject H0)	
	CatBoost vs. LGBM	7.0	0.3071	No significant difference in F1 scores (fail to reject H0)	

Table 13. Comparative analysis of models using the Wilcoxon signed-rank test with paired cross-validation folds, effect size estimation, and bootstrap-based confidence intervals.

Dataset	**Model**	**p-value**	**Wilcoxon _stat**	**Paired Cohen's d**	**Cliff's delta**	**Mean Difference**	**bootstrap 95% CI**
CCFD	RF vs. CatBoost	0.4961	0.000	7.140	1.000	0.0101	[0.0093, 0.0110]
	RF vs. LGBM	0.0273	0.000	0.714	1.000	0.3304	[0.0491, 0.6195]
	CatBoost vs.LGBM	0.1289	0.000	0.691	1.000	0.3203	[0.0380, 0.6100]
VICF	RF vs. CatBoost	0.0058	0.000	8.120	1.000	0.0169	[0.0158, 0.0183]
	RF vs. LGBM	0.0039	0.000	2.975	1.000	0.0599	[0.0490, 0.0722]
	CatBoost vs.LGBM	0.0020	0.000	2.089	1.000	0.0430	[0.0317, 0.0554]
FFD	RF vs. CatBoost	0.0019	0.000	1.113	0.900	0.0616	[0.0292, 0.0970]

Table 13 continued…

	RF vs. LGBM	0.0019	0.000	1.026	0.960	0.5193	[0.2205, 0.8108]
	CatBoost vs.LGBM	0.0312	10.000	0.907	0.400	0.4577	[0.1593, 0.7509]
DCCC	RF vs. CatBoost	0.0020	0.000	5.224	1.000	0.0568	[0.0505, 0.0634]
	RF vs. LGBM	0.01367	0.000	9.219	1.000	0.0597	[0.0558, 0.0633]
	CatBoost vs.LGBM	0.2754	16.000	0.312	0.340	0.0029	[-0.0034, 0.0076]
PW	RF vs. CatBoost	0.5566	0.000	1.766	1.000	0.0121	[0.0080, 0.0163]
	RF vs. LGBM	0.0644	0.000	4.231	1.000	0.0248	[0.0214, 0.0283]
	CatBoost vs.LGBM	0.0136	0.000	3.409	0.920	0.0127	[0.0100, 0.0144]
CA	RF vs. CatBoost	0.2500	7.000	0.502	0.300	0.0151	[-0.0004, 0.0341]
	RF vs. LGBM	0.9101	5.000	0.668	0.380	0.0560	[0.0116, 0.1085]
	CatBoost vs.LGBM	0.3071	11.000	0.550	0.210	0.0409	[0.0038, 0.09001

Table 14. Mean and fold-wise thresholding patterns across 10-fold cross-validation, highlighting model-specific threshold consistency and variability.

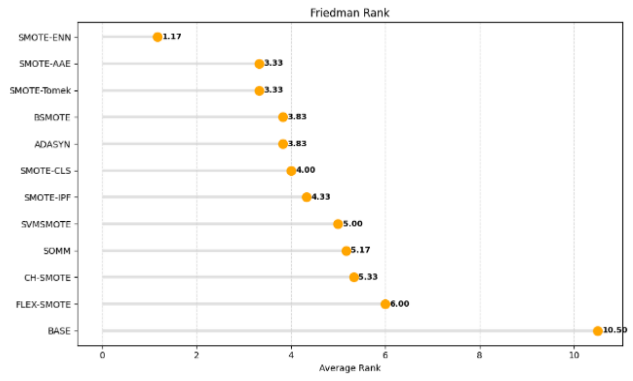
Dataset	Model	Mean threshold	Thresholds per fold
CCFD	RF	0.641	[0.62, 0.71, 0.61, 0.62, 0.64, 0.67, 0.65, 0.65, 0.67, 0.57]
	CatBoost	0.729	[0.69, 0.78, 0.77, 0.74, 0.76, 0.7, 0.7, 0.7, 0.73, 0.72]
	LGBM	0.827	[0.97, 0.75, 0.7, 0.8, 0.92, 0.61, 0.98, 0.94, 0.76, 0.84]
VICF	RF	0.484	[0.5, 0.46, 0.49, 0.47, 0.47, 0.5, 0.53, 0.46, 0.48, 0.48]
	CatBoost	0.368	[0.37, 0.39, 0.34, 0.34, 0.39, 0.34, 0.39, 0.4, 0.41, 0.31]
	LGBM	0.308	[0.37, 0.35, 0.29, 0.27, 0.28, 0.33, 0.31, 0.36, 0.25, 0.27]
FFD	RF	0.682	[0.69, 0.66, 0.64, 0.66, 0.71, 0.73, 0.69, 0.63, 0.65, 0.76]
	CatBoost	0.930	[0.9, 0.89, 0.93, 0.91, 0.96, 0.9, 0.96, 0.96, 0.93, 0.96]
	LGBM	0.911	[0.96, 0.97, 0.79, 0.75, 0.99, 0.9, 0.97, 0.89, 0.96, 0.93]
DCCC	RF	0.443	[0.45, 0.44, 0.44, 0.42, 0.46, 0.46, 0.45, 0.44, 0.44, 0.43]
	CatBoost	0.447	[0.48, 0.45, 0.46, 0.42, 0.44, 0.44, 0.47, 0.44, 0.44, 0.43]
	LGBM	0.458	[0.45, 0.47, 0.45, 0.45, 0.46, 0.46, 0.47, 0.49, 0.44, 0.44]
PW	RF	0.466	[0.43, 0.5, 0.53, 0.45, 0.48, 0.44, 0.47, 0.46, 0.43, 0.47]
	CatBoost	0.526	[0.52, 0.53, 0.56, 0.52, 0.51, 0.56, 0.6, 0.43, 0.5, 0.53]
	LGBM	0.528	[0.49, 0.45, 0.49, 0.52, 0.59, 0.67, 0.62, 0.46, 0.42, 0.57]
CA	RF	0.422	[0.42, 0.39, 0.46, 0.39, 0.41, 0.43, 0.46, 0.54, 0.37, 0.3]
	CatBoost	0.405	[0.25, 0.45, 0.33, 0.25, 0.35, 0.66, 0.45, 0.62, 0.27, 0.42]
	LGBM	0.396	[0.34, 0.84, 0.24, 0.04, 0.45, 0.46, 0.03, 0.74, 0.25, 0.57]

RF is likely to show modest overconfidence, especially in moderately imbalanced datasets like VICF and PW, while LGBM often showed underconfidence predictions in extreme imbalance scenarios like FFD and CA. RF varied more from the optimal calibration, but CatBoost and LGBM were generally able to achieve the best feasible balance between discrimination and calibration. These results indicate that CatBoost's gradient-based optimization improves probabilistic consistency and predictive reliability across a range of fraud detection datasets.

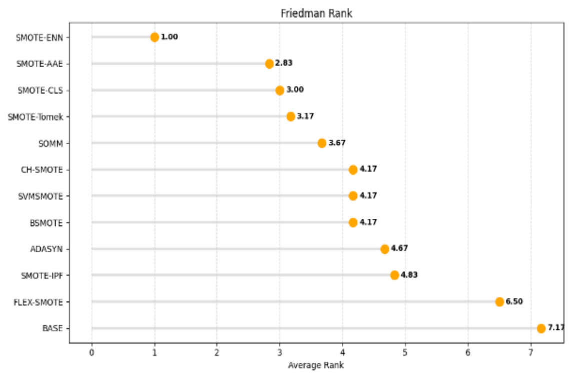
To solve the multiple comparisons issue that occurs when comparing several classifiers pairwise, the Holm–Bonferroni (Holm, 1979) test is used with the Wilcoxon signed-rank test on metrics like ROC-AUC or PR-AUC. Even if the Wilcoxon test determines that two classifier differences are statistically significant, the possibility of false positives rises when it is applied to several pairings. Compared to the normal Bonferroni approach, the Holm-Bonferroni test maintains better statistical power while more successfully controlling the family-wise error rate by progressively changing the p-values. This method guarantees accurate and equitable classifier comparisons without exaggerating their importance. When it comes to classifier differences, the Wilcoxon signed-rank test results (refer to **Table 15** in Appendix A.2) show that the significance varies depending on the dataset and evaluation criteria. In particular, ROC-AUC and PR-AUC values support statistically significant differences in performance between RF and CatBoost, especially for the datasets CCFD, VICF, FFD, DCCC, and CA. The DCCC dataset (ROC-AUC and PR-AUC) and the VICF dataset (PR-AUC only) are the only two datasets that differ statistically from RF and LGBM. On the other hand, CatBoost and LGBM often have comparable results, except for the PW and CA datasets, where CatBoost has a statistically significant edge. According to the Wilcoxon signed-rank test, these results indicate that performance differences are very dataset-dependent, with CatBoost and LGBM typically being more aligned and RF and CatBoost frequently differing the most. Classifiers' performance varied significantly across numerous fraud detection datasets, as shown by the Holm–Bonferroni (refer to **Table 16** in Appendix A.2) adjusted pairwise comparisons. CatBoost considerably outperformed RF for the CCFD dataset, according to both ROC-AUC and PR-AUC, although neither RF nor CatBoost and LGBM showed any discernible differences. The VICF dataset showed mixed results: RF and CatBoost did not substantially differ for ROC-AUC, while RF performed worse for PR-AUC than both CatBoost and LGBM, indicating the sensitivity of PR-AUC in unbalanced settings. RF once more outperformed CatBoost in the FFD dataset in terms of both ROC-AUC and PR-AUC, while CatBoost and LGBM did not differ substantially. RF consistently performed worse than CatBoost and LGBM, whereas CatBoost and LGBM were statistically indistinguishable for the DCCC dataset. In the PW dataset, CatBoost beat LGBM on both ROC-AUC and PR-AUC, although RF did not differ substantially from either of them. Lastly, while ROC-AUC only identified the CatBoost–LGBM contrast as significant, PR-AUC showed substantial differences favoring CatBoost over both RF and LGBM in the CA dataset. Overall, the data show that CatBoost beat RF and occasionally LGBM, whereas RF rarely equaled the performance of the gradient boosting techniques, particularly when evaluated using PR-AUC. **Figures 12(a)-(f)** in Appendix A.3 illustrates the ROC-AUC and PR-AUC distributions across 10-fold cross-validation calculated for the RF, CatBoost, and LGBM classifiers, which are used to further validate the results.

Comparative Performance Analysis:
To address the problems of imbalanced data, we use various techniques, including ADASYN, Borderline-SMOTE (BSMOTE), SMOTE-ENN, SMOTE-Tomek, SVMSMOTE, SMOTE-IPF (Sáez et al., 2015), CH-SMOTE (Yuan et al., 2023), FLEX-SMOTE, and two more recent techniques, SMOTE-CLS (Hong et al., 2025) and SOMM (Khorshidi & Aickelin, 2025). The approaches' performance is compared using four metrics: recall, F1-score, PR-AUC, and ROC-AUC. Using the Friedman rank test (Elamir, 2022), the method's performance is evaluated based on Friedman average rank and dataset-wise classification rank.

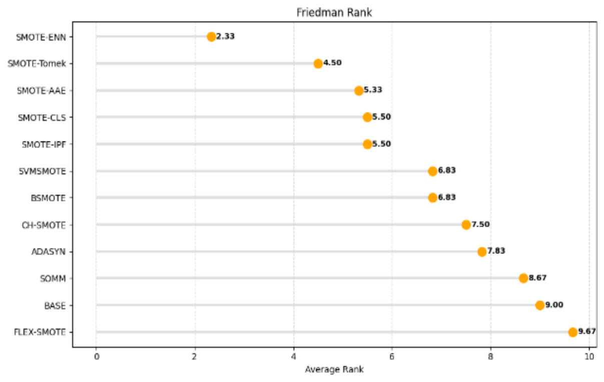
Tables 17-20 (see Appendix A.2) present the comparative performance assessments on recall, F1-score, ROC-AUC, and PR-AUC for the base method and 11 resampling methods. **Table 17** shows that SMOTE-AAE is one of the best oversampling methods, with a strong average rank of 5.33. It is comparable to SMOTE-IPF and SMOTE-CLS. Compared to lower-ranked methods like SOMM, BASE, and FLEX-SMOTE, its performance is substantially better. Although SMOTE-ENN remains the best overall, SMOTE-AAE is a reliable and effective choice for handling imbalanced datasets. The average ranking of 6.83 demonstrates a high degree of performance and consistent performance across datasets. Based on average ranks and dataset-wise ranks, **Table 18** shows that SMOTE-AAE routinely ranks among the best oversampling methods. While SMOTE-ENN remains the best overall, a statistically similar top-tier group consisting of SMOTE-AAE, ADASYN, SMOTE-Tomek, and SMOTE-CLS surpasses less adaptive or conventional methods such as BASE, FLEX-SMOTE, and SOMM. This indicates that SMOTE-AAE is a reliable and effective technique for jobs involving the identification of imbalanced fraud. According to **Table 19**, SMOTE-AAE is among the most effective resampling methods for the datasets under question. It regularly ranks near the top, second only to SMOTE-ENN, and performs better than SMOTE versions such as SMOTE-CLS and SMOTE-IPF. This shows how SMOTE-AAE successfully improves classifier performance on datasets that are unbalanced. According to the Friedman rank analysis, SMOTE-ENN performed best with the lowest average rank (1.0), followed by SMOTE-AAE (2.83) and SMOTE-CLS (3.0) (**Table 20**). These strategies continuously performed better than others, proving to be highly effective in reducing class inequality. BASE (7.17) and FLEX-SMOTE (6.5), on the other hand, produced less impressive outcomes. In general, the most effective oversampling strategies where hybrid SMOTE-based approaches such as SMOTE-ENN, SMOTE-AAE, and SMOTE-CLS. **Figures 13(a)-(d)** also shows the performance of different oversampling techniques used with Random Forest, based on the Recall, F1-score, ROC-AUC, and PR-AUC metrics, respectively. The ranking analysis shows how well each oversampling strategy performs overall across a range of metrics. In increasing RF's classification capability on imbalanced data sets, approaches with lower average ranks work better and more reliably. By providing an in-depth understanding of method performance that goes beyond the assessment of individual metrices, this analysis helps in selecting the most holistic and successful oversampling approach. According to Friedman's average rank, **Figures 13(a)-(d)** further show that, among 10 oversampling methods and BASE, the SMOTE-AAE ranks third in recall, fifth in F1-score, second in ROC-AUC, and second also in PR-AUC, respectively.



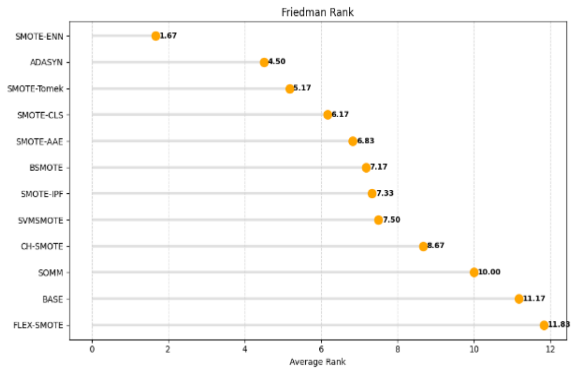
(a)



(b)



(c)



(d)

Figure 13. The performance analysis (based on the Friedman average rank) of oversampling methods used with Random Forest across the four metrics is as follows: (a) Recall, (b) F1-score, (c) ROC-AUC, and (d) PR-AUC.

7. Conclusion

This research aims to identify instances of financial fraud in situations involving imbalanced, large, complex, class-overlapped, and overfitted data. This study introduced a new SMOTE-AAE-RF method to address these challenges. Firstly, we measured and reduced the data difficulty (data complexity) of the data sets using various complexity measures discussed in this paper. We performed this measurement and reduction on three variants of each dataset: the original, resampled by the SMOTE, and encoded by the SMOTE-AAE method. The original CCFD dataset has received a difficulty score of 32.4, 15.4, and 13.5 for the original, resampled, and encoded datasets, respectively. Similarly, we performed the analysis on a dataset for detecting fraud in vehicle insurance claims and obtained scores of 53.6, 42.5, and 41.1 for the original, resampled, and encoded datasets, respectively. We also tested the effectiveness of the proposed method on other datasets listed in **Table 5**. Our SMOTE-AAE approach is more successful in reducing the problem's difficulty in the datasets, according to the results. Next, we conducted tests to evaluate the effectiveness of the SMOTE-AAE-RF method in detecting financial fraud in datasets with uneven balance. In this approach, SMOTE is used to class-balance the dataset by creating synthetic minority class examples. We refine the samples generated by the SMOTE using the AAE method. We then trained and tested the RF classifier on these data sets to identify the fraudulent cases. The results indicate that the SMOTE-AAE-RF method performed better than all the other methods we tested, achieving an accuracy of 0.9999, precision of 0.9998, recall of 1.0000, F1-score of 0.9999, MCC of 0.9996, Kappa of 0.9996, and ROC-AUC and PR-AUC of 1.0000 for the CCFD dataset that had an extremely imbalanced distribution. We also ran tests using the dataset for spotting fraud in vehicle insurance claims, and we got 0.9773 accuracy, 0.9992 precision, 0.9559 recall, 0.9762 F1-score, 0.9374 MCC, 0.9356 Kappa, 0.9961 AUC-ROC, and 0.9967 PR-AUC. The results indicate that our method outperformed the others in terms of PR-AUC and ROC-AUC metrics for detecting fraud in vehicle insurance claims. The superiority of the suggested method is also tested on different datasets with varying degrees of imbalance ratio, such as the DCCC, FFD, PW, and CA. All datasets produce ROC-AUC and PR-AUC scores that exceed 0.9400. The ablation study also validated these results. We also compared the computational complexity of the SMOTE-AAE-RF method with that of the RAW+RF, SMOTE+RF, and AAE+RF methods, and the results indicated that it performed better than these methods. We also conducted statistical performance tests (McNemar's test and Wilcoxon signed-rank test at $\alpha = 0.05$) on the suggested method using CatBoost and LightGBM as baseline classifiers. The results also indicate that the suggested method outperformed others and demonstrate that the computational efficiency and scalability of the SMOTE-AAE-RF approach make it a practically viable enhancement for imbalanced learning scenarios. In addition to demonstrating the SMOTE-AAE-RF method's superiority over ten other oversampling techniques in terms of recall, F1-score, ROC-AUC, and PR-AUC, the Friedman rank test also indicated that it attained the second-best average rank.

We have observed the following limitations in scenarios involving extremely high-dimensional feature spaces, the AAE may require careful architectural tuning or dimensionality reduction (e.g., via PCA or feature selection) to maintain stability and effectiveness. Additionally, for streaming or real-time data applications, the batch-oriented nature of AAE poses a challenge. In future work, we will be focused on the insurance fraud and incorporate:

1) Alternatives to the AAE, such as online learning or incremental learning, can be useful solutions.
2) By eliminating the possibility of data leaking between train and test sets, customer-level grouping—in which all transactions from the same customer are grouped into a single fold—provides a more realistic assessment environment.
3) Introduce a complexity-aware version of SMOTE, such as one that weights the likelihood of creating a synthetic sample based on neighborhood- or instance-level complexity indices.

Appendix

A.1 Pseudocode

Algorithm 2: SMOTE–AAE

```
Input: Imbalanced dataset $D_{imb} = \{X, y\}$.
Output: Filtered balanced dataset $D_{filtered} = \{X_{filtered}, y_{filtered}\}$.
Method:
// Data Balancing (SMOTE).
1) Apply SMOTE to balance classes:
                                   $(X_{res}, y_{res}) \leftarrow$ SMOTE.fit_resample$(X, y)$
2) Convert $X_{res}, y_{res}$ into PyTorch tensors.
   // Define Adversarial Autoencoder (AAE).
3) Initialize Encoder $E$: maps $X \rightarrow Z$
4) Initialize Decoder $D$: reconstructs $Z \rightarrow \hat{X}$
5) Initialize Discriminator $Disc$: distinguishes real vs. fake latent vectors.
// Train AAE.
   (i) For each epoch do:
         a. For each mini-batch $x$ in $X_{res}$:
            i. Discriminator update:
               - Sample $z_{real} \sim \mathcal{N}(0, I)$
               - Compute $z_{fake} = E(x)$
               - Update $Disc$ to minimize:
                 $L_D = BCE(Disc(z_{real}),1) + BCE(Disc(z_{fake}),0)$
            ii. Autoencoder update:
               - Reconstruct $\hat{x} = D(E(x))$
               - Compute losses:
                 Reconstruction: $L_{rec} = MSE(\hat{x}, x)$
                 Adversarial: $L_{adv} = BCE(Disc(E(x)),1)$
               - Total loss: $L_{AE} = L_{rec} + L_{adv}$
               - Update $E, D$ using $L_{AE}$.
// Filtering Score Computation
6) For each sample $x_i$ in $X_{res}$:
      a. Encode $z_i = E(x_i)$, reconstruct $\hat{x}_i = D(z_i)$
      b. Compute:
         - Reconstruction error: $RE_i = || x_i - \hat{x}_i ||^2$
         - Discriminator score: $DS_i = Disc(z_i)$
         - Log-likelihood: $LL_i = -0.5 \; || z_i ||^2 - \frac{d}{2} \log(2\pi)$
         - Density via KDE: $DEN_i =$ KDE$(z_i)$
7) Compute composite filtering score:
      $S_i = \alpha \cdot RE_i - \beta \cdot LL_i - \gamma \cdot \log(DS_i + \varepsilon) + \delta \cdot DEN_i$
// Sample Selection.
8) Determine threshold $\tau = P_{80}(S)$(80th percentile).
9) Keep samples where $S_i \leq \tau$.
// Output.
10)     $X_{filtered}, y_{filtered} \leftarrow (X_{res}, y_{res})$for samples satisfying $S_i \leq \tau$.
11)     Return $D_{filtered} = \{X_{filtered}, y_{filtered}\}$.
```

The SMOTE-AAE algorithm begins by addressing class imbalance in the dataset $D_{imb} = \{X, y\}$. First, the SMOTE procedure (Algorithm 1) is applied to generate synthetic minority samples, resulting in a balanced dataset (X_{res}, y_{res}), which is then converted into PyTorch tensors for model training. Next, an Adversarial Autoencoder (AAE) framework is constructed, consisting of an encoder E that maps input samples to a latent representation Z, a decoder D that reconstruct samples from the latent space, and a discriminator Disc that distinguishes real latent vectors sampled from a Gaussian prior from fake ones produced by the encoder. During training, the model iterates over multiple epochs, and for each mini-batch, the discriminator Disc is first updated by comparing real latent vectors drawn from a standard normal distribution with the encoder-generated latent vectors, optimizing a binary cross-entropy loss. Then, the autoencoder is updated by minimizing a combination of reconstruction loss (between original and reconstructed samples) and adversarial loss which encourages encoded samples to match the prior distribution), refining both encoder and decoder simultaneously. After training the AAE, each sample in X_{res} is evaluated using several statistical and model-based measures. The sample is encoded into $z_i = E(x_i)$ and reconstructed as $\hat{x_i} = D(z_i)$, from which the reconstruction error is computed. Additional metrics include the discriminator score, log-likelihood under the Gaussian prior, and latent-space density estimated through kernel density estimation (KDE). These components are combined into a composite filtering score S_i, weighted by parameters and α, β, γ, and δ. To filter out low-quality or noisy samples, an 80th – percentile threshold τ is calculated, and only samples with $S_i \leq \tau$ are retained. The final output is the filtered and balanced dataset $D_{filtered} = \{X_{filtered}, y_{filtered}\}$, which contains only the high-quality resampled data selected through the SMOTE-AAE method.

A.2 Additional Results

Table 6. The RF classifier's performance on different datasets resampled using the SMOTE-AAE technique.

Datasets	Accuracy	Precision	Recall	F1 score	MCC	Kappa	AUC-ROC	PR-AUC
CCFD	0.9999 ± 0.0000	0.9998 ± 0.0001	1.0000 ± 0.0000	0.9999 ± 0.0000	0.9996 ± 0.0001	0.9996 ± 0.0001	1.0000 ± 0.0000	1.0000 ± 0.0000
VICF	0.9773 ± 0.0008	0.9992 ± 0.0006	0.9559 ± 0.0063	0.9762 ± 0.0008	0.9374 ± 0.0051	0.9356 ± 0.0054	0.9961 ± 0.0008	0.9967 ± 0.0006
FFD	0.9707 ± 0.0036	1.0000 ± 0.0000	0.9903 ± 0.0074	0.9955 ± 0.0028	0.9899 ± 0.0042	0.9898 ± 0.0042	1.0000 ± 0.0000	1.0000 ± 0.0000
DCCC	0.8786 ± 0.0040	0.8816 ± 0.0036	0.8934 ± 0.0069	0.8961 ± 0.0045	0.7727 ± 0.0086	0.7664 ± 0.0095	0.9575 ± 0.0027	0.9543 ± 0.0033
PW	0.9756 ± 0.0051	0.9734 ± 0.0030	0.9823 ± 0.0062	0.9760 ± 0.0050	0.9514 ± 0.0093	0.9513 ± 0.0094	0.9965 ± 0.0006	0.9956 ± 0.0008
CA	0.8961 ± 0.0227	0.8998 ± 0.0255	0.8858 ± 0.0250	0.8926 ± 0.0233	0.8926 ± 0.0889	0.8874 ± 0.0885	0.9512 ± 0.0167	0.9472 ± 0.0192

Table 7. The various approaches to CCFD are examined in a comparative way.

References	Model	Accuracy	Precision	Recall	F1 score	MCC	AUC
Ileberi et al. (2021)	ET-AdaBoost	0.9999	0.9993	1.000	-	0.9900	-
Esenogho et al. (2022)	DL Ensemble Model	0.9900	-	0.9960	-	-	-
Sisodia & Sisodia (2022)	QDPSKNN	-	0.8360	0.7890	0.8190	-	-
Ileberi et al. (2022)	GA-DT	1.0000	-	-	-	-	1.0000
Alarfaj et al. (2022)	CNN	0.9990	0.9300	-	0.8571	-	0.9800
Mienye & Sun (2023)	Stacking-based DL Ensemble	0.9999	0.9993	1.000	-	0.9900	-
Du et al. (2024)	AE-XGB-SMOTE-CGAN	0.9993	-	-	-	0.8848	-
Sarker et al. (2024)	RF	0.9996	0.9743	-	0.8636	0.8636	-
Khalid et al. (2024)	PM (SMOTE)	0.9996	0.9996	0.9996	0.9996	-	-
Salam et al. (2024)	SMOTE-CNN	0.9993	0.959	0.8095	0.8178	-	0.9378
Gupta et al. (2025)	ML-based Fraud Detection	0.9995	0.9993	0.9997	0.9995	-	1.0000
Proposed Method	SMOTE-AAE-RF	0.9999	0.9998	1.0000	0.9999	0.9996	1.0000

As shown in **Table 6**, the results were obtained by applying eight evaluation metrics to an RF classifier trained on resampled balanced data sets from SMOTE-AAE. The findings are presented as a mean ± standard deviation.

Table 7 lists eleven current techniques for predicting credit card fraud, along with our suggested approach. We present the findings on average and compared them using the six evaluation measures. In every evaluation metric, our suggested approach yields the best outcomes.

Table 8 presents the average results of a comparison of our suggested method's performance with other recent methods on eight evaluation measures used to predict fraud in vehicle insurance claims.

Table 8. Compare how well the suggested approach performs with the most advanced methods for detecting vehicle insurance claim fraud.

Method	**Accuracy**	**Precision**	**Recall**	**F1 score**	**MCC**	**Kappa**	**AUC-ROC**	**PR-AUC**
Nordin et al. (2024)	0.7935	-	0.4470	-			0.8100	
Ming et al. (2024)	-	-	-	-	-	-	0.8954	0.3322
Combert et al. (2025)	0.9149	0.9362	0.8980	0.9167	-	0.8298	-	-
Gheysarbeigi et al. (2025)	0.9994	0.9893	1.0000	0.9940	-	-	-	-
Proposed method	0.9773	0.9992	0.9559	0.9762	0.9374	0.9356	0.9961	0.9967

To compare the statistical performance of RF, CatBoost, and LGBM on the data sets resampled using the SMOTE-AAE approach, **Tables 15-16** present the findings of the Wilcoxon signed-rank test with the Holm-Bonferroni test on the ROC-AUC and PR-AUC evaluation metrics. The p-value falls between 0 and 1.

Table 15. Wilcoxon signed-rank test (at $\alpha = 0.05$) for the resampled datasets using the SMOTE-AAE technique.

Datasets	**Classifiers**	**ROC-AUC**		**PR-AUC**	
		p-value	Difference	p -value	Difference
CCFD	RF vs CatBoost	0.0039	Significant difference	0.0020	Significant difference
	RF vs LGBM	1.0000	No significant difference	0.6953	No significant difference
	CatBoost vs LGBM	0.1309	No significant difference	0.1602	No significant difference
VICF	RF vs CatBoost	0.0195	Significant difference	0.0020	Significant difference
	RF vs LGBM	0.6250	No significant difference	0.0137	Significant difference
	CatBoost vs LGBM	0.1055	No significant difference	0.1309	No significant difference
FFD	RF vs CatBoost	0.0156	Significant difference	0.0156	Significant difference
	RF vs LGBM	0.8125	No significant difference	0.7148	No significant difference
	CatBoost vs LGBM	0.2500	No significant difference	0.2188	No significant difference
DCCC	RF vs CatBoost	0.0020	Significant difference	0.0020	Significant difference
	RF vs LGBM	0.0020	Significant difference	0.0020	Significant difference
	CatBoost vs LGBM	0.0840	No significant difference	0.0645	No significant difference
PW	RF vs CatBoost	0.3223	No significant difference	0.3223	No significant difference
	RF vs LGBM	0.6953	No significant difference	0.6953	No significant difference
	CatBoost vs LGBM	0.0020	Significant difference	0.0020	Significant difference
CA	RF vs CatBoost	0.0273	Significant difference	0.0195	Significant difference
	RF vs LGBM	0.3652	No significant difference	0.4922	No significant difference
	CatBoost vs LGBM	0.0039	Significant difference	0.0020	Significant difference

Table 16. Results of the Holm–Bonferroni test for the resampled datasets from SMOTE-AAE technique.

Datasets	Classifiers	ROC-AUC (Adjusted)			PR-AUC (Adjusted)		
		p-value (Raw)	Threshold	Reject	p-value (Raw)	Threshold	Reject
CCFD	RF vs CatBoost	0.0039	0.0167	True	0.0020	0.0167	True
	RF vs LGBM	1.0000	0.0500	False	0.6953	0.0500	False
	CatBoost vs LGBM	0.1309	0.0250	False	0.1602	0.0250	False
VICF	RF vs CatBoost	0.0195	0.0167	False	0.0020	0.0167	True
	RF vs LGBM	0.6250	0.0500	False	0.0137	0.0250	True
	CatBoost vs LGBM	0.1055	0.0250	False	0.1309	0.0500	False
FFD	RF vs CatBoost	0.0156	0.0167	True	0.0156	0.0167	True
	RF vs LGBM	0.8125	0.0500	False	0.7148	0.0500	False
	CatBoost vs LGBM	0.2500	0.0250	False	0.2188	0.0250	False
DCCC	RF vs CatBoost	0.0020	0.0167	True	0.0020	0.0167	True
	RF vs LGBM	0.0020	0.0250	True	0.0020	0.0250	True
	CatBoost vs LGBM	0.0840	0.0500	False	0.0645	0.0500	False
PW	RF vs CatBoost	0.3223	0.0250	False	0.3223	0.0250	False
	RF vs LGBM	0.6953	0.0500	False	0.6953	0.0500	False
	CatBoost vs LGBM	0.0020	0.0167	True	0.0020	0.0167	True
CA	RF vs CatBoost	0.0273	0.0250	False	0.0195	0.0250	True
	RF vs LGBM	0.3652	0.0500	False	0.4922	0.0500	False
	CatBoost vs LGBM	0.0039	0.0167	True	0.0020	0.0167	True

Table 17. Comparison of oversampling methods based on recall (mean ± SD) using RF classifier.

Dataset	**BASE**	**SMOTE-AAE**	**SMOTE-CLS**	**SOMM**	**CH-SMOTE**	**FLEX-SMOTE**
CCFD	0.7662 ± 0.0293 (12)	1.0000 ± 0.0000 (1)	1.0000 ± 0.0000 (1)	0.9997 ± 0.0001 (5)	0.9996 ± 0.0001 (6)	0.9997 ± 0.0001 (5)
VICF	0.0087 ± 0.0081 (12)	0.9559 ± 0.0063 (7)	0.9591 ± 0.0039 (5)	0.9373 ± 0.0058 (11)	0.9376 ± 0.0055 (10)	0.9472 ± 0.0073 (9)
FFD	1.0000 ± 0.0000 (2)	0.9903 ± 0.0074 (11)	1.0000 ± 0.0000 (2)	0.8747 ± 0.0667 (12)	1.0000 ± 0.0000 (2)	0.9360 ± 0.0085 (10)
DCCC	0.3379 ± 0.0155 (12)	0.8934 ± 0.0069 (4)	0.8613 ± 0.0065 (8)	0.8239 ± 0.0081 (11)	0.8220 ± 0.0078 (12)	0.7774 ± 0.0145 (12)
PW	0.9614 ± 0.0069 (12)	0.9823 ± 0.0062 (4)	0.9706 ± 0.0048 (8)	0.9807 ± 0.0035 (5)	0.9703 ± 0.0069 (9)	0.9672 ± 0.0074 (10)
CA	0.8877 ± 0.0708 (4)	0.8858 ± 0.0250 (4)	0.8746 ± 0.0533 (9)	0.8747 ± 0.0667 (8)	0.8799 ± 0.0608 (6)	0.8616 ± 0.0689 (11)
Avg. Rank	**9.00**	**5.33**	**5.50**	**8.67**	**7.5**	**9.67**
Dataset	**SMOTE-IPF**	**SVMSMOTE**	**SMOTE-ENN**	**SMOTE-Tomek**	**BSMOTE**	**ADASYN**
CCFD	1.0000 ± 0.0000 (1)	0.9998 ± 0.0001 (4)	1.0000 ± 0.0000 (1)	1.0000 ± 0.0000 (1)	0.9998 ± 0.0001 (4)	1.0000 ± 0.0000 (1)
VICF	0.9575 ± 0.0043 (6)	0.9526 ± 0.0048 (8)	0.9783 ± 0.0028 (1)	0.9558 ± 0.0046 (7)	0.9547 ± 0.0051 (9)	0.9552 ± 0.0045 (8)
FFD	1.0000 ± 0.0000 (2)	1.0000 ± 0.0000 (2)	0.9990 ± 0.0022 (9)	1.0000 ± 0.0000 (2)	1.0000 ± 0.0000 (2)	1.0000 ± 0.0000 (2)
DCCC	0.8322 ± 0.0063 (10)	0.8458 ± 0.0097 (9)	0.9405 ± 0.0044 (1)	0.8526 ± 0.0051 (8)	0.8749 ± 0.0064 (6)	0.8624 ± 0.0069 (7)
PW	0.9784 ± 0.0036 (7)	0.9779 ± 0.0032 (8)	0.9987 ± 0.0018 (1)	0.9818 ± 0.0032 (6)	0.9740 ± 0.0032 (11)	0.9781 ± 0.0063 (9)
CA	0.8798 ± 0.0550 (7)	0.8694 ± 0.0583 (10)	0.9680 ± 0.0392 (1)	0.8936 ± 0.0299 (3)	0.8746 ± 0.0490 (9)	0.8719 ± 0.0547 (10)
Avg. Rank	**5.50**	**6.83**	**2.33**	**4.50**	**6.83**	**7.83**

Friedman rank is given in brackets. Here, rank 1 = best.

Table 18. Comparison of oversampling methods based on F1-score (mean ± SD) using RF classifier.

Dataset	BASE	SMOTE-AAE	SMOTE-CLS	SOMM	CH-SMOTE	FLEX-SMOTE
CCFD	0.8520 ± 0.0300 (12)	0.9999 ± 0.0000 (4)	0.9999 ± 0.0000 (4)	0.9998 ± 0.0001 (8)	0.9998 ± 0.0001 (8)	0.9998 ± 0.0001 (8)
VICF	0.0171 ± 0.0159 (12)	0.9762 ± 0.0008 (7)	0.9773 ± 0.0022 (6)	0.9675 ± 0.0032 (11)	0.9677 ± 0.0030 (10)	0.9180 ± 0.0034 (11)
FFD	0.9910 ± 0.0041 (11)	0.9955 ± 0.0028 (5)	0.9971 ± 0.0020 (4)	0.8822 ± 0.0309 (12)	0.9933 ± 0.0024 (6)	0.9245 ± 0.0053 (11)
DCCC	0.4495 ± 0.0156 (12)	0.8961 ± 0.0045 (5)	0.8762 ± 0.0050 (6)	0.8753 ± 0.0061 (7)	0.8747 ± 0.0055 (8)	0.7803 ± 0.0103 (11)
PW	0.9682 ± 0.0036 (12)	0.9760 ± 0.0050 8)	0.9760 ± 0.0047 (8)	0.9755 ± 0.0039 (10)	0.9755 ± 0.0042 (10)	0.9726 ± 0.0044 (12)
CA	0.8885 ± 0.0515 (7)	0.8926 ± 0.0233 (5)	0.8869 ± 0.0260 (9)	0.8822 ± 0.0309 (11)	0.8841 ± 0.0284 (10)	0.8755 ± 0.0298 (12)
Avg. Rank	**11.17**	**6.83**	**6.17**	**10.00**	**8.67**	**11.83**
Dataset	**SMOTE-IPF**	**SVMSMOTE**	**SMOTE-ENN**	**SMOTE-Tomek**	**BSMOTE**	**ADASYN**
CCFD	0.9999 ± 0.0000 (4)	0.9998 ± 0.0001 (8)	1.0000 ± 0.0000 (1)	0.9999 ± 0.0001 (4)	0.9998 ± 0.0001 (8)	0.9999 ± 0.0001 (4)
VICF	0.9570 ± 0.0035 (9)	0.9743 ± 0.0031 (5)	0.9806 ± 0.0020 (1)	0.9747 ± 0.0028 (4)	0.9743 ± 0.0036 (5)	0.9747 ± 0.0024 (4)
FFD	0.9959 ± 0.0023 (8)	0.9960 ± 0.0020 (7)	0.9964 ± 0.0025 (5)	0.9965 ± 0.0020 (3)	0.9964 ± 0.0015 (5)	0.9978 ± 0.0016 (1)
DCCC	0.8481 ± 0.0050 (10)	0.8672 ± 0.0073 (9)	0.9423 ± 0.0045 (1)	0.8713 ± 0.0049 (11)	0.8752 ± 0.0058 (7)	0.8698 ± 0.0061 (9)
PW	0.9754 ± 0.0041 (9)	0.9760 ± 0.0044 (8)	0.9964 ± 0.0021 (1)	0.9767 ± 0.0022 (7)	0.9741 ± 0.0046 (12)	0.9763 ± 0.0059 (6)
CA	0.8925 ± 0.0262 (4)	0.8865 ± 0.0368 (8)	0.9814 ± 0.0218 (1)	0.9102 ± 0.0220 (2)	0.8908 ± 0.0389 (6)	0.8853 ± 0.0368 (3)
Avg. Rank	**7.33**	**7.50**	**1.67**	**5.17**	**7.17**	**4.50**

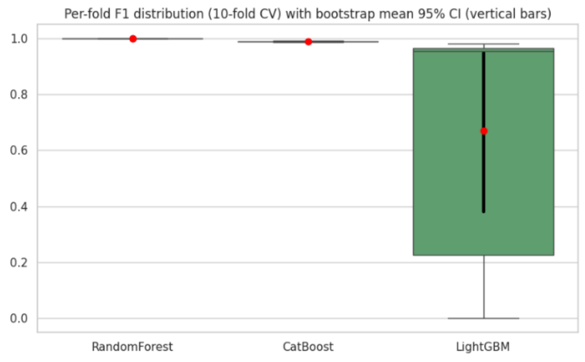
Table 19. Comparison of oversampling methods based on ROC-AUC (mean ± SD) using RF classifier.

Dataset	BASE	SMOTE-AAE	SMOTE-CLS	SOMM	CH-SMOTE	FLEX-SMOTE
CCFD	0.9520 ± 0.0126 (12)	1.0000 ± 0.0000 (1)	1.0000 ± 0.0000 (1)	0.9999 ± 0.0000 (4)	0.9999 ± 0.0000 (4)	0.9999 ± 0.0000 (4)
VICF	0.8607 ± 0.0098 (12)	0.9961 ± 0.0008 (6)	0.9968 ± 0.0007 (4)	0.9905 ± 0.0011 (10)	0.9906 ± 0.0012 (9)	0.9359 ± 0.0053 (11)
FFD	0.9997 ± 0.0006 (6)	1.0000 ± 0.0000 (1)	1.0000 ± 0.0000 (1)	0.9485 ± 0.0214 (12)	0.9999 ± 0.0001 (3)	0.9671 ± 0.0043 10)
DCCC	0.7694 ± 0.0085 (12)	0.9575 ± 0.0027 (3)	0.9465 ± 0.0029 (5)	0.9342 ± 0.0051 (6)	0.9338 ± 0.0050 (7)	0.8360 ± 0.0092 (11)
PW	0.9960 ± 0.0014 (6)	0.9965 ± 0.0006 (4)	0.9967 ± 0.0013 (3)	0.9967 ± 0.0009 (3)	0.9969 ± 0.0007 (1)	0.9964 ± 0.0011 (5)
CA	0.9311 ± 0.0480 (12)	0.9512 ± 0.0167 (5)	0.9442 ± 0.0184 (6)	0.9485 ± 0.0214 (5)	0.9477 ± 0.0194 (6)	0.9362 ± 0.0180 (8)
Avg. Rank	**10.50**	**3.33**	**4.00**	**5.17**	**5.33**	**6.0**
Dataset	**SMOTE-IPF**	**SVMSMOTE**	**SMOTE-ENN**	**SMOTE-Tomek**	**BSMOTE**	**ADASYN**
CCFD	1.0000 ± 0.0000 (1)	0.9999 ± 0.0000 (4)	1.0000 ± 0.0000 (1)	1.0000 ± 0.0000 (1)	0.9999 ± 0.0000 (4)	1.0000 ± 0.0000 (1)
VICF	0.9911 ± 0.0013 (8)	0.9948 ± 0.0010 (7)	0.9976 ± 0.0007 (2)	0.9964 ± 0.0010 (5)	0.9950 ± 0.0011 (3)	0.9964 ± 0.0007 (5)
FFD	1.0000 ± 0.0001 (1)	0.9999 ± 0.0001 (3)	1.0000 ± 0.0001 (1)	1.0000 ± 0.0001 (1)	1.0000 ± 0.0001 (1)	1.0000 ± 0.0000 (1)
DCCC	0.9255 ± 0.0033 (8)	0.9367 ± 0.0044 (4)	0.9781 ± 0.0030 (1)	0.9420 ± 0.0029 (5)	0.9403 ± 0.0042 (6)	0.9387 ± 0.0043 (5)
PW	0.9968 ± 0.0008 (2)	0.9966 ± 0.0010 (3)	0.9999 ± 0.0001 (1)	0.9968 ± 0.0010 (2)	0.9965 ± 0.0009 (4)	0.9963 ± 0.0020 (6)
CA	0.9482 ± 0.0200 (5)	0.9457 ± 0.0203 (6)	0.9985 ± 0.0024 (1)	0.9716 ± 0.0200 (2)	0.9470 ± 0.0220 (6)	0.9373 ± 0.0330 (8)
Avg. Rank	**4.33**	**5.00**	**1.17**	**3.33**	**3.83**	**3.83**

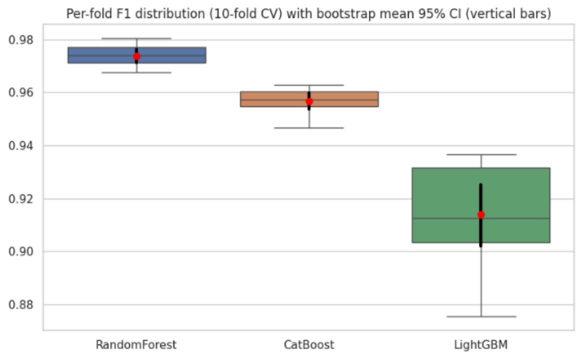
Table 20. Comparison of oversampling methods based on PR-AUC (mean ± SD) using RF classifier.

Dataset	**BASE**	**SMOTE-AAE**	**SMOTE-CLS**	**SOMM**	**CH-SMOTE**	**FLEX-SMOTE**
CCFD	0.8523 ± 0.0387 (3)	1.0000 ± 0.0000 (1)	1.0000 ± 0.0000 (1)	0.9999 ± 0.0000 (2)	0.9999 ± 0.0000 (2)	0.9999 ± 0.0000 (2)
VICF	0.3015 ± 0.0339 (12)	0.9967 ± 0.0006 (4)	0.9973 ± 0.0005 (3)	0.9922 ± 0.0009 (8)	0.9922 ± 0.0009 (8)	0.8953 ± 0.0103 (2)
FFD	0.9999 ± 0.0002 (2)	1.0000 ± 0.0000 (1)	1.0000 ± 0.0000 (1)	0.9405 ± 0.0276 (4)	0.9999 ± 0.0001 (2)	0.9618 ± 0.0050 (3)
DCCC	0.5411 ± 0.0154 (12)	0.9543 ± 0.0033 (2)	0.9452 ± 0.0046 (5)	0.9516 ± 0.0035 (3)	0.9513 ± 0.0035 (4)	0.8387 ± 0.0091 (11)
PW	0.9954 ± 0.0015 (6)	0.9956 ± 0.0008 (5)	0.9969 ± 0.0014 (2)	0.9960 ± 0.0016 (4)	0.9969 ± 0.0006 (2)	0.9960 ± 0.0013 (4)
CA	0.9307 ± 0.0592 (8)	0.9472 ± 0.0192 (3)	0.9357 ± 0.0293 (6)	0.9405 ± 0.0276 (5)	0.9401 ± 0.0275 (5)	0.9188 ± 0.0215 (9)
Avg. Rank	**7.17**	**2.83**	**3.00**	**3.67**	**4.17**	**6.5**
Dataset	**SMOTE-IPF**	**SVMSMOTE**	**SMOTE-ENN**	**SMOTE-Tomek**	**BSMOTE**	**ADASYN**
CCFD	1.0000 ± 0.0000 (1)	0.9999 ± 0.0000 (2)	1.0000 ± 0.0000 (1)	1.0000 ± 0.0000 (1)	0.9999 ± 0.0000 (2)	1.0000 ± 0.0000 (1)
VICF	0.9920 ± 0.0012 (10)	0.9958 ± 0.0007 (7)	0.9984 ± 0.0004 (1)	0.9969 ± 0.0008 (2)	0.9960 ± 0.0008 (6)	0.9968 ± 0.0006 (5)
FFD	1.0000 ± 0.0001 (1)	0.9999 ± 0.0001 (2)	0.9999 ± 0.0001 (2)	1.0000 ± 0.0001 (1)	1.0000 ± 0.0001 (1)	1.0000 ± 0.0000 (1)
DCCC	0.9313 ± 0.0036 (10)	0.9420 ± 0.0041 (6)	0.9853 ± 0.0023 (1)	0.9411 ± 0.0032 (8)	0.9413 ± 0.0049 (7)	0.9326 ± 0.0055 (9)
PW	0.9963 ± 0.0012 (3)	0.9960 ± 0.0016 (4)	0.9999 ± 0.0001 (1)	0.9961 ± 0.0015 (4)	0.9959 ± 0.0015 (6)	0.9958 ± 0.0020 (6)
CA	0.9418 ± 0.0267 (4)	0.9418 ± 0.0255 (4)	0.9987 ± 0.0021 (1)	0.9710 ± 0.0222 (2)	0.9475 ± 0.0246 (3)	0.9348 ± 0.0364 (7)
Avg. Rank	**4.83**	**4.17**	**1.00**	**3.17**	**4.17**	**4.67**

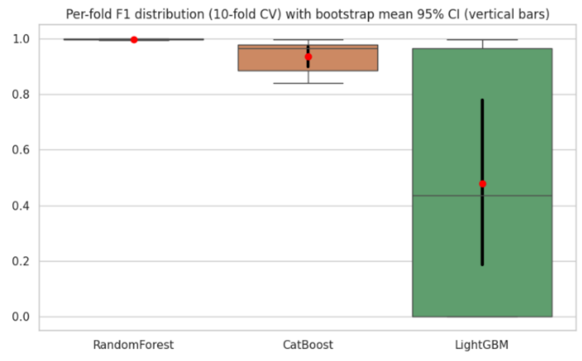
A.3 Results of Stratified 10-Fold Cross-Validation



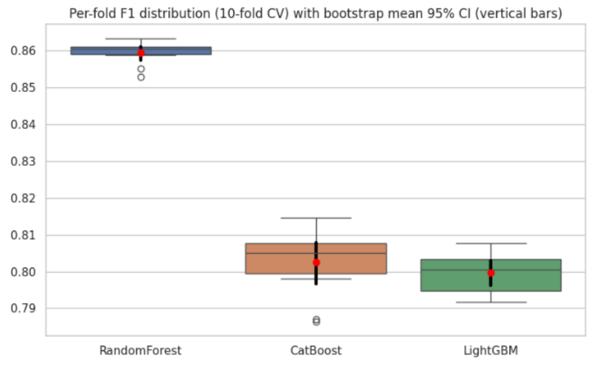
(a)



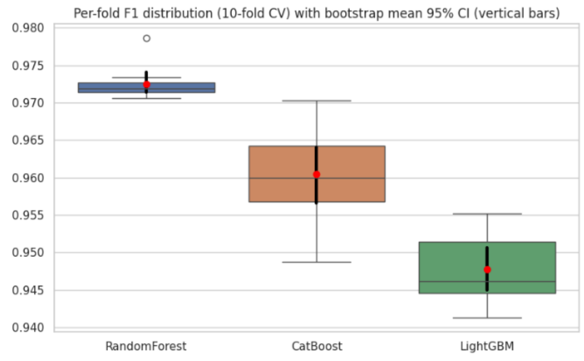
(b)



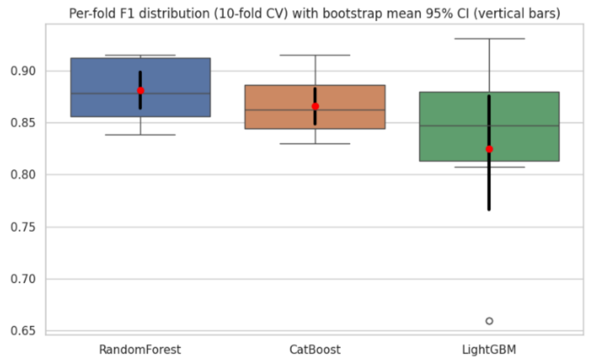
(c)



(d)

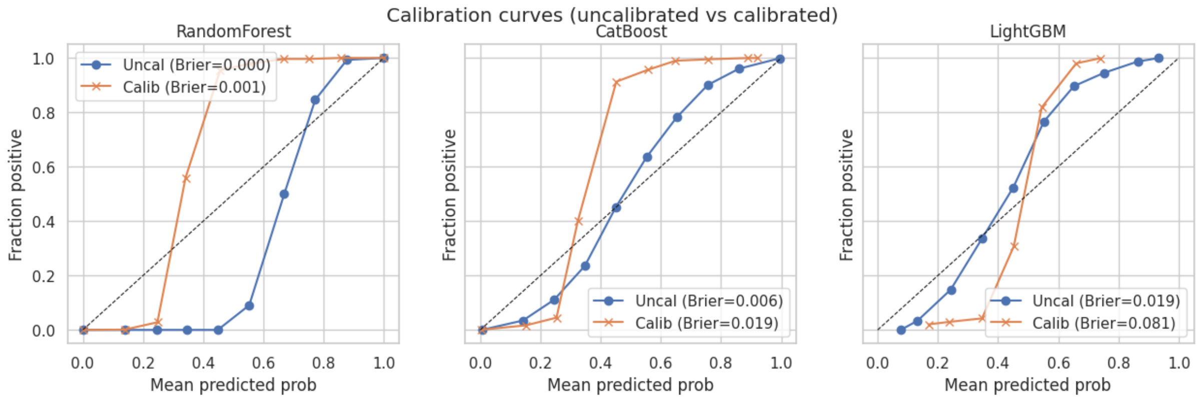


(e)

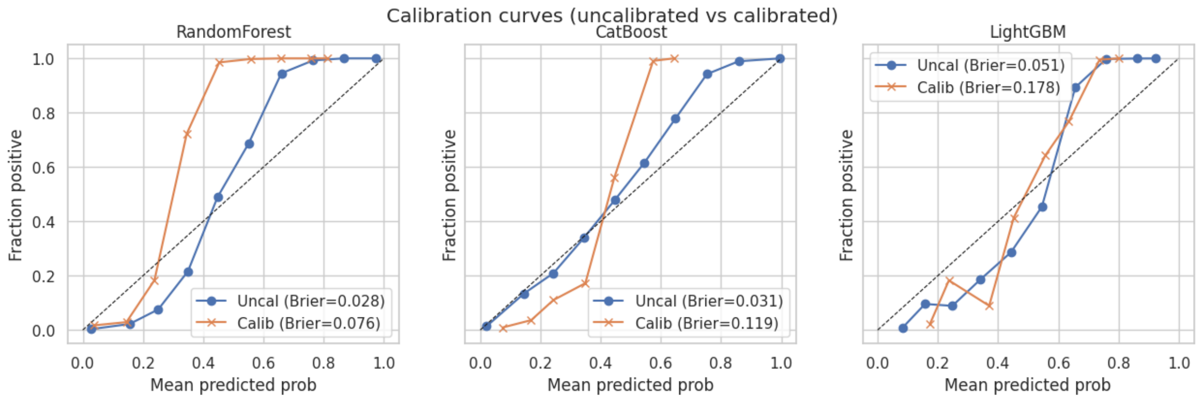


(f)

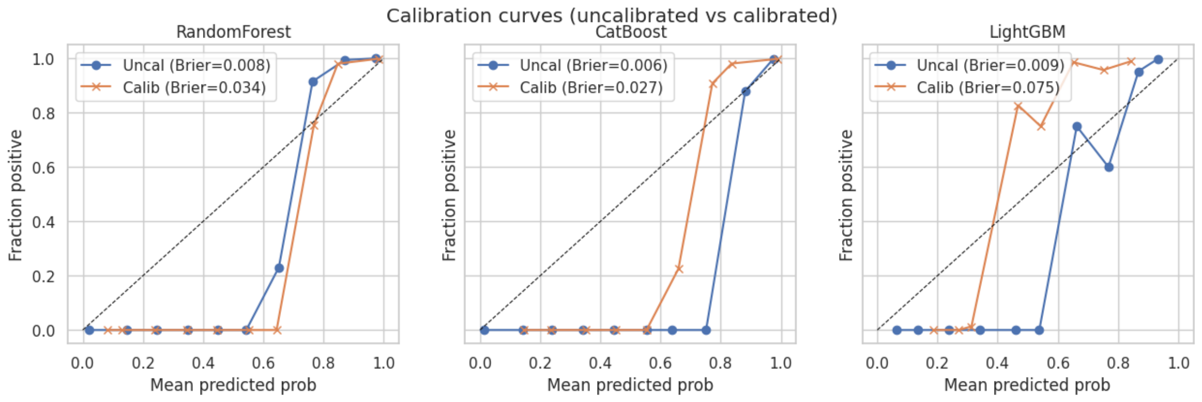
Figure 10. Comparison of RF, CatBoost, and LGBM using fold-wise F1-score distributions (10-fold CV) with bootstrap 95% CIs across datasets (a) CCFD, (b) VICF, (c) FFD, (d) DCCC, (e) PW, and (f) CA.



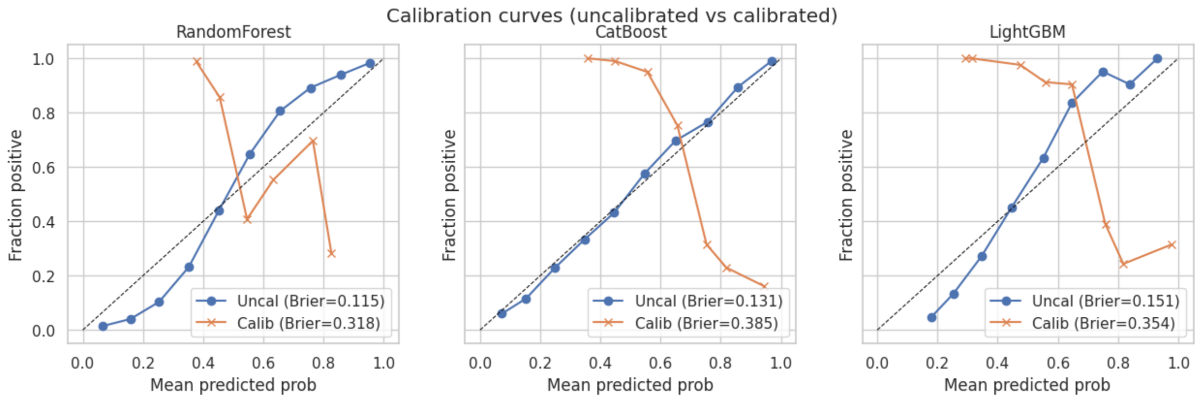
(a)


Calibration curves (uncalibrated vs calibrated)
RandomForest
CatBoost
LightGBM
Uncal (Brier=0.028)
Calib (Brier=0.076)
Uncal (Brier=0.031)
Calib (Brier=0.119)
Uncal (Brier=0.051)
Calib (Brier=0.178)
Fraction positive
Mean predicted prob

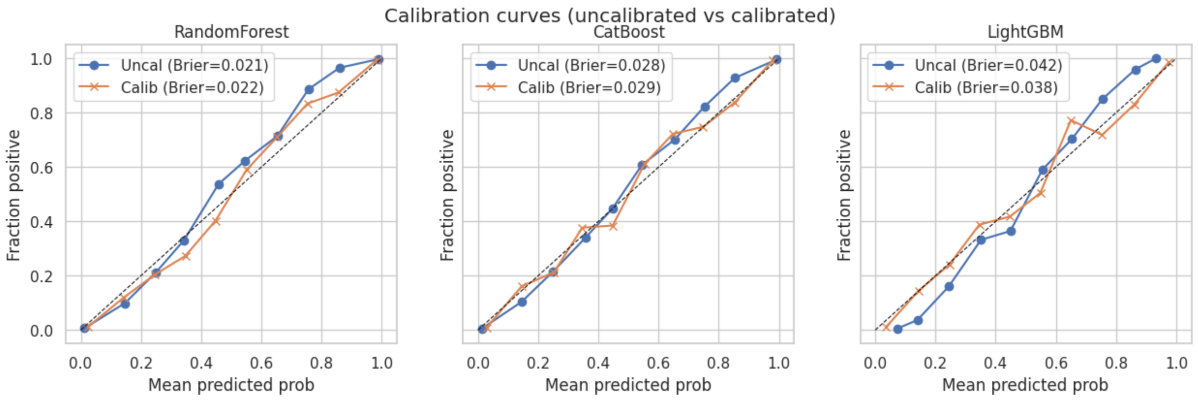
(b)


Calibration curves (uncalibrated vs calibrated)
RandomForest
CatBoost
LightGBM
Uncal (Brier=0.008)
Calib (Brier=0.034)
Uncal (Brier=0.006)
Calib (Brier=0.027)
Uncal (Brier=0.009)
Calib (Brier=0.075)
Fraction positive
Mean predicted prob

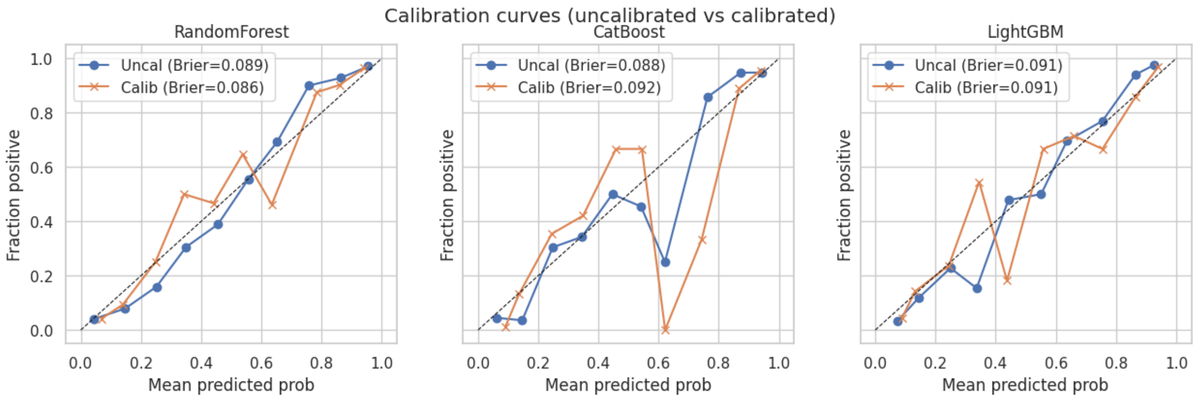
(c)


Calibration curves (uncalibrated vs calibrated)
RandomForest
CatBoost
LightGBM
Uncal (Brier=0.115)
Calib (Brier=0.318)
Uncal (Brier=0.131)
Calib (Brier=0.385)
Uncal (Brier=0.151)
Calib (Brier=0.354)
Fraction positive
Mean predicted prob

(d)

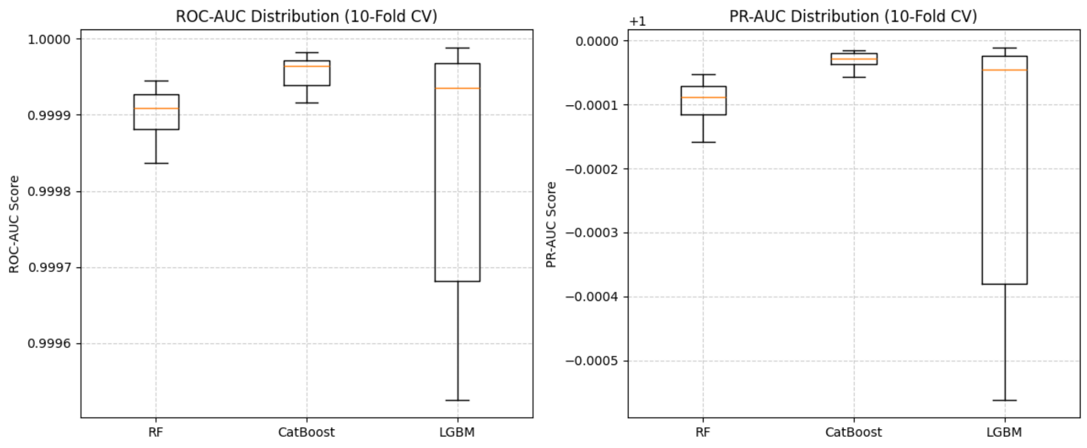


(e)

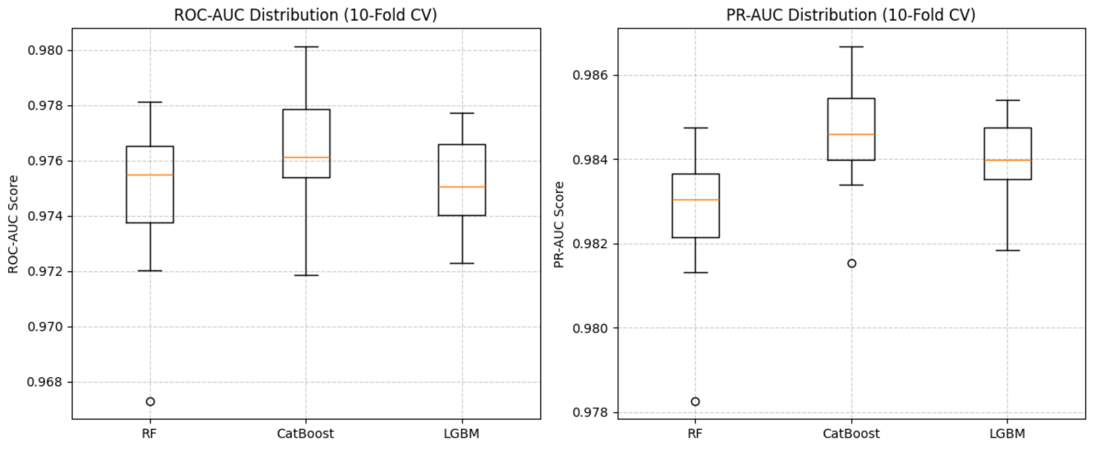


(f)

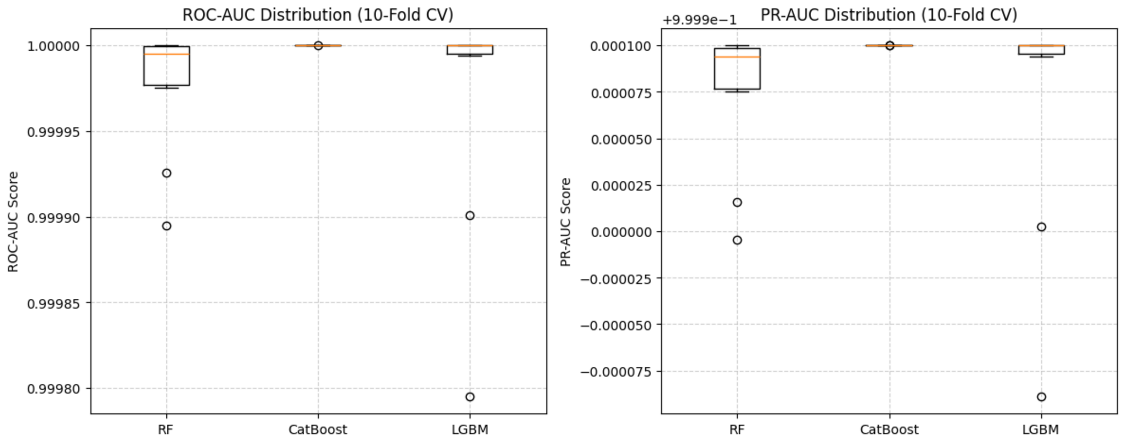
Figure 11. Wilcoxon signed-rank test calibration curves comparing RF, CatBoost, and LGBM across datasets (a) CCFD, (b) VICF, (c) FFD, (d) DCCC, (e) PW, and (f) CA, showing differences in model calibration and reliability.



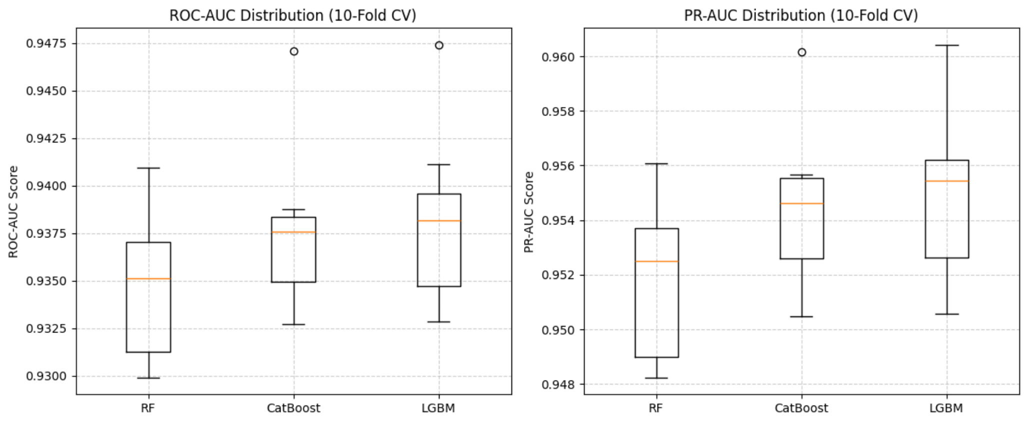
(a)


ROC-AUC Distribution (10-Fold CV)
ROC-AUC Score
PR-AUC Distribution (10-Fold CV)
PR-AUC Score
RF
CatBoost
LGBM

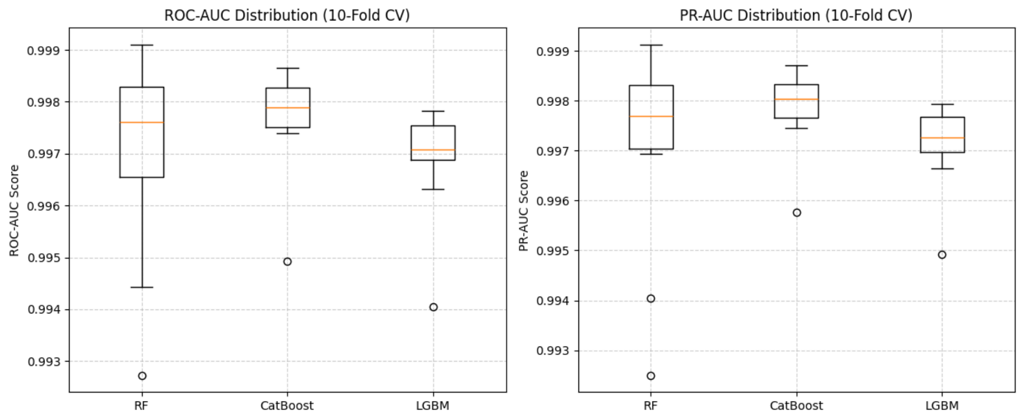
(b)


ROC-AUC Distribution (10-Fold CV)
ROC-AUC Score
PR-AUC Distribution (10-Fold CV)
+9.999e−1
PR-AUC Score
RF
CatBoost
LGBM

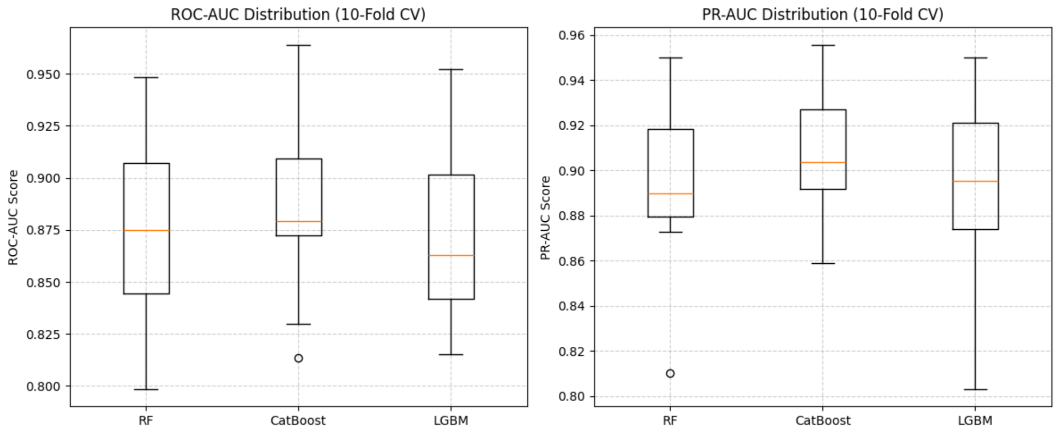
(c)


ROC-AUC Distribution (10-Fold CV)
ROC-AUC Score
PR-AUC Distribution (10-Fold CV)
PR-AUC Score
RF
CatBoost
LGBM

(d)



(e)



(f)

Figure 12. Comparison of ROC-AUC (left) and PR-AUC (right) distributions across 10-fold cross-validation for RF, CatBoost, and LGBM classifiers using several datasets: (a) CCFD, (b) VICF, (c) FFD, (d) DCCC, (e) PW and (f) CA.

Conflicts of Interest

The authors confirm that there is no conflict of interest to declare for this publication.

Acknowledgments

I would like to express my sincere gratitude to my supervisor, Dr. Vikas Thada, for their invaluable support and guidance throughout this work. Special thanks to co-guide Dr. Amit Kumar Mishra for their insightful feedback and encouragement. I also appreciate the resources and facilities provided by Amity University, Gwalior, India, which made this research possible.

AI Disclosure

The authors declare that no assistance is taken from generative AI to write this article.

References

Adiputra, I.N.M., & Wanchai, P. (2024). CTGAN-ENN: a tabular GAN-based hybrid sampling method for imbalanced and overlapped data in customer churn prediction. *Journal of Big Data*, *11*(1), 121. https://doi.org/10.1186/s40537-024-00982-x.

Al Hosni, O., & Starkey, A. (2023). Investigating the performance of data complexity & instance hardness measures as a meta-feature in overlapping classes problem. In *Proceedings of the 2023 7th International Conference on Cloud and Big Data Computing* (pp. 1-9). Association for Computing Machinery. Manchester, United Kingdom. https://doi.org/10.1145/3616131.3616132.

Alamri, M., & Ykhlef, M. (2024). Hybrid undersampling and oversampling for handling imbalanced credit card data. *IEEE Access*, *12*, 14050-14060. https://doi.org/10.1109/ACCESS.2024.3357091.

Alarfaj, F.K., Malik, I., Khan, H.U., Almusallam, N., Ramzan, M., & Ahmed, M. (2022). Credit card fraud detection using state-of-the-art machine learning and deep learning algorithms. *IEEE Access*, *10*, 39700-39715. https://doi.org/10.1109/ACCESS.2022.3166891.

Alharbi, S., Alorini, A., Alahmadi, K., Alhosaini, H., Zhu, Y., & Wang, X. (2024). Exploring oversampling techniques for fraud detection with imbalanced classes. *International Journal of Computer Vision and Signal Processing*, *14*(1), 26-33. http://cennser.org/IJCVSP.

Araf, I., Idri, A., & Chairi, I. (2024). Cost-sensitive learning for imbalanced medical data: a review. *Artificial Intelligence Review*, *57*(4), 80. https://doi.org/10.1007/s10462-023-10652-8.

Aros, L.H., Molano, L.X.B., Portela, F.G., Hernandez, J.J.M., & Barrero, M.S.R. (2024). Financial fraud detection through the application of machine learning techniques: a literature review. *Humanities and Social Sciences Communications*, *11*(1), 1-22. https://doi.org/10.1057/s41599-024-03606-0.

Bank, D., Koenigstein, N., & Giryes, R. (2020). *Autoencoders*. http://arxiv.org/abs/2003.05991.

Bansal, S. (2021). Vehicle insurance claim fraud detection dataset. URL: https://www.kaggle.com/datasets/shivamb/vehicle-claim-fraud-detection

Batista, G.E., Prati, R.C., & Monard, M.C. (2004). A study of the behavior of several methods for balancing machine learning training data. *ACM SIGKDD Explorations Newsletter*, *6*(1), 20-29. https://doi.org/10.1145/1007730.1007735.

Bengio, Y., Courville, A., & Vincent, P. (2013). Representation learning: a review and new perspectives. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *35*(8), 1798-1828. https://doi.org/10.1109/TPAMI.2013.50.

Berahmand, K., Daneshfar, F., Salehi, E.S., Li, Y., & Xu, Y. (2024). Autoencoders and their applications in machine learning: a survey. *Artificial Intelligence Review*, *57*(2), 28. https://doi.org/10.1007/s10462-023-10662-6.

Bishop, C.M. (2006). *Pattern recognition and machine learning.* Springer, New York.

Bounab, R., Zarour, K., Guelib, B., & Khlifa, N. (2024). Enhancing medicare fraud detection through machine learning: addressing class imbalance with SMOTE-ENN. *IEEE Access*, *12*, 54382-54396. https://doi.org/10.1109/ACCESS.2024.3385781.

Breiman, L. (2001). Random forests. *Machine Learning*, *45*(1), 5-32.

Bunkhumpornpat, C., Boonchieng, E., Chouvatut, V., & Lipsky, D. (2024). FLEX-SMOTE: Synthetic over-sampling technique that flexibly adjusts to different minority class distributions. *Patterns*, *5*(11), 101073. https://doi.org/10.1016/j.patter.2024.101073.

Bunkhumpornpat, C., Sinapiromsaran, K., & Lursinsap, C. (2012). DBSMOTE: density-based synthetic minority over-sampling technique. *Applied Intelligence*, *36*(3), 664-684. https://doi.org/10.1007/s10489-011-0287-y.

Carvalho, M., Pinho, A.J., & Brás, S. (2025). Resampling approaches to handle class imbalance: a review from a data perspective. *Journal of Big Data*, *12*(1), 71. https://doi.org/10.1186/s40537-025-01119-4.

Cavalcanti, G.D., Ren, T.I., & Vale, B.A. (2012). Data complexity measures and nearest neighbor classifiers: a practical analysis for meta-learning. In *2012 IEEE 24th International Conference on Tools with Artificial Intelligence* (Vol. 1, pp. 1065-1069). IEEE. Athens, Greece. https://doi.org/10.1109/ICTAI.2012.150.

Cemernek, D., Siddiqi, S., & Kern, R. (2024). Effects of class imbalance countermeasures on interpretability. *IEEE Access*, *12*, 45342-45358. https://doi.org/10.1109/ACCESS.2024.3381536.

Chawla, N.V., Bowyer, K.W., Hall, L.O., & Kegelmeyer, W.P. (2002). SMOTE: synthetic minority over-sampling technique. *Journal of Artificial Intelligence Research*, *16*, 321-357.

Chawla, N.V., Lazarevic, A., Hall, L.O., & Bowyer, K.W. (2003). SMOTEBoost: improving prediction of the minority class in boosting. In *European Conference on Principles of Data Mining and Knowledge Discovery* (pp. 107-119). Springer. Berlin, Heidelberg. https://doi.org/10.1007/978-3-540-39804-2_12.

Chicco, D., & Jurman, G. (2020). The advantages of the Matthews correlation coefficient (MCC) over F1 score and accuracy in binary classification evaluation. *BMC Genomics*, *21*(1), 6. https://doi.org/10.1186/s12864-019-6413-7.

Cohen, J. (1960). A coefficient of agreement for nominal scales. *Educational and Psychological Measurement*, *20*(1), 37-46. https://doi.org/10.1177/001316446002000104.

Combert, F.K., Xie, S., & Lawniczak, A.T. (2025). Bi-partitioned feature-weighted k-means clustering for detecting insurance fraud claim patterns. *Mathematics*, *13*(3), 34. https://doi.org/10.3390/math13030434.

de Vargas, V.W., Aranda, J.A.S., dos Santos Costa, R., da Silva Pereira, P.R., & Barbosa, J.L.V. (2023). Imbalanced data preprocessing techniques for machine learning: a systematic mapping study. *Knowledge and Information Systems*, *65*(1), 31-57. https://doi.org/10.1007/s10115-022-01772-8.

Distefano, V., Palma, M., & de Iaco, S. (2024). Multi-class random forest model to classify wastewater treatment imbalanced data. *Socio-Economic Planning Sciences*, *95*, 102021. https://doi.org/10.1016/j.seps.2024.102021.

Du, H., Lv, L., Wang, H., & Guo, A. (2024). A novel method for detecting credit card fraud problems. *PLOS One*, *19*(3), 294-537. https://doi.org/10.1371/journal.pone.0294537.

Eberlein, J., Rodriguez, D., & Harrison, R. (2025). The effect of data complexity on classifier performance. *Empirical Software Engineering*, *30*(1), 16. https://doi.org/10.1007/s10664-024-10554-5.

Elamir, E.A. (2022). A graphical approach for Friedman test: moments approach. *arXiv preprint arXiv:2202.09131*.

Elreedy, D., Atiya, A.F., & Kamalov, F. (2024). A theoretical distribution analysis of synthetic minority oversampling technique (SMOTE) for imbalanced learning. *Machine Learning*, *113*(7), 4903-4923. https://doi.org/10.1007/s10994-022-06296-4.

Esenogho, E., Mienye, I.D., Swart, T.G., Aruleba, K., & Obaido, G. (2022). A neural network ensemble with feature engineering for improved credit card fraud detection. *IEEE Access*, *10*, 16400-16407. https://doi.org/10.1109/ACCESS.2022.3148298.

Fernández, A., Garcia, S., Herrera, F., & Chawla, N.V. (2018). SMOTE for learning from imbalanced data: progress and challenges, marking the 15-year anniversary. *Journal of Artificial Intelligence Research*, *61*, 863-905. https://doi.org/10.1613/jair.1.11192.

Fonseca, J., Douzas, G., & Bacao, F. (2021). Improving imbalanced land cover classification with k-means smote: Detecting and oversampling distinctive minority spectral signatures. *Information*, *12*(7), 266. https://doi.org/10.3390/info12070266.

Gheysarbeigi, A., Rakhshaninejad, M., Fathian, M., & Barzinpour, F. (2025). An ensemble-based auto insurance fraud detection using BQANA hyperparameter tuning. *IEEE Access*. *13*, 42997-43012. https://doi.org/10.1109/ACCESS.2025.3548529.

Gupta, R.K., Hassan, A., Majhi, S.K., Parveen, N., Zamani, A.T., Anitha, R., Ojha, B., Singh, A.K., & Muduli, D. (2025). Enhanced framework for credit card fraud detection using robust feature selection and a stacking ensemble model approach. *Results in Engineering*, *26*, 105084. https://doi.org/10.1016/j.rineng.2025.105084.

Han, H., Wang, W.Y., & Mao, B.H. (2005). Borderline-SMOTE: a new over-sampling method in imbalanced data sets learning. In *International Conference on Intelligent Computing* (pp. 878-887). Springer. Berlin, Heidelberg. https://doi.org/10.1007/11538059_91.

Harliman, R., & Uchida, K. (2018). Data- and algorithm-hybrid approach for imbalanced data problems in deep neural network. *International Journal of Machine Learning and Computing*, *8*(3), 208-213. https://doi.org/10.18178/ijmlc.2018.8.3.689.

Hashemi, S.K., Mirtaheri, S.L., & Greco, S. (2023). Fraud detection in banking data by machine learning techniques. *IEEE Access*, *11*, 3034-3043. https://doi.org/10.1109/ACCESS.2022.3232287.

He, H., Bai, Y., Garcia, E.A., & Li, S. (2008). ADASYN: adaptive synthetic sampling approach for imbalanced learning. In *2008 IEEE International Joint Conference on Neural Networks* (pp. 1322-1328). IEEE. Hong Kong. https://doi.org/10.1109/IJCNN.2008.4633969.

Ho, T.K., & Basu, M. (2002). Complexity measures of supervised classification problems. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *24*(3), 289-300. https://doi.org/10.1109/34.990132.

Holm, S. (1979). A simple sequentially rejective multiple test procedure. *Scandinavian Journal of Statistics, 6*(2), 65-70. https://www.jstor.org/stable/4615733.

Hong, S., An, S., & Jeon, J.J. (2025). Improving SMOTE via fusing conditional VAE for data-adaptive noise filtering. *Applied Intelligence*, *55*(12), 841. https://doi.org/10.1007/s10489-025-06692-y.

Ileberi, E., Sun, Y., & Wang, Z. (2021). Performance evaluation of machine learning methods for credit card fraud detection using SMOTE and AdaBoost. *IEEE Access*, *9*, 165286-165294. https://doi.org/10.1109/ACCESS.2021.3134330.

Ileberi, E., Sun, Y., & Wang, Z. (2022). A machine learning based credit card fraud detection using the GA algorithm for feature selection. *Journal of Big Data*, *9*(1), 24. https://doi.org/10.1186/s40537-022-00573-8.

Jemai, J., Zarrad, A., & Daud, A. (2024). Identifying fraudulent credit card transactions using ensemble learning. *IEEE Access*, *12*, 54893-54900. https://doi.org/10.1109/ACCESS.2024.3380823.

Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., Ye, Q., & Liu, T.Y. (2017). Lightgbm: a highly efficient gradient boosting decision tree. *Advances in Neural Information Processing Systems*, *30*, 3146-3154.

Kennedy, R.K., Salekshahrezaee, Z., Villanustre, F., & Khoshgoftaar, T.M. (2023). Iterative cleaning and learning of big highly-imbalanced fraud data using unsupervised learning. *Journal of Big Data*, *10*(1), 106. https://doi.org/10.1186/s40537-023-00750-3.

Kennedy, R.K., Villanustre, F., Khoshgoftaar, T.M., & Salekshahrezaee, Z. (2024). Synthesizing class labels for highly imbalanced credit card fraud detection data. *Journal of Big Data*, *11*(1), 38. https://doi.org/10.1186/s40537-024-00897-7.

Khalid, A.R., Owoh, N., Uthmani, O., Ashawa, M., Osamor, J., & Adejoh, J. (2024). Enhancing credit card fraud detection: an ensemble machine learning approach. *Big Data and Cognitive Computing*, *8*(1), 6. https://doi.org/10.3390/bdcc8010006.

Khorshidi, H.A., & Aickelin, U. (2025). A synthetic over-sampling method with minority and majority classes for imbalance problems. *Knowledge and Information Systems*, *67*(7), 5965-5998. https://doi.org/10.1007/s10115-025-02394-6

Komorniczak, J., & Ksieniewicz, P. (2022). Problexity--an open-source Python library for binary classification problem complexity assessment. *arXiv preprint arXiv:2207.06709*.

Kubat, M. (1997). Addressing the curse of imbalanced training sets: one-sided selection. In *Proceedings of the 14th International Conference on Machine Learning* (pp. 179-186). Morgan Kaufmann. Nashville, Tennessee, USA.

Kumbhar, M., Bandaru, S., & Karlsson, A. (2026). Imbalanced data oversampling through subspace optimization with Bayesian reinforcement. *Artificial Intelligence Review*, *59*(1), 1. https://doi.org/10.1007/s10462-025-11417-1.

Lamane, H., Mouhir, L., Moussadek, R., Baghdad, B., Kisi, O., & El Bilali, A. (2024). Interpreting machine learning models based on SHAP values in predicting suspended sediment concentration. *International Journal of Sediment Research*, *40*(1), 91-107. https://doi.org/10.1016/j.ijsrc.2024.10.002.

Lancho, C., De Diego, I.M., Cuesta, M., Acena, V., & Moguerza, J. M. (2023). Hostility measure for multi-level study of data complexity. *Applied Intelligence*, *53*(7), 8073-8096. https://doi.org/10.1007/s10489-022-03793-w.

Laurikkala, J. (2001). Improving identification of difficult small classes by balancing class distribution. In *Conference on Artificial Intelligence in Medicine in Europe* (pp. 63-66). Springer. Berlin, Heidelberg.

Li, P. (2022). Credit card fraud detection based on random forest model. *Academic Journal of Computing & Information Science*, *5*(13), 55-61. https://doi.org/10.25236/ajcis.2022.051309

Lorena, A.C., Garcia, L.P., Lehmann, J., Souto, M.C., & Ho, T.K. (2019). How complex is your classification problem? a survey on measuring classification complexity. *ACM Computing Surveys, 52*(5), 1-34. https://doi.org/10.1145/3347711.

Mahmood, Z., Safran, M., Abdussamad, Alfarhood, S., & Ashraf, I. (2025). Algorithmic and mathematical modeling for synthetically controlled overlapping. *Scientific Reports*, *15*(1), 7517. https://doi.org/10.1038/s41598-025-87992-8.

Mahmudah, K.R., Indriani, F., Takemori-Sakai, Y., Iwata, Y., Wada, T., & Satou, K. (2021). Classification of imbalanced data represented as binary features. *Applied Sciences*, *11*(17), 7825. https://doi.org/10.3390/app11177825.

Maiano, L., Montuschi, A., Caserio, M., Ferri, E., Kieffer, F., Germanò, C., & Anagnostopoulos, A. (2023). A deep-learning–based antifraud system for car-insurance claims. *Expert Systems with Applications*, *231*, 120644. https://doi.org/10.1016/j.eswa.2023.120644.

Makhzani, A., Shlens, J., Jaitly, N., Goodfellow, I., & Frey, B. (2015). Adversarial autoencoders. *arXiv preprint arXiv:1511.05644*.

Michelucci, U. (2022). An introduction to autoencoders. *arXiv preprint arXiv:2201.03898*.

Mienye, I.D., & Sun, Y. (2023). A deep learning ensemble with data resampling for credit card fraud detection. *IEEE Access*, *11*, 30628-30638. https://doi.org/10.1109/ACCESS.2023.3262020.

Ming, R., Mohamad, O., Innab, N., & Hanafy, M. (2024). Bagging vs. boosting in ensemble machine learning? an integrated application to fraud risk analysis in the insurance sector. *Applied Artificial Intelligence*, *38*(1), 2355024. https://doi.org/10.1080/08839514.2024.2355024.

Mohammad, R.M., Thabtah, F., & McCluskey, L. (2015). Phishing website dataset: UCI machine learning repository. *University of California, School of Information and Computer Science, Irvine*. https://doi.org/10.24432/C51W2X.

Mujahid, M., Kına, E.R.O.L., Rustam, F., Villar, M.G., Alvarado, E.S., De La Torre Diez, I., & Ashraf, I. (2024). Data oversampling and imbalanced datasets: an investigation of performance for machine learning and feature engineering. *Journal of Big Data*, *11*(1), 87. https://doi.org/10.1186/s40537-024-00943-4.

Nordin, S.Z.S., Wah, Y.B., Haur, N.K., Hashim, A., Norimah, R., & Jalil, N.A. (2024). Predicting automobile insurance fraud using classical and machine learning models. *International Journal of Electrical and Computer Engineering*, *14*(1), 911-921. https://doi.org/10.11591/ijece.v14i1.pp911-921.

Nouas, S., Oukid, L., & Boumahdi, F. (2025). Syngo: synthetic genetic oversampling technique for textual data. *Social Network Analysis and Mining*, *15*(1), 1-21. https://doi.org/10.1007/s13278-025-01423-0.

Palivela, H., Rishiwal, V., Bhushan, S., Alotaibi, A., Agarwal, U., Kumar, P., & Yadav, M. (2024). Optimisation of deep learning based model for identification of credit card frauds. *IEEE Access*, *12*, 125629-125642. https://doi.org/10.1109/ACCESS.2024.3440637.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Vanderplas, J., Cournapeau, D., Pedregosa, F., Varoquaux, G., Gramfort, A., Thirion, B., Grisel, O., Dubourg, V., Passos, A., Brucher, M., Perrot, M., & Duchesnay, É. (2011). Scikit-learn: machine learning in python. *The Journal of Machine Learning Research*, *12*, 2825-2830.

Pourhabibi, T., Ong, K.L., Kam, B.H., & Boo, Y.L. (2020). Fraud detection: a systematic literature review of graph-based anomaly detection approaches. *Decision Support Systems*, *133*, 113303. https://doi.org/10.1016/j.dss.2020.113303.

Powers, D.M. (2020). Evaluation: from precision, recall and F-measure to ROC, informedness, markedness and correlation. *International Journal of Machine Learning Technology*, *2*(2011), 37-63.

Pozzolo, D.A., Caelen, O., Johnson, R.A., & Bontempi, G. (2015). Calibrating probability with undersampling for unbalanced classification. In *2015 IEEE Symposium Series on Computational Intelligence* (pp. 159-166). IEEE. Cape Town, South Africa. https://doi.org/10.1109/SSCI.2015.33.

Prokhorenkova, L., Gusev, G., Vorobev, A., Dorogush, A.V., & Gulin, A. (2018). CatBoost: unbiased boosting with categorical features. *Advances in Neural Information Processing Systems*, *31*, 6638-6648.

Quinlan, J. (1987). Credit approval [Dataset]. UCI machine learning repository. https://doi.org/10.24432/C5FS30.

Research, G. (2024). Financial fraud detection. URL: https://colab.research.google.com/.accessed:2025-03-31.

Rousseeuw, P.J. (1987). Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. *Journal of Computational and Applied Mathematics*, *20*, 53-65. https://doi.org/10.1016/0377-0427(87)90125-7.

Ruff, L., Vandermeulen, R., Goernitz, N., Deecke, L., Siddiqui, S.A., Binder, A., Müller, E., & Kloft, M. (2021). A unifying review of deep and shallow anomaly detection. *Proceedings of the IEEE, 109*(5), 756-795. https://doi.org/10.1109/JPROC.2021.3052449.

Sáez, J.A., Luengo, J., Stefanowski, J., & Herrera, F. (2015). SMOTE–IPF: addressing the noisy and borderline examples problem in imbalanced classification by a re-sampling method with filtering. *Information Sciences*, *291*, 184-203. https://doi.org/10.1016/j.ins.2014.08.051.

Saito, T., & Rehmsmeier, M. (2015). The precision-recall plot is more informative than the ROC plot when evaluating binary classifiers on imbalanced datasets. *PLOS One*, *10*(3), e0118432. https://doi.org/10.1371/journal.pone.0118432.

Salam, M.A., Fouad, K.M., Elbably, D.L., & Elsayed, S.M. (2024). Federated learning model for credit card fraud detection with data balancing techniques. *Neural Computing and Applications*, *36*(11), 6231-6256. https://doi.org/10.1007/s00521-023-09410-2.

Salehi, A.R., & Khedmati, M. (2024). A cluster-based SMOTE both-sampling (CSBBoost) ensemble algorithm for classifying imbalanced data. *Scientific Reports*, *14*(1), 5152. https://doi.org/10.1038/s41598-024-55598-1.

Salekshahrezaee, Z., Leevy, J.L., & Khoshgoftaar, T.M. (2023). The effect of feature extraction and data sampling on credit card fraud detection. *Journal of Big Data*, *10*(1), 6. https://doi.org/10.1186/s40537-023-00684-w.

Sarker, A., Yasmin, Must, A., Md Rahman, A., Md Rashid, H.O., & Roy, B.R. (2024). Credit card fraud detection using machine learning techniques. *Journal of Computer and Communications*, *12*(6), 1-11. https://doi.org/10.4236/jcc.2024.126001.

Seiffert, C., Khoshgoftaar, T.M., Van Hulse, J., & Napolitano, A. (2010). RUSBoost: a hybrid approach to alleviating class imbalance. *IEEE Transactions on Systems, Man, and Cybernetics Part A: Systems and Humans*, *40*(1), 185-197. https://doi.org/10.1109/TSMCA.2009.2029559.

Singh, A., Ranjan, R.K., & Tiwari, A. (2022). Credit card fraud detection under extreme imbalanced data: a comparative study of data-level algorithms. *Journal of Experimental & Theoretical Artificial Intelligence*, *34*(4), 571-598. https://doi.org/10.1080/0952813X.2021.1907795.

Sisodia, D., & Sisodia, D.S. (2022). Quad division prototype selection-based k-nearest neighbor classifier for click fraud detection from highly skewed user click dataset. *Engineering Science and Technology, an International Journal*, *28*, 101011. https://doi.org/10.1016/j.jestch.2021.05.015.

Smith, M.Q.P., & Ruxton, G.D. (2020). Effective use of the McNemar test. *Behavioral Ecology and Sociobiology*, *74*(11), 133. https://doi.org/10.1007/s00265-020-02916-y.

Tao, L., Li, H., Wang, F., Liu, M., Tang, Z., & Wang, Q. (2024). An adaptive safe-region diversity oversampling algorithm for imbalanced classification. *IEEE Access*, *12*, 63713-63724. https://doi.org/10.1109/ACCESS.2024.3396155.

Templ, M., & Ulmer, M. (2024). The impact of misclassifications and outliers on imputation methods. *Journal of Applied Statistics*, *51*(14), 2894-2928. https://doi.org/10.1080/02664763.2024.2325969.

Tomek, I. (1976). Two modifications of CNN. *IEEE Transactions on Systems, Man, and Cybernetics*, *6*(11), 769-772.

Wang, A.X., Chukova, S.S., & Nguyen, B.P. (2023). Synthetic minority oversampling using edited displacement-based k-nearest neighbors. *Applied Soft Computing*, *148*, 110895. https://doi.org/10.1016/j.asoc.2023.110895.

Wen, J., Tang, X., & Lu, J. (2024). An imbalanced learning method based on graph tran-smote for fraud detection. *Scientific Reports*, *14*(1), 16560. https://doi.org/10.1038/s41598-024-67550-4.

Wilson, D.L. (1972). Asymptotic properties of nearest neighbor rules using edited data. *IEEE Transactions on Systems, Man, and Cybernetics*, *SMC-2*(3), 408-421. https://doi.org/10.1109/TSMC.1972.4309137.

Woolson, R.F. (2007). Wilcoxon signed-rank test. *Wiley Encyclopedia of Clinical Trials*, *1*(1), 1-3. https://doi.org/10.1002/9780471462422.eoct979.

Wu, E., Cui, H., & Welsch, R.E. (2020). Dual autoencoders generative adversarial network for imbalanced classification problem. *IEEE Access*, *8*, 91265-91275. https://doi.org/10.1109/ACCESS.2020.2994327.

Xu, C., Liang, X., Sun, Y., & He, X. (2024). Fraudsters beware: unleashing the power of metaverse technology to uncover financial fraud. *International Journal of Human–Computer Interaction*, *40*(18), 4987-5002. https://doi.org/10.1080/10447318.2023.2238367.

Yeh, I. (2009). *Default of credit card clients* [*Dataset*]. UCI Machine Learning Repository. https://doi.org/10.24432/C55S3H.

Yuan, X., Chen, S., Zhou, H., Sun, C., & Yuwen, L. (2023). CHSMOTE: convex hull-based synthetic minority oversampling technique for alleviating the class imbalance problem. *Information Sciences*, *623*, 324-341. https://doi.org/10.1016/J.INS.2022.12.056.

Zhou, C., & Paffenroth, R.C. (2017). Anomaly detection with robust deep autoencoders. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (pp. 665-674). Association for Computing Machinery. Canada. https://doi.org/10.1145/3097983.3098052.

Zhou, H., Pan, H., Zheng, K., Wu, Z., & Xiang, Q. (2025). A novel oversampling method based on Wasserstein CGAN for imbalanced classification. *Cybersecurity*, *8*(1), 7. https://doi.org/10.1186/s42400-024-00290-0.

Zong, B., Song, Q., Min, M.R., Cheng, W., Lumezanu, C., Cho, D., & Chen, H. (2018). Deep autoencoding Gaussian mixture model for unsupervised anomaly detection. *Journal of Machine Learning Research, 21*(1), 1-37.