

Enhancing Money Laundering Detection: A Comparative Study of Machine Learning Techniques and Sampling Methods

Vinh Dinh Nguyen(✉) and Kha Hoang Nguyen

Department of Information Technology, FPT University, Can Tho Campus, Can Tho City, Vietnam
vinhnd29@fe.edu.vn, khanhce171115@fpt.edu.vn

Abstract. Money laundering poses a significant threat to global financial systems, enabling illicit activities such as drug trafficking and terrorism. Traditional anti-money laundering methods often fail to detect complex and evolving patterns in financial transactions. This paper explores advanced machine learning techniques to enhance the detection of suspicious activities. We evaluated the performance of six classifiers—RandomForest, LogisticRegression, KNeighbors, GradientBoosting, LightGBM, and XGBoost—on an imbalanced dataset for laundering transaction classification. Four sampling methods were employed: Without Sampling, SMOTE, RandomUnderSampler, and ADASYN. The results indicate that RandomForest, particularly when combined with SMOTE and ADASYN, consistently outperformed other classifiers. Without sampling, RandomForest achieved an F1 Score of 0.37 for the minority class (Class 1), which improved to 0.41 with SMOTE and 0.40 with ADASYN. These findings underscore the robustness of RandomForest in managing imbalanced datasets and emphasize the critical role of effective sampling techniques. This study offers valuable insights for enhancing predictive models in financial fraud detection.

Keywords: Money laundering detection · machine learning · imbalanced datasets · sampling techniques · fraud detection

1 Introduction

Money laundering is a major threat to global financial systems, facilitating illicit activities like drug trafficking and terrorism [1]. Traditional anti-money laundering (AML) methods often struggle to detect the evolving complexities of financial transactions, necessitating more sophisticated approaches [2]. This process allows criminals to disguise illicit funds as legitimate, significantly impacting society by enabling crimes such as human trafficking and terrorism, particularly in developing countries [3, 4]. Techniques like smurfing—breaking large sums into smaller transactions to evade detection—make tracing these funds difficult [1]. Detecting money laundering is challenging due to its secretive nature and multi-stage process, including placement, layering, and integration [5, 6]. The Financial Action Task Force (FATF) emphasizes the complexity

© The Author(s), under exclusive license to Springer Nature Switzerland AG 2025
O. Gervasi et al. (Eds.): ICCSA 2025, LNCS 15650, pp. 150–161, 2025.
https://doi.org/10.1007/978-3-031-96962-1_10

of laundering networks and the necessity of advanced detection systems [7]. Traditional rule-based AML approaches lack the flexibility to adapt to these challenges [8]. Recently, machine learning (ML) models have shown promise in improving detection by identifying hidden patterns in complex transactions [9]. However, applying ML to AML remains difficult due to the inherent class imbalance in the data—most transactions are legitimate, with only a small fraction being illicit [9]. This imbalance hinders the performance of ML algorithms, which often misclassify the minority class (illicit transactions) [10]. Researchers have developed various techniques to address this, including resampling methods like SMOTE, RandomUnderSampler, and ADASYN, which improve the representation of minority classes without modifying the predictive models [11]. The terms "imbalance learning technique" and "resampling technique" are used interchangeably in this study, as the focus is on enhancing the dataset rather than modifying the predictive model.

This study evaluates the performance of these resampling methods in combination with six popular ML models—RandomForest, LogisticRegression, KNeighbors, GradientBoosting, LightGBM, and XGBoost—using the IBM "Transactions for Anti-Money Laundering" dataset from Kaggle, which includes over 5 million records. The objective is to enhance AML detection by addressing class imbalance and leveraging advanced ML techniques.

2 Related Work

2.1 Supervised Algorithms-Based Methods for Fraud Detection

ML algorithm-based methods are divided into two categories: supervised and unsupervised algorithms [12]. Supervised ML algorithms are widely used in fraud detection due to their ability to learn from labeled datasets, which contain known instances of fraudulent and non-fraudulent activities [13]. These algorithms build models that can predict the likelihood of future transactions being fraudulent based on historical data, thereby supporting and improving the decision-making process in financial institutions [9]. It has been suggested to integrate SMOTE with various supervised ML techniques such as logistic regression (LR), decision trees (DT), neural networks (NN), and random forests (RF) to handle class imbalance effectively [14, 15]. This integration helps create a more balanced training set, enabling the algorithms to better identify and predict fraudulent transactions.

Nguyen et al. (2023) compared several supervised algorithms including DT, RF, Support Vector Machine (SVM), Artificial Neural Network (ANN), and Bayesian Logistic Regression (BLR). They found that ANN outperformed other algorithms in rare event classification, which is crucial in fraud detection where fraudulent transactions are significantly fewer than legitimate ones. Moreover, combining resampling techniques like SMOTE with algorithms such as SVM and RF yielded competitive results, demonstrating the importance of addressing class imbalance in improving model performance [15].

Khalid et al. (2024) demonstrated that XGBoost effectively addresses suboptimal outcomes in fraud detection classification tasks. XGBoost treats non-reported and normal alerts equally, allowing for the development of a robust detection algorithm that predicts

the likelihood of fraudulent transactions [14]. This approach considers various factors, including transaction history and patterns, which enhances the accuracy and reliability of fraud detection systems.

Supervised algorithms leverage historical data to adapt to new fraud patterns, enhancing their ability to distinguish between legitimate and fraudulent activities, making them essential tools in combating financial fraud.

2.2 Class Imbalance in Fraud Detection and Anti-Money Laundering

Class imbalance is a critical issue in machine learning (ML), especially in fraud detection and AML, where most instances belong to the majority class, while the minority class represents rare but crucial events like fraudulent transactions or money laundering [16, 17]. In these domains, detecting the minority class is essential to prevent financial losses and protect system integrity [18]. However, ML models trained on imbalanced datasets often misclassify these minority instances due to their bias towards the majority class, achieving high accuracy at the expense of critical misclassifications [19].

This not only skews prediction results but also erodes public confidence, as failure to detect fraud accurately can result in significant financial losses and reduced trust in institutions [20, 21].

2.3 Addressing Class Imbalance with SMOTE, RandomUnderSampler, and ADASYN

To address the class imbalance problem, various resampling techniques have been developed to reduce the bias towards the majority class and enhance sensitivity towards the minority class [10]. These techniques can be categorized into three main approaches: data-level approaches (resampling techniques), algorithm-level approaches (modifying classification algorithms), and hybrid approaches (combining data and algorithm-level methods) [22].

This study focuses on comparing data-level approaches due to their widespread use and effectiveness in handling imbalanced datasets. Specifically, we examine RandomUnder-Sampler, SMOTE, and ADASYN (Adaptive Synthetic Sampling). By comparing these techniques, we aim to identify the most effective method for improving classification performance on imbalanced datasets [23].

RandomUnderSampler

Random undersampling reduces the number of majority class samples to balance the dataset. While straightforward, this method can lead to the loss of valuable information from the majority class [20]. Jones et al. (2017) proposed an undersampling technique leveraging the Naïve Bayes classifier. This method selects the most informational samples from the training dataset through random primary selection. The model is initially trained on a small set, followed by iterative teaching over the current samples, demonstrating that this undersampling technique is comparable to other resampling methods [20, 23]. Xie et al. (2021) introduced an innovative undersampling technique that utilizes consecutive density peaks to methodically remove samples from the majority class [24]. This approach evaluates two key factors to ascertain the importance of each sample,

creating an optimized sequence for training classifiers. The proposed algorithm demonstrated superior performance compared to six established undersampling methods across 40 public benchmark datasets. Undersampling can be seen as a "lower" computational burden compared to oversampling because it simplifies the dataset rather than expanding it. While oversampling aims to augment the minority class to match the majority class's size, undersampling reduces the majority class's size to achieve balance, often resulting in quicker processing times and lower storage requirements. Both techniques have their pros and cons, and the choice between them often depends on the specific characteristics of the dataset and the goals of the analysis [22].

SMOTE

SMOTE is a widely-used oversampling technique introduced by Chawla et al. [25]. It generates synthetic samples for the minority class by interpolating between existing minority class instances, thus increasing the number of minority class samples. SMOTE has been shown to improve the classification performance of ML models on imbalanced datasets [15, 25]. Several advanced versions of SMOTE have been proposed to further enhance its effectiveness. Dablain et al. (2022) introduced DeepSMOTE, an advanced version of SMOTE that combines an encoder/decoder structure, SMOTE-based properties, and an improved loss function. This method has demonstrated efficiency and power, particularly in GAN-based oversampling cases, enhancing model performance on imbalanced datasets [26]. Moreover, Maulidevi and Surendro (2022) proposed SMOTE-LOF, an enhancement of SMOTE that incorporates the Local Outlier Factor (LOF) to identify and handle noise in imbalanced datasets [27]. The results indicated that SMOTE and its variants outperformed traditional methods, particularly for large-scale datasets with small imbalance ratios, demonstrating their effectiveness in addressing imbalanced datasets [23].

ADASYN (Adaptive Synthetic Sampling)

ADASYN, introduced by He et al. (2008), operates similarly to SMOTE by generating synthetic samples for the minority class [28]. However, ADASYN focuses on creating more synthetic data for instances that are harder for the model to learn. Like SMOTE, it generates synthetic samples along a straight line between a minority class instance and its k-nearest minority class neighbors, typically set to 5. The key difference lies in how ADASYN prioritizes which instances to synthesize. It creates more synthetic samples for minority class instances that have a higher number of majority class neighbors within the k-nearest neighbors' region. Conversely, if a minority class instance has no majority class neighbors within this range, no synthetic samples are generated for that instance. This approach targets the more challenging instances for learning, enhancing the model's ability to handle imbalanced datasets effectively [29]. Thejas et al. (2022) proposed Kalman-SMOTE (KSMOTE), a development of SMOTE integrated with the Kalman filter, which shares similarities with ADASYN in terms of focusing on difficult-to-learn samples [11]. KSMOTE excludes noisy samples from the dataset, including both initial data and synthetically added samples, showing improved performance over existing methods.

3 Methodology

3.1 Dataset and Data Preprocessing

The dataset used is the "IBM Transactions for Anti-Money Laundering" from Kaggle, containing 5,078,345 transaction records with attributes such as timestamps, amounts, and currencies. Each transaction record includes 11 attributes (details see Table 1). Preprocessing steps involved handling missing values, encoding categorical features, and extracting useful features from timestamps [30, 31]. The Fig. 1 show the study framework. All experiments were conducted using Google Colab Pro, a cloud-based platform that provides enhanced computational resources for machine learning development. The models were implemented using Python 3.10 with libraries including pandas, scikit-learn, seaborn, etc. [32, 33] for data processing, machine learning algorithms and imbalanced-learn (Lemaître et al. 2017) for handling class imbalance with sampling techniques such as SMOTE and ADASYN.

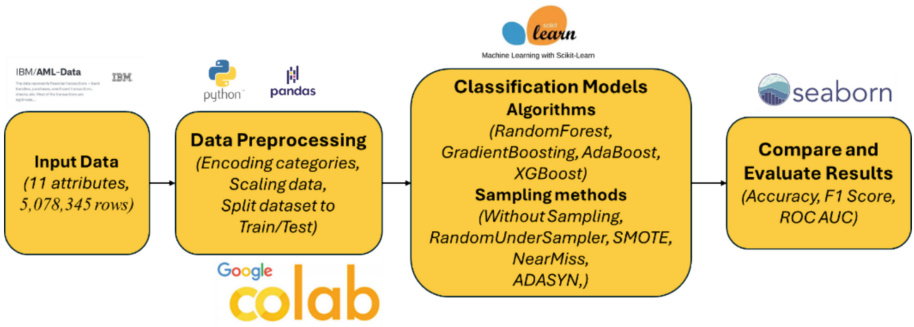


Fig. 1. Experimental workflow

The dataset exhibits a significant imbalance between the classes, with a disproportionately high number of normal transactions compared to laundering transactions. Specifically, there are 5,073,168 instances of non-laundering transactions, whereas laundering transactions account for only 5,177 instances. This pronounced disparity highlights the rarity of laundering transactions within the dataset, posing potential challenges for predictive modeling and necessitating the consideration of appropriate techniques to address class imbalance.

3.2 Imbalance Learning Techniques

To address the class imbalance in the dataset, three sampling methods were used:

- Random Undersampling [20]: Reduces the number of samples in the majority class.
- SMOTE [25]: Balances the dataset by oversampling the minority class.
- ADASYN [28]: Similar to SMOTE but focuses on harder-to-learn examples.Classification Models Training and Evaluation
- The study implements six ML models RandomForest, LogisticRegression, KNeighbors, GradientBoosting, LightGBM and XGBoost.

Table 1. Summary attributes of dataset

No.	Attribute	Data Type	Description
1	Timestamp	DateTime	The date and time when the transaction occurred
2	From Bank	String	Identifier of the originating bank
3	Account	String	Account number or identifier from which the funds were sent
4	To Bank	String	Identifier of the receiving bank
5	Account.1	String	Account number or identifier to which the funds were sent
6	Amount Received	Float	The amount credited to the recipient's account
7	Receiving Currency	Category	Currency in which the recipient received the funds (e.g., USD, EUR)
8	Amount Paid	Float	The amount debited from the sender's account
9	Payment Currency	Category	Currency in which the sender made the payment
10	Payment Format	Category	Method of payment (e.g., wire transfer, credit card, cash)
11	Is Laundering	Binary	Binary label indicating whether the transaction is identified as money laundering (1) or not (0)

3.3 Classification Models Training and Evaluation

The study implements five ML models to evaluate their effectiveness in detecting money laundering activities:

- RandomForest [34]: An ensemble learning method that constructs multiple decision trees during training and outputs the mode of the classes (classification) of the individual trees. It is robust to overfitting and can handle large datasets with high dimensionality.
- LogisticRegression [35]: A linear model used for binary classification tasks. It estimates the probability of a binary response based on one or more predictor variables and is effective for simple and linearly separable datasets.

$$J(\varphi) = -\frac{1}{m}\left[\sum_{i=1}^{m} y^{(i)} log h^{\varphi}\left(x^{(i)}\right) + \left(1 - y^{(i)}\right)\log(1 - h^{\varphi}\left(x^{(i)}\right))\right] \quad (1)$$

- KNeighbors (K-Nearest Neighbors) [36]: A non-parametric method used for classification by comparing the distance between different instances. It is simple to implement and understand but can be computationally intensive with large datasets. To measure the classification cost, we used the Minknowski distance as follows:

$$\Delta(x, y) = \left(\sum_{j=0}^{n=-1} |x_i - y_i|^p\right)^{1/p} \quad (2)$$

GradientBoosting [37]: An ensemble technique that builds models sequentially, each correcting the errors of its predecessor. It is highly effective for a variety of classification tasks but requires careful tuning to avoid overfitting. The loss function was calculated by using the log-likelyhood function as follows

$$\Upsilon = -\sum_{j=1}^{n} y_i \log(p) + (1-p)\log(1-p) \tag{3}$$

:

- LightGBM (Light Gradient Boosting Machine) [38]: A gradient boosting framework that uses tree-based learning algorithms. It is designed for speed and efficiency, capable of handling large datasets with low memory usage.
- XGBoost (Extreme Gradient Boosting) [39]: An efficient and powerful gradient boosting algorithm widely used for its high accuracy and performance. It features regularization to prevent overfitting, handles missing data well, and supports parallel processing. Its flexibility and scalability make it ideal for various ML tasks, including classification and regression. The loss function of XGBoost is depicted as follows:

$$\mathcal{H}^t = \sum_{j=1}^{n} \Upsilon\left(y_i, \widehat{y}_i^{t-1} + f_t(x_i)\right) + \Psi(f_t) \tag{4}$$

where $\Psi(f_t)$ is a regularization term.

The models were evaluated using precision (ζ), recall (∇), F1-score ($\mathcal{F}1$), and confusion matrices as follows:

$$\zeta = \frac{\rho}{\psi + \rho} \tag{5}$$

$$\nabla = \frac{\rho}{\rho + \alpha} \tag{6}$$

$$\mathcal{F}1 = 2 * \frac{\zeta * \nabla}{\zeta + \nabla} \tag{7}$$

where ρ, ψ, and α are the number true positives, number of false positives, and number of false negatives, respectively.

To evaluate the models, this study employs the confusion matrix, which systematically summarizes the number of correct and incorrect predictions in terms of true positives, false positives, true negatives, and false negatives [40]. This framework supports the calculation of key performance metrics, including precision, recall, and F1-score. In addition, metrics such as accuracy, ROC AUC, and F1-score are utilized to provide a comprehensive assessment of each model's effectiveness, particularly in handling class imbalance.

4 Experimental Results

4.1 Data Exploration

It is the process of discovering data insights via the use of various visualization approaches. It is the initial phase in the data analysis process. Pre-processing stage is a crucial stage in ML since it enhances the quality of the data, allowing us to extract

relevant knowledge from big data with greater ease. Additionally, data becomes enough for modelling purposes. In this study, we evaluated the performance of six different classifiers: RandomForest, LogisticRegression, KNeighbors, GradientBoosting, LightGBM, and XGBoost. Each classifier was tested using four different sampling methods: Without Sampling, SMOTE, RandomUnderSampler, and ADASYN. The performance metrics used for evaluation were Accuracy, F1 Score for Class 0 and Class 1, and ROC AUC. Given the imbalanced nature of the dataset, we will focus our discussion on the F1 scores, particularly for Class 1 (the minority class). Table 2 shows detailed metrics of the models.

Table 2. Summary results of ML models with sampling methods

Models	Sampling Methods	Accuracy	F1 Score (Class 0)	**F1 Score (Class 1)**	ROC AUC
Random Forest	*Without Sampling*	1	1	0.37	0.86
	SMOTE	**1**	**1**	**0.41**	**0.89**
	RandomUnderSampler	0.87	0.93	0.01	0.96
	ADASYN	**1**	**1**	**0.4**	**0.9**
Logistic Regression	*Without Sampling*	1	1	0	0.57
	SMOTE	0.62	0.76	0	0.75
	RandomUnderSampler	0.86	0.93	0.01	0.89
	ADASYN	0.86	0.92	0.01	0.89
KNeighbors	*Without Sampling*	1	1	0.1	0.56
	SMOTE	0.97	0.98	0.02	0.67
	RandomUnderSampler	0.67	0.8	0	0.74
	ADASYN	0.97	0.98	0.02	0.66
Gradient Boosting	*Without Sampling*	1	1	0.21	0.96
	SMOTE	0.96	0.98	0.04	0.96
	RandomUnderSampler	0.81	0.9	0.01	0.96
	ADASYN	0.96	0.98	0.04	0.96
LightGBM	*Without Sampling*	1	1	0.3	0.9
	SMOTE	0.99	1	0.13	0.96
	RandomUnderSampler	0.86	0.92	0.01	0.97
	ADASYN	0.99	1	0.12	0.96
XGBoost	***Without Sampling***	**1**	**1**	**0.41**	**0.97**
	SMOTE	0.99	0.99	0.08	0.96
	RandomUnderSampler	0.87	0.93	0.01	0.97
	ADASYN	0.99	0.99	0.1	0.96

4.2 Results Without Sampling

Without applying any sampling method, the classifiers exhibited high F1 scores for Class 0, indicating their effectiveness at identifying the majority class. However, the F1 scores for Class 1 were significantly lower across all classifiers, highlighting the difficulty in correctly identifying the minority class in an imbalanced dataset. For example, RandomForest achieved an F1 Score of 1.00 for Class 0 and 0.37 for Class 1. LogisticRegression performed poorly for Class 1 with an F1 Score of 0.00 while maintaining an F1 Score of 1.00 for Class 0. KNeighbors had an F1 Score of 1.00 for Class 0 and 0.10 for Class 1. GradientBoosting, LightGBM, and XGBoost also showed similar trends, with high F1 Scores for Class 0 (around 1.00) and much lower scores for Class 1 (ranging from 0.21 to 0.41). For detailed values, see Table 2.

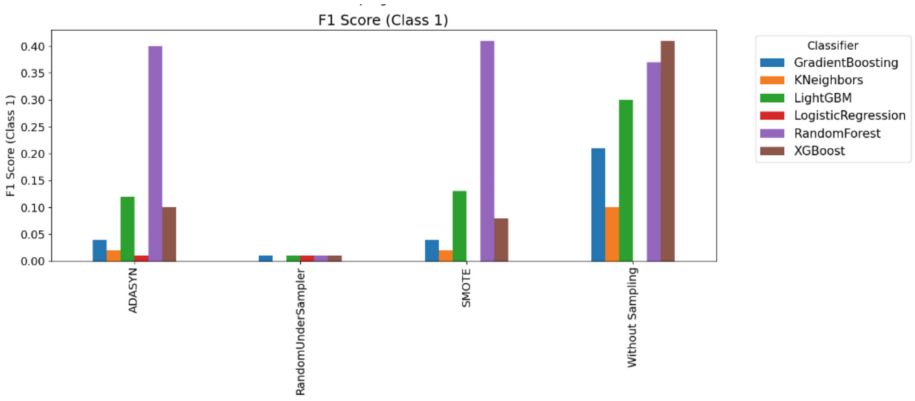


Fig. 2. The comparison F1 Scores of models and sampling methods

4.3 Results with Sampling Methods

Sampling methods were applied to address data imbalance, with notable improvements seen in F1 scores for Class 1 (minority class). Key findings are summarized below, with detailed values available in Table 2:

RandomUnderSampler: This method showed limited improvements for Class 1, with many classifiers exhibiting significant drops in F1 scores for Class 0. For example, RandomForest's F1 score for Class 1 remained at 0.01, with Class 0 slightly reduced to 0.93. Other classifiers, such as LogisticRegression and KNeighbors, also showed low improvements for Class 1.

SMOTE: This method generally boosted F1 scores for Class 1 across most classifiers. RandomForest saw a notable increase in its Class 1 F1 score to 0.41 while maintaining 1.00 for Class 0. LightGBM and XGBoost also saw moderate improvements for Class 1, with F1 scores improving to 0.13 and 0.08, respectively.

ADASYN: ADASYN provided results similar to SMOTE, with RandomForest's F1 score for Class 1 rising to 0.40. Other classifiers like GradientBoosting and LightGBM also showed modest gains for Class 1, though LogisticRegression and KNeighbors showed little improvement. For further details, see Table 2 and Fig. 2.

5 Conclusion

In this study, we evaluated the performance of six classifiers—RandomForest, LogisticRegression, KNeighbors, GradientBoosting, LightGBM, and XGBoost—on an imbalanced dataset for laundering transaction classification. We employed four sampling methods: Without Sampling, SMOTE, RandomUnderSampler, and ADASYN, and assessed their impact on classifier performance, focusing on the F1 scores for the minority class (Class 1).

Our findings indicate that RandomForest, particularly when combined with SMOTE and ADASYN, consistently outperformed other classifiers. Without any sampling, RandomForest achieved an impressive F1 Score of 0.37 for Class 1, which improved to 0.41 with SMOTE and 0.40 with ADASYN. These results demonstrate the robustness of RandomForest in handling imbalanced datasets, particularly when augmented with effective sampling techniques (see Table 2).

Other classifiers, such as LogisticRegression and KNeighbors, struggled to achieve comparable performance, especially in identifying the minority class. Despite various sampling methods, these classifiers maintained low F1 scores for Class 1, indicating limited effectiveness in this context. Ensemble methods like GradientBoosting, LightGBM, and XGBoost showed moderate improvements with sampling methods but did not reach the performance levels of RandomForest.

The application of sampling methods proved crucial in improving the classifiers' ability to detect minority class instances. SMOTE and ADASYN, in particular, were effective in enhancing the F1 scores for Class 1 across most classifiers, with RandomForest benefiting the most. RandomUnderSampler, on the other hand, yielded mixed results, often leading to a decrease in F1 scores for the majority class without significant gains for the minority class.

In conclusion, while techniques such as Random Forest and Logistic Regression are well-established, this study highlights their effectiveness in addressing class imbalance in AML detection—a context that remains underexplored. The findings underscore the critical importance of selecting appropriate combinations of classifiers and sampling methods to improve classification performance in imbalanced datasets [40]. Future research should focus on further model tuning, the development of advanced imbalance handling strategies, and the exploration of additional algorithms to enhance detection performance. These insights contribute to the development of more robust predictive models for financial fraud detection.

Acknowledgements. The authors would like to thank FPT University, CanTho Campus for funding this research.

Disclosure of Interests. . The authors declare no competing interests relevant to the content of this article.

References

1. Levi, M., Reuter, P.: Money laundering. Crime Justice **34**(1), 289–375 (2006)
2. Sobh, T.S.: An intelligent and secure framework for anti-money laundering. J. Appl. Secur. Res. **15**(4), 517–546 (2020)
3. Omar, N., Johari, Z.A.: An international analysis of FATF recommendations and compliance by DNFBPS. Procedia Econ. Financ. **28**, 14–23 (2015)
4. Remeikienė, R., Gasparėnienė, L., Schneider, F.G.: The definition of digital shadow economy. Technol. Econ. Dev. Econ. Vilnius: Technika **24**(2) (2018)
5. Vogel, B.: Potentials and limits of public-private partnerships against money laundering and terrorism financing. In: Eucrim-the European Criminal Law Associations' Forum, no. 01, pp. 52–60 (2022)
6. Berg, S., McCarthy, K.J.: An introduction to the Challenges of Money Laundering. The Money Laundering Market, pp. 3–32 (2018)
7. Nance, M.T.: The regime that FATF built: an introduction to the financial action task force. Crime Law Soc. Chang. **69**, 109–129 (2018)
8. Bamberger, K.A.: Technologies of compliance: risk and regulation in a digital age. Tex. L. Rev. **88**, 669 (2009)
9. Bakhshinejad, N., Soltani, R., Nguyen, U., Messina, P.: A survey of machine learning based anti-money laundering solutions. Researchgate preprint. Accessed, vol. 5 (2023)
10. Johnson, J.M., Khoshgoftaar, T.M.: Survey on deep learning with class imbalance. J. Big Data **6**(1), 1–54 (2019)
11. Thejas, G., Hariprasad, Y., Iyengar, S., Sunitha, N., Badrinath, P., Chennupati, S.: An extension of synthetic minority oversampling technique based on kalman filter for imbalanced datasets. Mach. Learn. Appl. **8**, 100267 (2022)
12. Naeem, S., Ali, A., Anam, S., Ahmed, M.M.: An unsupervised machine learning algorithms: comprehensive review. Int. J. Comput. Digit. Syst. (2023)
13. Dhankhad, S., Mohammed, E., Far, B.: Supervised machine learning algorithms for credit card fraudulent transaction detection: a comparative study. In: 2018 IEEE International Conference on Information Reuse and Integration (IRI), pp. 122–125. IEEE (2018)
14. Khalid, A.R., Owoh, N., Uthmani, O., Ashawa, M., Osamor, J., Adejoh, J.: Enhancing credit card fraud detection: an ensemble machine learning approach. Big Data Cogn. Comput. **8**(1), 6 (2024)
15. Almhaithawi, D., Jafar, A., Aljnidi, M.: Example-dependent cost-sensitive credit cards fraud detection using SMOTE and Bayes minimum risk. SN Appl. Sci. **2**, 1–12 (2020)
16. Kute, D.V., Pradhan, B., Shukla, N., Alamri, A.: Deep learning and explainable artificial intelligence techniques applied for detecting money laundering–a critical review. IEEE Access **9**, 82300–82317 (2021)
17. Segovia-Vargas, M.-J.: Money laundering and terrorism financing detection using neural networks and an abnormality indicator. Expert Syst. Appl. **169**, 114470 (2021)
18. Priscilla, C.V., Prabha, D.P.: Influence of optimizing XGBoost to handle class imbalance in credit card fraud detection. In: 2020 Third International Conference on Smart Systems and Inventive Technology (ICSSIT), pp. 1309–1315. IEEE (2020)
19. Ramyachitra, D., Manikandan, P.: Imbalanced dataset classification and solutions: a review. Int. J. Comput. Bus. Res. (IJCBR) **5**(4), 1–29 (2014)
20. LemaÃŽtre, G., Nogueira, F., Aridas, C.K.: Imbalanced-learn: a python toolbox to tackle the curse of imbalanced datasets in machine learning. J. Mach. Learn. Res. **18**(17), 1–5 (2017)

21. Makki, S., Assaghir, Z., Taher, Y., Haque, R., Hacid, M.-S., Zeineddine, H.: An experimental study with imbalanced classification approaches for credit card fraud detection. IEEE Access **7**, 93010–93022 (2019)
22. Gong, P., Gao, J., Wang, L.: A hybrid evolutionary under-sampling method for handling the class imbalance problem with overlap in credit classification. J. Syst. Sci. Syst. Eng. **31**(6), 728–752 (2022)
23. Mohammed, R., Rawashdeh, J., Abdullah, M.: Machine learning with oversampling and undersampling techniques: overview study and experimental results. In: 2020 11th International Conference on Information and Communication Systems (ICICS), , pp. 243–248. IEEE (2020)
24. Xie, X., Liu, H., Zeng, S., Lin, L., Li, W.: A novel progressively undersampling method based on the density peaks sequence for imbalanced data. Knowl.-Based Syst. **213**, 106689 (2021)
25. Chawla, N.V., Bowyer, K.W., Hall, L.O., Kegelmeyer, W.P.: SMOTE: synthetic minority over-sampling technique. J. Artif. Intell. Res. **16**, 321–357 (2002)
26. Dablain, D., Krawczyk, B., Chawla, N.V.: DeepSMOTE: fusing deep learning and SMOTE for imbalanced data. IEEE Trans. Neural Netw. Learn. Systems **34**(9), 6390–6404 (2022)
27. Maulidevi, N.U., Surendro, K.: SMOTE-LOF for noise identification in imbalanced data classification. J. King Saud Univ. Comput. Inf. Sci. **34**(6), 3413–3423 (2022)
28. He, H., Bai, Y., Garcia, E.A., Li, S.: ADASYN: Adaptive synthetic sampling approach for imbalanced learning. In: 2008 IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence), pp. 1322–1328. IEEE (2008)
29. Brandt, J., Lanzén, E.: A comparative review of SMOTE and ADASYN in imbalanced data classification (2021)
30. Ngai, E.W., Hu, Y., Wong, Y.H., Chen, Y., Sun, X.: The application of data mining techniques in financial fraud detection: a classification framework and an academic review of literature. Decis. Support. Syst. **50**(3), 559–569 (2011)
31. Jullum, M., Løland, A., Huseby, R.B., Ånonsen, G., Lorentzen, J.: Detecting money laundering transactions with machine learning. J. Money Laundering Control **23**(1), 173–186 (2020)
32. Pedregosa, F., et al.: Scikit-learn: machine learning in Python. J. Mach. Learn. Res. **12**, 2825–2830 (2011)
33. Molin, S.: Hands-On Data Analysis with Pandas: Efficiently perform data collection, wrangling, analysis, and visualization using Python. Packt Publishing Ltd (2019)
34. Breiman, L.: Random forests. Mach. Learn. **45**, 5–32 (2001)
35. Cox, D.R.: The regression analysis of binary sequences. J. R. Stat. Soc. Ser. B Stat Methodol. **20**(2), 215–232 (1958)
36. Fix, E., Hodges, J.L.: Discriminatory Analysis. Nonparametric Discrimination: Small Sample Performance. Report A, vol. 193008 (1951)
37. Friedman, J.H.: Greedy function approximation: a gradient boosting machine. Ann. Stat. 1189–1232 (2001)
38. Ke, G., et al.: Lightgbm: a highly efficient gradient boosting decision tree. Adv. Neural Inf. Process. Syst. **30** (2017)
39. Chen, T., Guestrin, C.: Xgboost: a scalable tree boosting system. In: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 785–794 (2016)
40. Saito, T., Rehmsmeier, M.: The precision-recall plot is more informative than the ROC plot when evaluating binary classifiers on imbalanced datasets. Plos One **10**(3), e0118432 (2015)