INFORMATICA, 2025, Vol. 0, No. 0, 1–34
© 2025 *Vilnius University*
DOI: https://doi.org/10.15388/25-INFOR598

Anti-Money Laundering Compliance Using Feature Engineering with SQL Analytics, TF-IDF and Oversampling: Conditional Tabular Generative Adversarial Networks

Anca Ioana ANDREESCU[1], Simona-Vasilica OPREA[1], Alin Gabriel VĂDUVA[1,2,*], Adela BÂRA[1]

[1] *Department of Economic Informatics and Cybernetics, Bucharest University of Economic Studies, 010374 Bucharest, Romania*
[2] *Doctoral School of Economic Informatics, Bucharest University of Economic Studies, 010374 Bucharest, Romania*
e-mail: anca.andreescu@ie.ase.ro, simona.oprea@csie.ase.ro, alin.vaduva@csie.ase.ro, bara.adela@ie.ase.ro

Received: March 2025; accepted: June 2025

Abstract. Traditional Anti-Money Laundering (AML) systems rely on rule-based approaches, which often fail to adapt to evolving money laundering tactics and produce high false-positive rates, overwhelming compliance teams. This study proposes an innovative machine learning (ML) framework that leverages Conditional Tabular Generative Adversarial Networks (CTGANs) to address severe class imbalance, a common challenge in Suspicious Activity Reporting (SAR). Implemented in Python, CTGAN generates realistic synthetic samples to enhance minority-class representation, improving recall and F1-scores. For instance, the Random Forest (RF) model achieves a recall of 0.991 and an F1-score of 0.528 in oversampled datasets with engineered variables, highlighting the effectiveness of CTGAN in mitigating imbalance. This framework also incorporates SQL-based feature engineering using Oracle Analytics, creating dynamic variables such as cumulative sums, rolling averages, and ranks. The modelling phase and exploratory data analysis are conducted in the SAS programming language, employing Logistic Regression (LR) as baseline, Decision Trees (DT), and RF. Evaluation across undersampled and oversampled datasets, combined with varying probability thresholds, reveals key trade-offs between sensitivity and precision. Among the models, RF consistently achieves the highest ROC-AUC scores, ranging from 0.945 in undersampled datasets to 0.951 in oversampled configurations, demonstrating its robustness and accuracy in SAR detection. By integrating CTGAN and TF-IDF (textual feature transformation in Python) with SQL-engineered variables, this framework provides a comprehensive data-driven approach to AML. It reduces false positives, strengthens the detection of suspicious activities, and ensures scalability, adaptability, and compliance with regulatory standards.

Key words: anti-money laundering, synthetic data generation, SAS-Python, SQL analytics, TF-IDF.

[*]Corresponding author.

1. Introduction

A key aspect of any AML compliance program is the continuous monitoring of transactions for suspicious activities. The transaction landscape is broad, encompassing a wide range of financial activities such as deposits, withdrawals, fund transfers, purchases, loans and payments (Antwi *et al.*, 2023). Monitoring these transactions is important to identifying potential money laundering schemes, which often involve complex patterns designed to evade detection (Saragih, 2023; Gilmour, 2023). Traditionally, this monitoring begins with a rule-based system that scans customer transactions for red flags, typically based on predetermined thresholds and conditions that are consistent with known money laundering techniques. For instance, large cash deposits or frequent cross-border transfers may trigger alerts. Once a transaction matches one of these predefined rules, an alert is generated and sent to the bank's internal investigation team for further scrutiny. If investigators, after reviewing the case, determine that the behaviour is indicative of money laundering, a SAR is filed with the appropriate regulatory authorities.

While rule-based systems have been the cornerstone of AML transaction monitoring for years, they are not without significant limitations (Ketenci *et al.*, 2021). One of the primary drawbacks is the high rate of false positives: ordinary, legitimate transactions that are flagged as suspicious due to the rigid and often oversimplified nature of the rules. False positives occur when the system misclassifies normal financial behaviour as high-risk based on generalized criteria, such as arbitrary transaction thresholds. The inefficiency places a considerable burden on financial institutions, as compliance teams must manually review these flagged transactions, consuming time and resources (Benzerrouk *et al.*, 2023; Ahmad Tarmizi *et al.*, 2023). The consequences of relying heavily on traditional systems lead to operational inefficiencies, increased compliance costs and delayed identification of actual money laundering activities (Pavlidis, 2023).

Moreover, these rule-based systems are often static and reactive, failing to adapt to evolving money laundering tactics. Fraudsters continually modify their strategies, making it difficult for static rules to keep pace with the dynamic nature of financial crime. In response to these limitations, the financial industry is turning to ML and statistical models, which offer a more proactive and adaptive approach to transaction monitoring. These models have the potential to learn from vast amounts of data, identify subtle patterns and correlations and continuously evolve with new information. By capturing more nuanced relationships between transactions and risk, ML systems significantly reduce the number of false positives while improving the identification of genuinely suspicious activities (Singh and Best, 2019).

The implementation of ML in AML systems aims to address the critical weaknesses of traditional methods (Labanca *et al.*, 2022) by leveraging algorithms capable of analysing large datasets, recognizing complex behavioural patterns and predicting potential money laundering with a higher degree of accuracy (Jensen and Iosifidis, 2023). Usually, these models assess multiple variables simultaneously, going beyond simplistic rules to provide a more comprehensive risk assessment. As a result, financial institutions better allocate resources to truly suspicious cases, reducing the strain on compliance teams while improving overall effectiveness in detecting illicit activities.

However, banking transactions involve highly sensitive and confidential information, which presents significant challenges for researchers (Jensen and Iosifidis, 2023), as access to individual transaction data is restricted to ensure privacy and security. Direct access to personal financial details is typically prohibited due to regulatory and ethical considerations surrounding data protection. As a result, researchers often rely on aggregated data sets rather than individual-level records. These aggregated datasets provide a summary of financial activity over a specified period, such as the past 90 days, and represent collective trends rather than detailed, client-specific transactions. By using this temporal grouping, researchers may analyse broader patterns in financial behaviour without compromising the privacy of individual clients.

In this paper, we propose a framework for detecting fraud in the banking system by integrating SQL-based feature engineering, Natural Language Processing (NLP) for textual data transformation and synthetic data generation using CTGAN. Feature engineering, performed using SQL analytics, creates dynamic variables such as cumulative metrics, rolling averages and ranks, adding more behavioural insights. Textual data, such as customer service notes, is transformed using the TF-IDF method in Python, providing a structured representation for ML models. CTGAN, also implemented in Python, generates synthetic customer profiles that closely mimic real data, effectively addressing the issue of severe class imbalance in SAR. All preprocessing operations, including feature engineering, text transformation, and CTGAN-based oversampling, are performed exclusively on the training set, leaving the testing set untouched to ensure a realistic, leakage-free evaluation of model performance. The generated synthetic data represents new, unseen fraud patterns, enhancing model generalization and improving recall in rare-event detection. The methodology is evaluated across various scenarios, including undersampled and oversampled datasets, with and without SQL-engineered variables, under varying probability thresholds. The modelling phase and exploratory data analysis are conducted in SAS programing, employing LR, DT and RF. Thus, the features created in Python and Oracle database are analysed and trained in SAS. Our results demonstrate the impact of combining CTGAN-generated synthetic data and SQL-engineered features, yielding models that reduce false positives, enhance fraud detection and ensure compliance with evolving regulatory standards.

By addressing three key research questions (RQ), we examine the influence of the proposed preprocessing techniques on models performance. Additionally, we evaluate varying probability thresholds to balance false positives and the accurate detection of high-risk clients, aiming to create a scalable and interpretable framework for AML systems.

RQ1: What is the contribution of feature engineering using SQL analytic functions and TF-IDF on text notes for improving the predictive accuracy of ML models for AML detection?

RQ2: How does oversampling with CTGAN influence class imbalance issues and models performance, particularly in detecting rare suspicious activity scenarios?

RQ3: How do different probability thresholds (0.5 vs. 0.115) affect the precision, recall, F1 score and ROC-AUC across undersampled and oversampled datasets with and without engineered variables?

This paper is structured in several sections. In this section, the general context, motivation, challenges, research questions, objectives and contributions are underlined, whereas in the second section, a brief literature review is provided. The proposed framework is presented in Section 3. Section 4 is dedicated to the main findings, Section 5 discusses the implications of these findings and in Section 6 the conclusions are drawn.

2. Literature Review

2.1. *Previous Research in AML*

The impact of AML regulations on global financial sector development from 2012 to 2018, covering 165 economies, was examined (Ofoeda *et al.*, 2022). The research explored whether this effect varies between developing and developed nations and investigated nonlinearities in the AML-financial sector development relationship. Using Prais-Winsten and panel threshold estimation approaches, the findings showed that AML regulations generally boost financial sector development, especially in developing countries. The positive impact was primarily observed in nations below a certain AML regulation threshold, indicating that strengthening AML measures benefits developing economies. Moreover, the extent to which financial facilitators in the Netherlands exhibit business-like behaviours and how they organize into money laundering networks were explored (Kramer *et al.*, 2023). Using police intelligence data on 198 facilitators involved with drug criminals between 2016–2020, the analysis revealed that these facilitators form extensive money laundering networks.

How illicit funds are laundered using the gold trade in German-speaking European countries was examined (Teichmann and Falker, 2023). Through 60 semi-standardized interviews with both money launderers plus compliance officers and a survey of 200 compliance officers, the research found that the gold trade is highly suitable for money laundering, particularly for placement and layering. Additionally, Jensen and Iosifidis (2023) introduced a terminology with two main components: (a) client risk assessment and (b) detection of suspicious behaviour. Client risk assessment involves analysing and explaining potential risk factors, whereas suspicious behaviour detection uses undisclosed attributes and custom risk metrics. An issue is the scarcity of publicly available datasets, which could be mitigated through synthetic data.

Another research presented a longitudinal case study of a UK bank's efforts to enhance its money laundering detection by broadening its profiling of behaviours (Demetis, 2018). Using the concept of structural coupling from systems theory, the researcher reflected on the bank's approach to profiling. The security risks posed by banditry and terrorism in Nigeria were addressed (Chitimira and Animashaun, 2023). These funds were laundered through channels such as bureau de change, exploiting flaws in Nigeria's AML and anti-terrorism laws. Using a doctrinal and qualitative research method, the research analysed these legal shortcomings and suggests measures to strengthen Nigeria's efforts to combat money laundering and terrorist financing. Furthermore, Korejo *et al.* (2021)

traced the evolution of money laundering laws. It found that the broadening of predicate offenses, from drug money to corruption and terrorist financing, led to concerns of over-criminalization and conflicts with criminal law principles. Moreover, researchers and practitioners have explored how financial institutions assess money laundering risks, since the 1980s, noting a tendency to rely on box-ticking rather than case-by-case judgment (Ogbeide *et al.*, 2023). This approach raised questions about whether experts are immune to cognitive biases that novices face during risk assessments. It found that both experts and novices displayed overconfidence in their distribution judgments, with experts being slightly more prone to this effect.

2.2. *Reviews in the AML*

The growing issue of money laundering and its harmful effects on the global economy and society were addressed (Isolauri and Ameer, 2023). Despite its significance, international business research on the topic remains limited and dispersed across disciplines. The researchers conducted a systematic review of 57 studies from the past two decades, identifying five key research streams. They also highlighted six theoretical approaches, with normative standards and business/economics theories being the most common. Another research reviewed the literature on money laundering, aiming to identify research gaps and guide future investigations (Tiwari *et al.*, 2020), also identifying six broad themes.

Few methods addressed both money laundering and financial fraud together (Goecks *et al.*, 2022). This research aimed to identify techniques for AML and financial fraud detection through a systematic literature review using SCOPUS and Web of Science databases. Of 48 relevant articles, 20 used quantitative methods, 13 were literature reviews, 7 employed qualitative methods and 8 used mixed approaches. Furthermore, Salehi *et al.* (2017) studied reviews fraud detection research, focusing on money laundering and the limitations of current data mining techniques, which typically rely on predefined rules and thresholds. It highlighted the effectiveness of data mining in identifying unusual behaviours and suggested that unsupervised data mining techniques may better detect new money laundering patterns. Additionally, Chen *et al.* (2018) surveyed ML algorithms and methods used to detect suspicious transactions, focusing on AML typologies, link analysis, behavioural modelling, risk scoring, anomaly detection and geographic analysis. It reviewed key steps in data preparation, transformation and analytics, categorizing and comparing existing ML techniques.

2.3. *Blockchain Technology and AML*

As financial institutions transition from the SWIFT network to blockchain, they must revise money laundering detection practices to adapt to this new paradigm (Jovicic and Tan, 2018). The efficiency of blockchain speeded up transactions, requiring more advanced money laundering detection methods. This research explored how blockchain applies to electronic fund transfers and examines ML techniques for detecting money laundering. Thommandru and Chakka (2023) explored how the banking sector is exploited for money laundering and terrorist financing, noting the burden of compliance with strict AML laws. It suggested that emerging technologies, particularly blockchain, can mitigate financial crimes by transforming processes like peer-to-peer payments and trade agreements. Blockchain's ability to enhance Know Your Customer (KYC) verification and recalibrate compliance policies could reduce the financial strain on banks while improving AML measures. Additionally, Oad *et al.* (2021) introduced a blockchain-enabled transaction scanning (BTS) method to detect anomalous actions in financial transactions. The BTS method sets rules for outlier detection and rapid fund movements, identifying patterns of malicious activities. It scanned transaction histories to flag suspicious entities and uses blockchain to prevent money laundering.

2.4. *Crypto and AML*

Money laundering in cryptocurrency transactions differs from traditional financial crimes due to its anonymity and decentralization, making conventional AML techniques unsuitable (Zhong *et al.*, 2022). This research proposed a four-stage money laundering detection approach tailored to cryptocurrency. Experimental results on a real-world dataset demonstrated high accuracy: 96.02%, 95.05%, 95.83% and 95.81% for detecting abnormal behaviours, suspected launderers, loud and subtle transactions. With the rise of the crypto economy, cryptocurrency is seen as a potential vehicle for money laundering (Wang and Hsieh, 2024). They analysed the features of cryptocurrency, like anonymity and decentralization, that contribute to its appeal for money laundering. A money laundering triangle was introduced within a criminological framework. It recommended that future AML strategies should focus on cryptocurrency's characteristics to deter its use in laundering activities. Liu *et al.* (2023) introduced GTN2vec, an improved graph embedding algorithm specifically for detecting money laundering on Ethereum. By analysing transaction records, GTN2vec captured the behavioural patterns of money launderers and the structure of transaction networks.

Money laundering through Bitcoin has become a threat (Yu *et al.*, 2023). Traditional detection methods rely on fixed rules, limiting accuracy and scalability. To address this, the research proposed AEtransGAT, an approach for detecting money laundering by mining Bitcoin transaction records. AEtransGAT used transGat as an encoder to assess the importance of surrounding transactions based on transaction flows and a graph autoencoder as the decoder to capture structural information. By combining transaction classification and structure reconstruction, the model improved detection. Moreover, researchers responded by developing new detection techniques (Al Badawi and Al-Haija, 2021). They presented an AML system that used ML, specifically shallow Neural Networks (NN) and decision trees, to classify licit and illicit transactions. Evaluated on the Bitcoin dataset, the models achieved accuracies of 89.9% and 93.4%.

2.5. *Statistics, ML and Deep Learning in AML*

With an estimated \$800 billion to \$2 trillion money laundered annually, including \$5 billion through cryptocurrency, the Financial Action Task Force highlighted how criminals may convert illegally obtained fiat money into cryptocurrency, posing a significant challenge in detecting and preventing illegal transactions (Alotibi *et al.*, 2022). This research explored the use of deep learning and ML for detecting suspicious cryptocurrency transactions, employing Deep NN (DNN), RF, K-Nearest Neighbours (KNN) and Naive Bayes (NB) on the Bitcoin dataset. Results show that DNN and RF achieved the highest accuracy. Moreover, Inspection-L, a graph neural network (GNN) framework utilizing self-supervised Deep Graph Infomax (DGI) and Graph Isomorphism Network (GIN), combined with supervised learning algorithms like RF was introduced (Lo *et al.*, 2023), to detect illicit transactions for AML.

The dynamic nature of information systems has weakened traditional detection mechanisms (Caglayan and Bahtiyar, 2022). This research explored ML algorithms as complementary solutions, focusing on graph-based data representation using Node2Vec to improve classification for money laundering detection. Experimental results showed that Node2Vec helps identify the most effective ML algorithms for detecting money laundering. Yang *et al.* (2023) presented a two-tier algorithm combining heuristic rules and integrated learning techniques to detect money laundering in virtual currencies. The algorithm established heuristic rules based on statistical risk attributes and employs a model combining Long Short-Term Memory (LSTM) and graph convolutional NN to identify suspicious patterns. A hard voting mechanism further enhanced detection by integrating classifiers like Histogram-Based Outlier Scoring and Isolation Forest. With monitoring financial transactions in AML, the ML-based systems increasingly complemented traditional rule-based methods to reduce false positives and manual review (Labanca *et al.*, 2022). However, ML models face challenges: unsupervised models detect novel patterns but generate many false alarms, while supervised models have higher accuracy but require substantial labelled data.

Additionally, Huong *et al.* (2024) introduced a novel approach to improve detection by constructing network graphs from bank transaction datasets. They were transformed into directed node representations that encode relationships and community structures within the network. A RF model was also used to predict suspicious behaviours. To address class imbalance, oversampling and undersampling techniques were applied, with undersampling yielding the highest accuracy at 92%, compared to 86% with oversampling. Another research explored the interplay between ML and sampling techniques in detecting money laundering through an empirical analysis using real transaction data from a U.S. financial institution (Zhang and Trubey, 2019). It evaluated five ML algorithms: Bayes LR, DT, RF, support vector machine (SVM) and ANN.

To address traditional AML systems, a KNN model was developed using open financial transaction datasets from Kaggle (Hampo *et al.*, 2023). The model achieved an accuracy of 98.4%. Another research addressed the failure of traditional models, including SVM, NN and KNN, to detect simulated money laundering accounts from the Panama Papers dataset (Sheu and Li, 2022). A new money laundering detection tool, a graph attention network, was developed. It has three modules: a feature extraction module that encodes transaction data into a weighted graph, a graph attention module that uses a self-attention mechanism to highlight suspicious nodes and a classification module that filters targets using a rectified linear unit function. It outperformed existing methods, including Naïve Bayes and RF.

AML analysis usually requires processing large volumes of data, such as billings and bank transactions, to support investigations (Drezewski *et al.*, 2015). To assist human analysts, the Money Laundering Detection System (MLDS) was proposed as a software tool. This research introduced a social network analysis component for MLDS, which utilizes data from bank statements and the National Court Register to construct and analyse social networks in money laundering investigations. The system assigned roles to individuals within the network and analysed their connections. While existing methods such as ML, graph mining and anomaly detection have been applied, they often fail to account for the dynamic characteristics of transactions that could aid in detection (Luo *et al.*, 2022). To address it, the research proposed a dynamic transaction pattern aggregation neural network (DTPAN) for money laundering detection. DTPAN uses two feature extractors to capture the dynamic features of transaction behaviours and the evolving relationships between accounts. A feature enhancement module further strengthened the dynamic behaviour features by identifying latent dependencies between behaviour dynamics and relationship evolution.

Another research proposed a new system for detecting money laundering by comparing tax data, particularly value-added tax (VAT), with banking transactions (Bidabad, 2017). The MLDS, part of the Rastin Banking system but operable independently, helped identify financial deception and fraud. It worked by requiring all transactions to go through banks, then comparing the tax data of transactors with their banking transactions. Discrepancies between these datasets may reveal money laundering activities. Cheng *et al.* (2023) introduced a group-aware deep graph learning approach to detect organized money laundering. A community-centric encoder modelled user transaction graphs and identified group-level interactions, while a local enhancement scheme aggregated similar transaction features into gangs. Tests on a major bank card alliance dataset showed this method outperformed existing approaches. Kannan and Somasundaram (2017) proposed an autoregressive (AR) outlier-based money laundering detection system to reduce the time needed to process large, non-uniform transactions. The AR model enhanced demand forecasting, while inter-quartile range (IQR) formulations aided in analysing time-series data. The study found that detecting outliers in high-dimensional data and complex time-series relationships can be challenging.

The current rule-based detection systems proved highly ineffective, with over 90% false positives (Ketenci *et al.*, 2021). Thus, the researchers introduced a novel feature set using time-frequency analysis, creating 2-D representations of financial transactions. An RF model with simulated annealing for hyperparameter tuning was tested on real banking data, demonstrating that time-frequency features significantly enhance detection performance. A comprehensive model to enhance self- and group-comparisons for detecting suspicious transactions related to money laundering and terrorism financing in financial systems was proposed (Rocha-Salazar *et al.*, 2021). Self-comparisons were improved by expanding KYC policies, incorporating non-transactional characteristics into four categories: inherent, product, transactional and geographic. Group-comparisons were enhanced using an innovative transaction abnormality indicator based on variable variance. The model significantly reduces false positives and improves accuracy compared to rule-based systems, leading to reduced investigation costs for suspicious customers. Unsupervised learning is better at detecting irregularities, though it often lacks state-of-the-art accuracy (Chen *et al.*, 2021). The research proposed a system using unsupervised and deep learning models: autoencoder (AE), variational autoencoder (VAE) and a Wasserstein GAN (WGAN). The WGAN generated synthetic fraud transactions to balance the dataset, which is then used to train AE and VAE models. Two versions of the AE model were tested: single-loss and multi-loss, along with a novel anomaly score thresholding method, Recall-First Threshold (RFT). Experimental results showed a reduction in the false positive rate to 7% with the multi-loss AE model. A tabular comparison, summarizing the key details from the previous studies, is provided in Table A1 Appendix A.

3. Methodology and Data

The banking industry faces increasing regulatory requirements and compliance demands, particularly concerning AML efforts. Traditional transaction monitoring systems, predominantly rule-based frameworks relying on IF-ELSE conditions, are widely employed to identify suspicious activities. However, these systems have significant limitations, including their inability to adapt to evolving fraud patterns and their propensity to generate a high rate of false positives-normal transactions flagged as suspicious. This high rate of false positives not only increases the operational costs for banks but also diverts resources away from genuinely suspicious cases, thereby reducing the overall efficiency of the AML process. Thus, there is a pressing need to develop advanced statistical systems that go beyond rule-based approaches. By leveraging ML techniques, such systems dynamically identify high-risk clients with greater accuracy while minimizing the number of false positive cases. It is important to note that the dataset is cross-sectional: all variables are aggregated over the same 90-day observation window, so no temporal or panel structure exists. Consequently, time-ordered train/test splits are not applicable; instead, we prevent data leakage by performing a random stratified split before any imputation, feature engineering, or oversampling.

In this section, we describe the methodology for implementing a ML-based system to enhance AML. Our research is conducted using a synthetic dataset sourced from the DataRobot[1] website, specifically from their AML model demo. This dataset simulates a credit card company's AML compliance program, focusing on scenarios such as customers overpaying their credit card bills to request cash refunds and customers receiving merchant credits without offsetting transactions, then spending the credited amount or requesting cash refunds. The unit of analysis is an individual alert, generated by a rule-based engine, with a binary target variable, SAR, indicating whether a Suspicious Activity Report was filed.

[1] https://docs.datarobot.com/en/docs/get-started/gs-dr5/biz-accelerators/money-launder.html

3.1. *Dataset Splitting*

The dataset is split into training and testing sets to ensure robust evaluation of the ML models. We use the SURVEYSELECT procedure in SAS, which employs simple random sampling, ensuring that the training set comprises 70% of the data and the testing set 30%. A seed parameter set to 123 is specified for reproducibility across experiments.

3.2. *Data Preprocessing*

This section outlines the preprocessing steps applied to the dataset, including feature encoding, missing data imputation, feature engineering and oversampling using CTGAN over the training dataset.

3.2.1. *Feature Encoding*

To make the dataset suitable for ML algorithms, categorical and textual variables are encoded into numerical formats. Specifically:

a) Frequency encoding applied to the categorical variable "state" to encapsulate the relative frequency of each state within the dataset. This encoding provides a numeric value, reflecting the occurrence of each category. Let N denote the total number of instances and $f(s)$ the frequency of state s. Then, the encoded value for s is calculated as:

$$\text{Encoded Value for } s = \frac{f(s)}{N}. \tag{1}$$

b) Term frequency-inverse document frequency (TF-IDF) applied to textual variables, which contains textual customer service notes. TF-IDF captures the importance of each term t in document d within the corpus C.

$$\textit{TF-IDF}(t, d) = \textit{TF}(t, d) \cdot \textit{IDF}(t, C). \tag{2}$$

where:

$$\textit{TF}(t, C) = \frac{\text{Number of occurences of } t \text{ in } d}{\text{Total number of terms in } d}, \tag{3}$$

$$\textit{IDF}(t, C) = \log\left(\frac{|C|}{1 + |\{d \in C : t \in d\}|}\right). \tag{4}$$

c) Binary encoding for binary variables, mapping Y to 1 and N to 0.

3.2.2. *Missing Data Imputation*

To address missing data, we utilize multiple imputation (MI) in SAS programming. MI replaces missing values with multiple plausible estimates, reflecting the uncertainty associated with the imputation process. This method creates m complete datasets, analyses each dataset separately and then combines the results.

For a given variable X with missing values, the first step (imputation step) is represented by replacing missing values in X, m times using plausible estimates derived from the observed data distribution. In step 2 (analysis step), each imputed dataset is analysed separately. In the pooling phase, the final estimates and standard errors are obtained by combining results across all imputed datasets using Rubin's rules:

$$\overline{\theta} = \frac{1}{m}\sum_{i=1}^{m}\hat{\theta}_i, \tag{5}$$

$$Var = \overline{U} + \left(1 + \frac{1}{m}\right)B, \tag{6}$$

where $\overline{\theta}$ is the pooled estimate, $\overline{U}$ is the within-imputation variance, B is the between-imputation variance and Var is the total variance of the parameter estimates.

3.2.3. *Feature Engineering*

Feature engineering is performed using SQL Oracle analytic functions to generate additional variables that provide more insights into the dataset. The following variables are engineered:

1. Cumulative sum by group computes the cumulative sum of a numeric column x for each group g, ordered by a specified variable. It provides insights into the running total of a quantity within each group, allowing the identification of patterns in cumulative behaviour.

$$CumulativeSum(i, g) = \sum_{j=1}^{i} x_j. \tag{7}$$

2. Rank within group ranks rows within a group g based on a specified column x, in ascending or descending order. It is used to identify the relative position of a record within a group, such as identifying the top performers or contributors.

$$Rank(i, g) = Rank_g(-x_i). \tag{8}$$

3. *Difference between consecutive rows* calculates the difference between the current row's value x_i and the previous row's value x_{i-1}, useful for identifying changes or trends in sequential data, such as the growth or decline of a quantity over time.

$$Difference(i) = x_i - x_{i-1}. \tag{9}$$

4. *Rolling average within group* computes the average of a numeric column x over a rolling window of k rows, within a group g, ordered by a specified variable, providing a smoothed view of trends over time or within a sequence, reducing the impact of short-term fluctuations.

$$RollingAvg_i(g) = \frac{\sum_{j=i-k+1}^{i} x_j}{k}. \tag{10}$$

5. *Global rank* is used for ranking rows globally based on a numeric column x in ascending or descending order, independent of grouping. It is used to identify the relative position of a record across the entire dataset, such as determining overall performance or priority.

$$Rank(i) = Rank(-x_i). \tag{11}$$

6. *Cumulative average* calculates the cumulative average of a numeric column x across all rows up to the current row r. It provides insights into the overall average trend as more data points are considered.

$$CumulativeAvg_r = \frac{\sum_{j=1}^{k} x_j}{r}. \tag{12}$$

7. *Cumulative count by group* computes the cumulative count of a numeric column x for each group g, ordered by a specified variable. It provides insights into the cumulative total of a quantity for specific groups.

$$CumulativeCount(i, g) = \sum_{j=1}^{i} x_j. \tag{13}$$

3.2.4. *Oversampling Using Conditional Tabular Generative Adversarial Networks*

Class imbalance is a common challenge in datasets for AML, as suspicious activity cases (SAR = 1) often represent a small fraction of the total observations. To address this imbalance, we employ CTGAN in Python to generate synthetic data that closely mimics the distribution of the original minority class, enriching the dataset with additional examples of suspicious cases. CTGAN is specifically designed to handle the unique challenges of tabular datasets, such as mixed data types, highly imbalanced classes and complex feature-target dependencies.

CTGAN builds upon the traditional GAN framework, consisting of a Generator (G) and a Discriminator (D), but introduces additional mechanisms tailored for tabular data. A key feature of CTGAN is its conditional generation mechanism, which allows the generator to model the relationship between categorical and numerical features and generate realistic synthetic samples conditioned on specific feature values to ensure that the synthetic data accurately reflects the real-world distribution of both the majority and minority classes.

The objective function for CTGAN remains similar to that of traditional GAN, with the Generator and Discriminator engaged in a zero-sum game. The combined objective function is:

$$\min_{G} \max_{D} V(D, G) = \boldsymbol{E}_{x\sim p_{data}}\big[\log D(x)\big] + \boldsymbol{E}_{z\sim p_z, v\sim p(v)}\big[\log\big(1 - D\big(G(z, v)\big)\big)\big], \tag{14}$$

where x represents real data sampled from the true data distribution p_{data}, z is the noise vector sampled from a prior distribution p_Z, v is the conditional vector representing feature values, $G(z, v)$ represents synthetic data generated by the generator conditioned on v and $D(x)$ represents the probability that x is real, predicted by the discriminator.

The Discriminator Loss function measures the ability of the discriminator to distinguish real samples from synthetic ones:

$$\mathcal{L}_D = -\boldsymbol{E}_{x\sim p_{data}}\big[\log D(x)\big] - \boldsymbol{E}_{z\sim p_z, v\sim p_v}\big[\log\big(1 - D\big(G(z, v)\big)\big)\big]. \tag{15}$$

The generator seeks to minimize the discriminator's ability to differentiate real from synthetic samples:

$$\mathcal{L}_G = -\boldsymbol{E}_{z\sim p_z, v\sim p_v}\big[\log D\big(G(z, v)\big)\big]. \tag{16}$$

The CTGAN training process alternates between training the discriminator and the generator. First, the discriminator is trained using real data (x) and synthetic data $G(z, v)$ to maximize its ability to classify real versus synthetic samples. Then, the generator is updated based on feedback from the discriminator, aiming to minimize the discriminator's ability to differentiate real from synthetic samples. The training continues until the generator produces synthetic data that is indistinguishable from real data to the discriminator.

CTGAN uses conditional vector encoding to manage categorical variables effectively. For each categorical variable, a one-hot encoded vector is created to represent all possible categories, while numerical variables are modelled using Gaussian distributions. During training, the conditional vector (v) is sampled alongside the noise vector (z) to guide the generation process, ensuring the synthetic samples preserve realistic feature-target dependencies.

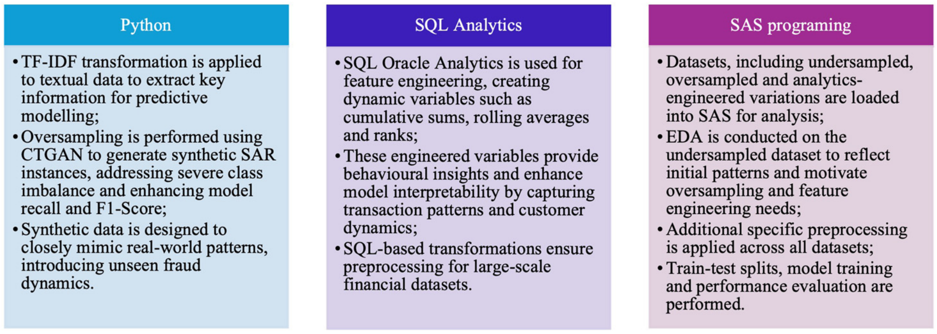
3.3. *Models' Training and Evaluation*

This section describes the process of preparing the data for ML models, training the models and evaluating their performance. Our research employs three ML algorithms: LR, DT and RF to analyse the dataset. Each algorithm is implemented in SAS programming and the training process is performed on the prepared training dataset. The three programming languages employed in this research and the processing stages are presented in Fig. 1.

3.3.1. *Logistic Regression*

LR is a linear model used for binary classification, where the target variable (SAR) represents the probability of an account being suspicious. The model is trained by estimating coefficients β for the input features X to maximize the likelihood of observing the training data. The predicted probability is given by the logistic function:

$$P(SAR = 1 \mid X) = \frac{1}{1 + e^{-\beta^T X}}. \tag{17}$$



Python

- TF-IDF transformation is applied to textual data to extract key information for predictive modelling;
- Oversampling is performed using CTGAN to generate synthetic SAR instances, addressing severe class imbalance and enhancing model recall and F1-Score;
- Synthetic data is designed to closely mimic real-world patterns, introducing unseen fraud dynamics.

SQL Analytics

- SQL Oracle Analytics is used for feature engineering, creating dynamic variables such as cumulative sums, rolling averages and ranks;
- These engineered variables provide behavioural insights and enhance model interpretability by capturing transaction patterns and customer dynamics;
- SQL-based transformations ensure preprocessing for large-scale financial datasets.

SAS programing

- Datasets, including undersampled, oversampled and analytics-engineered variations are loaded into SAS for analysis;
- EDA is conducted on the undersampled dataset to reflect initial patterns and motivate oversampling and feature engineering needs;
- Additional specific preprocessing is applied across all datasets;
- Train-test splits, model training and performance evaluation are performed.

Fig. 1. Overview of the AML detection workflow.

During training, the cost function minimized is the negative log-likelihood:

$$\mathcal{L} = -\frac{1}{N}\sum_{i=1}^{N}\big[y_i \log \hat{y}_i + (1 - y_i)\log(1 - \hat{y}_i)\big], \tag{18}$$

where y_i is the actual label, $\hat{y}_i$ is the predicted probability and N is the number of samples. LR is particularly effective for datasets where features have a linear relationship with the log-odds of the target variable.

3.3.2. *Decision Tree*

A DT is constructed by recursively splitting the dataset into subsets based on feature values, forming a tree structure where each node represents a decision rule. In this study, the splitting criterion used is entropy, which measures the impurity of a node. Entropy is calculated as:

$$H = -\sum_{i=1}^{n_c} p_i \log(p_i), \tag{19}$$

where n_c is the number of classes and p_i is the proportion of samples belonging to class i in a node. A perfectly pure node (all samples belong to one class) has an entropy of 0, while a node with evenly distributed classes has maximum entropy. The algorithm selects the feature and split point that result in the greatest reduction in entropy, measured by information gain:

$$\mathit{Information\ Gain} = H_{parent} - (w_{left}H_{left} + w_{right}H_{right}), \tag{20}$$

where H_{parent} is the entropy of the parent node, H_{left} and H_{right} represent the entropies of the left and right child nodes, whilst w_{left} and w_{right} denotes the proportions of samples in the left and right child nodes.

To prevent overfitting, the tree is pruned using cost complexity pruning, which penalizes the complexity of the tree by adding a regularization term based on the number of terminal nodes.

3.3.3. *Random Forest*

RF is an ensemble learning method that constructs multiple DTs during training and combines their predictions to improve model accuracy and robustness. Each tree is trained on a bootstrap sample of the dataset, where samples are selected randomly with replacement. This process introduces variability across the trees, reducing the risk of overfitting. Additionally, at each split in a tree, only a random subset of features is considered, ensuring diversity in the splitting criteria and further reducing correlation among the trees. The final prediction is made by aggregating the predictions of all trees in the forest. For classification, the model uses majority voting to determine the class. For probabilistic outputs, which are used in this study, the random forest calculates the average probability predicted by individual trees. This can be expressed as:

$$P(SAR = 1) = \frac{1}{T}\sum_{t=1}^{T} P_t(SAR = 1), \tag{21}$$

where T is the total number of trees and P_t is the predicted probability from the t-th tree.

3.3.4. *Classification Metrics*

Several statistical indicators are used to evaluate the generalization capabilities of a model. One of the most useful tools for understanding a model's performance is the confusion matrix, which summarizes the results of classification by categorizing predictions into four categories: true negatives (*TN*), false positives (*FP*), false negatives (*FN*) and true positives (*TP*).

Precision measures the accuracy of positive predictions. It reflects the proportion of instances predicted as positive that are actually positive. This metric is particularly important when the cost of false positives is high. Precision is calculated as:

$$Precision = \frac{TP}{TP + FP}. \tag{22}$$

Recall assesses the model's ability to identify all actual positive cases. It is a widely used metric in scenarios where missing positive cases (false negatives) have serious consequences, such as fraud detection or medical diagnosis. Recall is defined as:

$$Recall = \frac{TP}{TP + FN}. \tag{23}$$

F1 Score provides a harmonic mean between precision and recall, balancing the trade-off between the two metrics. This score is particularly useful when the dataset is imbalanced, as it accounts for both false positives and false negatives.

$$F1\ Score = \frac{2 \cdot Precision \cdot Recall}{Precision + Recall}. \tag{24}$$

The Receiver Operating Characteristic (ROC) curve is a graphical tool used to evaluate the performance of binary classifiers by plotting the True Positive Rate (TPR) against the

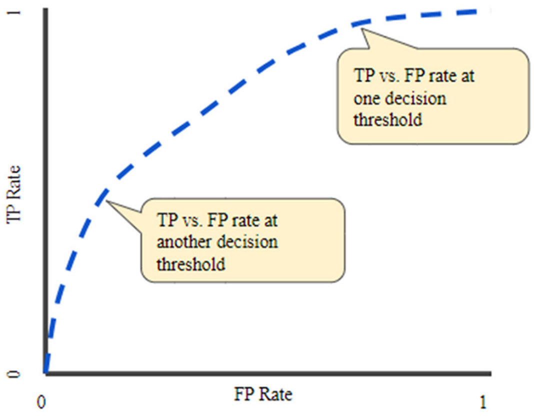

TP Rate
FP Rate
0
1
0
1
TP vs. FP rate at one decision threshold
TP vs. FP rate at another decision threshold

Fig. 2. ROC-AUC curve and the decision threshold.

False Positive Rate (FPR) at various threshold levels. It provides insights into the trade-off between sensitivity (recall) and specificity as the classification threshold is varied. The ROC curve enables a visual comparison of different classifiers or model configurations, as a curve closer to the top-left corner indicates a better-performing model.

To summarize the performance of a classifier using the ROC curve, the Area Under the Curve (AUC) is computed. The AUC is a scalar value representing the entire two-dimensional area beneath the ROC curve, as in Fig. 2. It provides an aggregate measure of classifier performance across all classification thresholds, where a higher AUC indicates better discriminative ability. A perfect classifier achieves an AUC of 1, while a random classifier achieves an AUC of 0.5.

3.3.5. *Scenarios for Evaluation*

To account for varying levels of sensitivity and specificity, the models are trained and evaluated under two distinct probability threshold scenarios. These scenarios, as shown in Fig. 3, are designed to explore the impact of threshold adjustments on classification performance, particularly in the context of imbalanced datasets.

The evaluation metrics, including precision, recall, F1 score and ROC-AUC, are computed for each model across both scenarios and dataset configurations. The evaluation allows for a detailed analysis of model performance, emphasizing the influence of different preprocessing techniques and probability threshold adjustments on classification results. The technical pseudocode for AML detection system is presented in Algorithm 1.

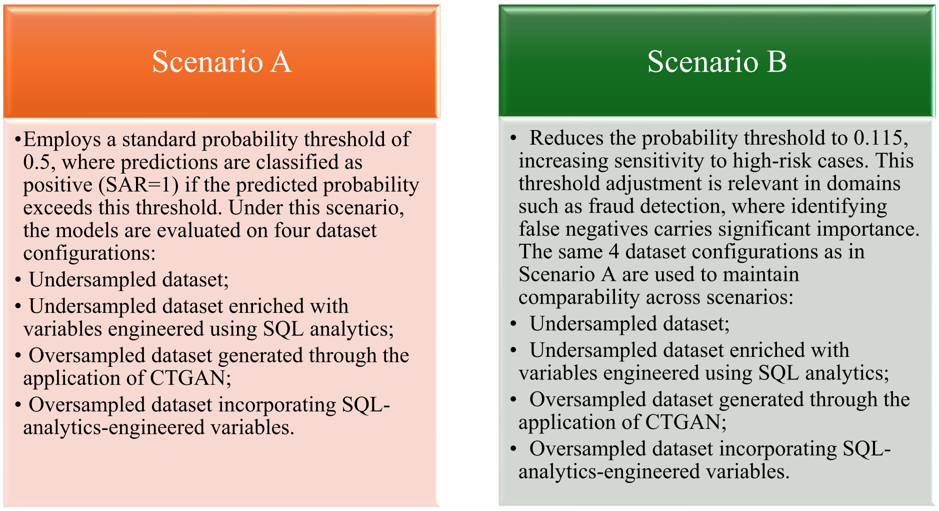
4. Results

4.1. *Exploratory Data Analysis (EDA)*

The dataset comprises 30 variables, including numerical, categorical, binary and textual data, such as csrNotes, which contains customer service notes related to the account. Table 1 contains the variables existing in the dataset along with their corresponding definitions.

Algorithm 1 Algorithm for SAR prediction

```
1. Dataset Splitting

Split dataset: 70% -> Training Set; 30% -> Testing Set

2. Data Preprocessing
2.1 Feature Encoding

For each categorical variable (cv):
     freq[cv] = count(cv) / total_count
     encoded_cv = freq[cv] // Eq. (1)
For each text variable:
     TF(t, d) = count(t in d) / total_terms(d) // Eq. (3)
     IDF(t, C) = log(|C| / (1 + count(d ∈ C : t ∈ d))) // Eq. (4)
     TF-IDF(t, d) = TF(t, d) * IDF(t, C) // Eq. (2)
For binary variables:
     if value == 'Y':
          encoded_value = 1
     else:
          encoded_value = 0

2.2 Missing Data Imputation

For variable X with missing values:
     For i = 1 to m:
          Impute missing X with plausible estimates -> Xi
     $\overline{\theta} = \frac{1}{m}\sum_{i=1}^{m}\hat{\theta_i}$ // Eq. (5)
     $Var = \overline{U} + (1 + \frac{1}{m})B$ // Eq. (6)

2.3 Feature Engineering with SQL Analytics

$CumulativeSum(i, g) = \sum_{j=1}^{i} x_j$ // Eq. (7)
$Rank(i, g) = Rank_g(-x_i)$ // Eq. (8)
$Difference(i) = x_i - x_{i-1}$ // Eq. (9)
$RollingAvg_i(g) = \frac{\sum_{j=i-k+1}^{i} x_j}{k}$ // Eq. (10)
$Rank(i) = Rank(-x_i)$ // Eq. (11)
$CumulativeAvg_r = \frac{\sum_{j=1}^{k} x_j}{r}$ // Eq. (12)
$CumulativeCount(i,\ g) = \sum_{j=1}^{i} x_j$ // Eq. (13)

2.4 Oversampling using CTGAN

Objective: $\min_G \max_D V(D, G) = \boldsymbol{E}_{x\sim p_{data}}[logD(x)] + \boldsymbol{E}_{z\sim p_z, v\sim p(v)}[log(1 - D(G(z, v)))]$ // Eq. (14)
Discriminator Loss: $\boldsymbol{\mathcal{L}}_D = -\boldsymbol{E}_{x\sim p_{data}}[\log D(x)] - \boldsymbol{E}_{z\sim p_z, v\sim p_v}[\log(1 - D(G(z, v)))]$ // Eq. (15)
Generator Loss: $\boldsymbol{\mathcal{L}}_G = -\boldsymbol{E}_{z\sim p_z, v\sim p_v}[\log D(G(z, v))]$ // Eq. (16)

3. Model Training and Evaluation
3.1 Logistic Regression (baseline)

$P(SAR = 1 \mid X) = \frac{1}{1+e^{-\beta^T X}}$ // Eq. (17)
Cost Function: $\mathcal{L} = -\frac{1}{N}\sum_{i=1}^{N}[y_i \log \hat{y}_i + (1 - y_i)\log(1 - \hat{y}_i)]$ // Eq. (18)

3.2 Decision Tree

Entropy: $H = -\sum_{i=1}^{nc} p_i \log(p_i)$ // Eq. (19)
Information Gain: $Information\ Gain = H_{parent} - (w_{left}H_{left} + w_{right}H_{right})$ // Eq. (20)

3.3 Random Forest

$P(SAR = 1) = \frac{1}{T}\sum_{t=1}^{T} P_t(SAR = 1)$ // Eq. (21)

3.4 Evaluation Metrics

Precision = TP / (TP + FP) // Eq. (22)
Recall = TP / (TP + FN) // Eq. (23)
F1 Score = 2 * (Precision * Recall) / (Precision + Recall) // Eq. (24)

3.5 ROC-AUC

Plot ROC curve:
     TPR = TP / (TP + FN)
     FPR = FP / (FP + TN)
Compute AUC: AUC = Area under ROC curve

4. Scenario Analysis. Threshold Adjustment

For threshold ∈ {0.5, 0.115}:
          Undersampled dataset;
          Undersampled dataset enriched with variables engineered using SQL analytics;
          Oversampled dataset generated through the application of CTGAN;
          Oversampled dataset incorporating SQL-analytics-engineered variables.
     Adjust model predictions
     Compute Precision, Recall, F1, ROC-AUC
     Compare metrics across thresholds
```



Scenario A

- Employs a standard probability threshold of 0.5, where predictions are classified as positive (SAR=1) if the predicted probability exceeds this threshold. Under this scenario, the models are evaluated on four dataset configurations:
- Undersampled dataset;
- Undersampled dataset enriched with variables engineered using SQL analytics;
- Oversampled dataset generated through the application of CTGAN;
- Oversampled dataset incorporating SQL-analytics-engineered variables.

Scenario B

- Reduces the probability threshold to 0.115, increasing sensitivity to high-risk cases. This threshold adjustment is relevant in domains such as fraud detection, where identifying false negatives carries significant importance. The same 4 dataset configurations as in Scenario A are used to maintain comparability across scenarios:
- Undersampled dataset;
- Undersampled dataset enriched with variables engineered using SQL analytics;
- Oversampled dataset generated through the application of CTGAN;
- Oversampled dataset incorporating SQL-analytics-engineered variables.

Fig. 3. Scenarios applied in the analysis.

Most variables have complete data with no missing values. However, there are two notable exceptions, represented by income, which has 200 missing values, representing 2% of the dataset, suggesting the need for imputation to handle these gaps effectively and totalPaymentAmt90d which has 55 missing values, accounting for 0.55% of the dataset, also requiring imputation to ensure model compatibility. For all other variables, no missing data is observed, allowing them to be directly used in the analysis without additional preprocessing steps. It is also important to note that the variable csrNotes is transformed using the TF-IDF method.

Figure 4 presents histograms illustrating the distribution of three numerical variables: kycRiskScore, income and tenureMonths. The first histogram, representing kycRiskScore, reveals a concentration of values within the lower range, predominantly between 0 and 3. This suggests that most customers are assigned low risk scores, with relatively few cases exceeding a score of 4. This distribution is indicative of the dataset's emphasis on lower-risk customers, aligning with the expected structure of a typical banking dataset. The second histogram, depicting income, shows a highly right-skewed distribution. The majority of income values are clustered below 100,000, with a gradual tapering off as income increases. It is indicative that while most customers fall within a lower to moderate income range, there are a few higher-income customers present as outliers. The distribution underscores the importance of scaling or normalization techniques to ensure this feature does not disproportionately influence model performance.

The third histogram, for tenureMonths, highlights a distribution concentrated at lower tenure values, with most customers having been active for fewer than 50 months. The frequency decreases sharply as tenure increases, with only a small fraction of customers exhibiting long-term activity. This pattern may reflect a relatively young or dynamic customer base, and it also suggests that customer tenure may have a significant relationship with other variables, such as risk score or income.

Table 1
Variables and their corresponding descriptions.

Variable name	Description
SAR	Indicates whether a SAR was filed. 1: SAR filed, 0: no SAR.
kycRiskScore	KYC risk score at account opening. Higher score indicates higher risk.
Income	Annual income of the account holder, typically in local currency.
tenureMonths	Length of time (in months) the account has been active.
creditScore	Credit bureau score of the account holder. Higher score indicates better creditworthiness.
State	State of the account holder's billing address.
nbrPurchases90d	Number of purchases made in the last 90 days.
avgTxnSize90d	Average size of transactions (monetary value) in the last 90 days.
totalSpend90d	Total amount of money spent in the last 90 days.
csrNotes	Notes recorded by Customer Service Representatives during interactions with the account holder.
nbrDistinctMerch90d	Number of distinct merchants where purchases were made in the last 90 days.
nbrMerchCredits90d	Number of credits received from merchants in the last 90 days.
nbrMerchCredits-RndDollarAmt90d	Number of credits received from merchants in round dollar amounts in the last 90 days.
totalMerchCred90d	Total amount of merchant credits received in the last 90 days.
nbrMerchCredits-WoOffsettingPurch	Number of merchant credits issued without offsetting purchases in the last 90 days.
nbrPayments90d	Number of payments made in the last 90 days.
totalPaymentAmt90d	Total payment amount made in the last 90 days.
overpaymentAmt90d	Total amount overpaid by the account holder in the last 90 days.
overpaymentInd90d	Indicates whether the account was overpaid in the last 90 days. 1: overpaid, 0: no overpayment.
nbrCustReqRefunds90d	Number of refund requests made in the last 90 days.
indCustReqRefund90d	Indicates whether a refund was requested in the last 90 days. 1: refund requested, 0: no refund request.
totalRefundsToCust90d	Total refund amount issued in the last 90 days.
nbrPaymentsCashLike90d	Number of cash-like payments (e.g. money orders) made in the last 90 days.
maxRevolveLine	Maximum revolving line of credit available to the account holder.
indOwnsHome	Indicates whether the account holder owns a home. 1: owns home, 0: does not own home.
nbrInquiries1y	Number of credit inquiries made about the account holder in the last year.
nbrCollections3y	Number of collections associated with the account holder in the last three years.
nbrWebLogins90d	Number of online banking logins in the last 90 days.
nbrPointRed90d	Number of loyalty points redemptions made in the last 90 days.
PEP	Indicates whether the account holder is a Politically Exposed Person (PEP). 1: PEP, 0: not PEP.

Figure 5 illustrates the frequency distribution of the state variable within the dataset. Each bar represents the number of observations associated with a specific state. The highest frequency is observed for the state of NY (New York), with a count of 2.424 records, followed closely by MA (Massachusetts), which has 2.328 records. Other states such as PA (Pennsylvania) and CT (Connecticut) also show relatively high frequencies, with counts of 1.242 and 1.108, respectively. Conversely, NH (New Hampshire) and VT (Vermont) exhibit the lowest frequencies, with only 95 and 101 observations, respectively.

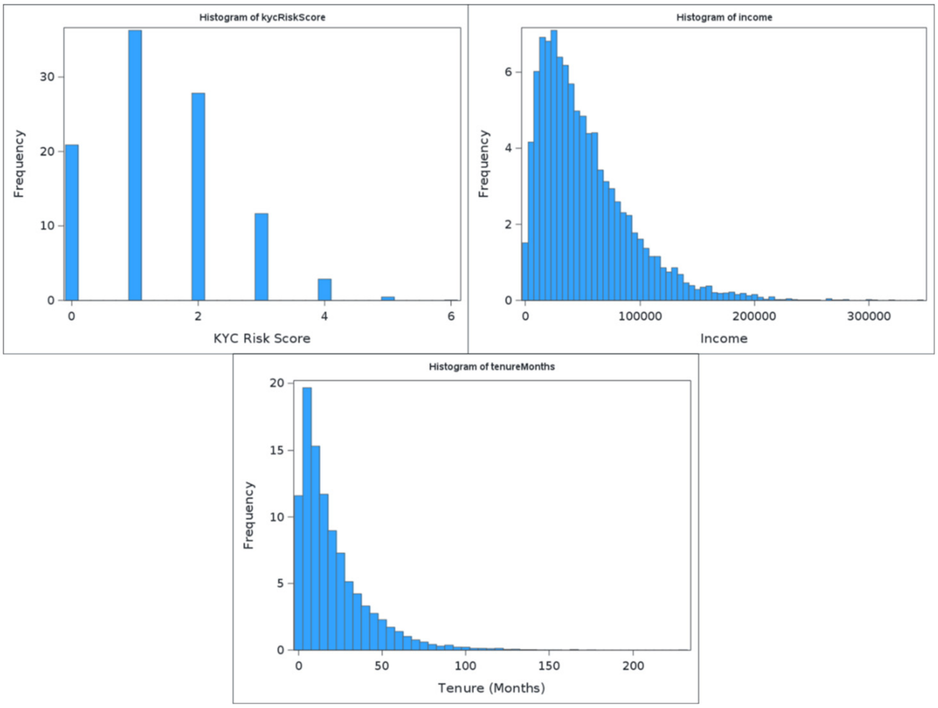


Fig. 4. Histograms for kycRiskScore, Income and tenureMonths.

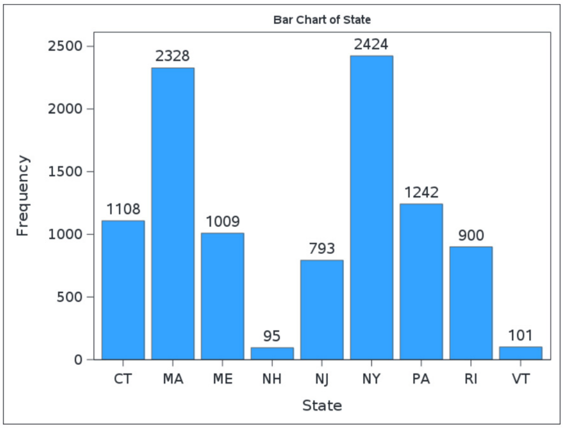


Fig. 5. Frequency distribution of the SAR variable.

Figure 6 illustrates the distribution of the SAR variable, which serves as the dependent variable in this study. The SAR variable distinguishes between instances where a SAR was filed (Y) and those where no such report was filed (N). The chart reveals a pronounced class imbalance, with the majority of observations classified as N (8.974 instances) compared to only 1.026 instances classified as Y.

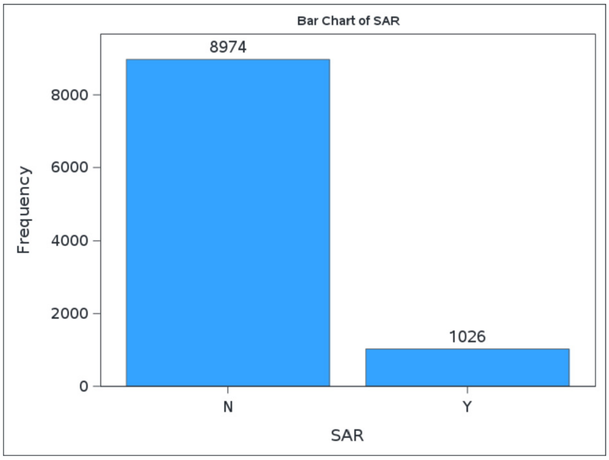


Fig. 6. Distribution of the SAR variable.

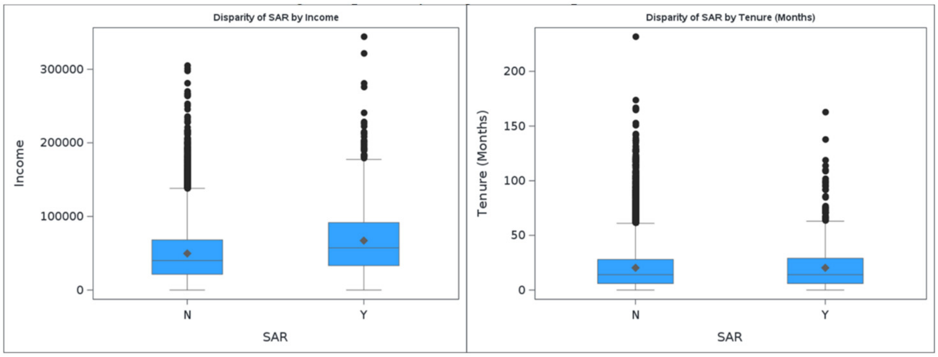


Fig. 7. Disparity of the SAR variable with respect to income and tenure.

To address this imbalance, we employ the CTGAN to generate synthetic data that closely mimics the distribution of the minority class. This oversampling method enriches the dataset with additional instances of SAR $= Y$, ensuring a more balanced representation of the classes during the model training process. By incorporating CTGAN-generated data, we aim to improve the model's ability to identify suspicious transactions without overfitting to the limited real instances of the minority class.

Figure 7 depicts the disparity of the SAR variable with respect to income (left) and tenure (right). In the first boxplot, for both categories of the SAR, the median income is fairly similar. However, there is a noticeable difference in the distribution and range of income values. Category N exhibits a slightly broader interquartile range (IQR), indicating a higher variability in income among customers without a SAR. The presence of numerous outliers, particularly in the higher-income range for both categories, highlights that certain customers have significantly higher income values, which may influence the

Table 2
Engineered variables using SQL Oracle Analytics.

Engineered variable	Description
Cumulative spend by state	Computes the cumulative total spend for accounts within each state, ordered by income.
Rank by number of purchases in state	Assigns a rank to accounts within each state based on the number of purchases in descending order.
Difference in number of purchases	Calculates the difference in the number of purchases between consecutive rows, ordered by income.
Rolling average of transaction size by state	Calculates the rolling average of transaction sizes within each state, ordered by income, using a fixed window size.
Rank by income	Assigns a global rank to accounts based on income in descending order.
Cumulative average transaction size	Computes the cumulative average transaction size across all accounts, ordered by transaction size.
Difference in total spend	Calculates the difference in total spend between consecutive rows, ordered by income.
Cumulative purchases by income	Computes the cumulative number of purchases ordered by income across all accounts.
Rank by credit score within state	Assigns a rank to accounts within each state based on credit score in descending order.
Cumulative income by state	Computes the cumulative income within each state, ordered by income.

models' predictions. In the second boxplot the median tenure for both N and Y categories is again comparable, but the interquartile range for the N category is slightly wider, indicating greater variability in customer tenure among those without an SAR. Similar to income, outliers are present in the higher ranges of tenure, with a small number of customers having exceptionally long relationships with the institution.

4.2. *Engineered Features using SQL Analytics*

To enrich the dataset and capture more nuanced patterns in customer behaviour, 10 engineered variables were created using SQL Oracle Analytics. These variables provide insights into cumulative trends, rankings and differences across various dimensions, enabling enhanced feature representation for ML models. Table 2 summarizes the names and descriptions of the engineered variables.

4.3. *LR, DT and RF Architectures*

a) LR model

The LR is implemented using the PROC LOGISTIC procedure in SAS, which is specifically designed to handle binary classification problems. The model is configured to predict the likelihood of a SAR being filed, focusing on the event SAR = 1. To ensure this focus, the DESCENDING option is used, which models the higher-ordered category (SAR = 1) instead of the default lower-ordered category (SAR = 0). The relationship between the target variable SAR and the independent variables is defined in the MODEL statement, which includes all predictors in the regression equation.

By default, the LR model in SAS uses the logit link function, which models the log-odds of the event occurring as a linear combination of the predictor variables. The model employs Fisher's scoring method, an iterative optimization technique for estimating the maximum likelihood of the parameters. The convergence criterion is set by default to a gradient convergence (GCONV) value of 1E-8, ensuring that the iterative estimation process terminates only when the parameter estimates stabilize. Additionally, the default significance level for including terms in the model (*alpha*) is 0.05, controlling the threshold for statistical significance in variable selection when stepwise or other selection methods are applied.

b) *DT model*

The DT model is implemented using the PROC HPSPLIT procedure in SAS, a high-performance tool for building classification and regression trees. The MODEL statement specifies the target variable (SAR) and the independent variables used for prediction. The event = '1' option ensures that the model focuses on identifying instances where SAR = 1. This configuration aligns with the study's objective of accurately identifying suspicious activities.

In constructing the tree, the grow entropy option is used to guide the splitting process. Entropy is a measure of impurity, and the algorithm selects splits that maximize information gain, effectively reducing impurity and improving classification accuracy at each node. To prevent overfitting, the tree is pruned using the prune cost-complexity option. Cost-complexity pruning evaluates the trade-off between the complexity of the tree (number of splits) and its predictive accuracy on validation data. By minimizing this trade-off, the model achieves a balance between overfitting and underfitting, ensuring better generalization to unseen data. The seed = 123 option ensures reproducibility by setting a fixed random seed for any stochastic processes involved during training.

c) *RF model*

The RF model in our research is implemented using the PROC HPFOREST procedure in SAS, a high-performance tool designed for building ensemble-based DT models. RF operates by constructing an ensemble of DTs, each trained on a random subset of the data and features and then aggregating their predictions to improve accuracy and robustness. The target variable in this model is SAR, with the level = binary option indicating that the model is designed for binary classification tasks, specifically predicting whether a SAR is filed (SAR = 1) or not (SAR = 0).

The *maxtrees* = 500 parameter configures the model to construct a maximum of 500 trees, ensuring a sufficiently large ensemble to capture the complexity of the data. The *maxdepth* = 10 parameter limits the depth of each tree to 10 levels, preventing the trees from becoming overly complex and mitigating the risk of overfitting. The *leafsize* = 15 parameter specifies the minimum number of observations required in a terminal node, further controlling the granularity of the splits and enhancing the model's generalizability.

The seed = 123 option also ensures reproducibility. During training, the RF builds each tree using a bootstrap sample of the data, and a random subset of features is considered for splitting at each node. The randomness introduces diversity among the trees, which improves the model's performance when the predictions are aggregated.

Table 3
Performance metrics for the undersampled datasets, scenario A.

Model/Metric	*Precision*	*Recall*	*F1*
LR	0.463	0.215	0.293
DT	0.514	0.346	0.413
RF	0.631	0.374	0.470

Table 4
Performance metrics for the undersampled dataset containing engineered variables, scenario A.

Model/Metric	*Precision*	*Recall*	*F1*
LR	0.644	0.205	0.311
DT	0.623	0.388	0.478
RF	0.702	0.367	0.482

4.4. *Performance Metrics*

In our study, the performance of three ML models (LR, DT and RF), is evaluated under two distinct scenarios with varying probability thresholds (0.5 and 0.115). The analysis is conducted on four dataset configurations for each scenario: an undersampled dataset, an undersampled dataset with SQL-analytics-engineered variables, an oversampled dataset and an oversampled dataset with SQL-analytics-engineered variables.

4.4.1. *Scenario A, threshold = 0.5*

a) Undersampled dataset

Table 3 presents the performance metrics for the undersampled dataset using a 0.5 probability threshold in Scenario A. Precision, Recall and F1-scores are reported for the three ML models.

The RF model demonstrates the highest overall performance across all metrics, achieving a precision of 0.631, a recall of 0.374 and an F1-score of 0.470. The DT model follows with a moderate recall (0.346) and an F1-score of 0.413, while LR exhibits the lowest performance, with a recall of 0.215 and an F1-score of 0.293.

b) Undersampled dataset analytics-engineered variables

After integrating SQL-engineered features, there is a clear improvement across the performance metrics, especially noticeable in precision and the F1-score (Table 4). The RF model remains the top performer, showing improved precision (0.702) and F1-score (0.482), despite maintaining a moderate recall (0.367). The DT model also benefits substantially from the engineered variables, exhibiting an improved precision of 0.623, higher recall of 0.388, and an F1-score of 0.478. LR sees a noteworthy improvement in precision (0.644), yet its recall remains low (0.205), leading to a modest increase in the F1-score (0.311).

c) Oversampled dataset

In the oversampled scenario, all models display noticeable improvements in recall and F1-scores compared to the undersampled setting, highlighting the effectiveness of

Table 5
Performance metrics for the oversampled dataset, scenario A.

Model/Metric	*Precision*	*Recall*	*F1*
LR	0.641	0.229	0.337
DT	0.755	0.321	0.450
RF	0.696	0.385	0.495

Table 6
Performance metrics for the oversampled dataset with engineered features, scenario A.

Model/Metric	*Precision*	*Recall*	*F1*
LR	0.623	0.217	0.321
DT	0.495	0.440	0.465
RF	0.678	0.373	0.481

CTGAN-generated synthetic samples in mitigating class imbalance. RF remains the highest-performing model, reaching the best balance of precision (0.696), recall (0.385), and F1-score (0.495). DT notably improves in precision (0.755) and recall (0.321), leading to an enhanced F1-score of 0.450. LR achieves modest improvements, maintaining good precision (0.641) but limited recall (0.229), resulting in an F1-score of 0.337 (see Table 5).

d) Oversampled dataset with analytics-engineered variables

In this oversampled scenario with engineered features, the RF model achieves the best overall balance, showing precision of 0.678, recall of 0.373, and the highest F1-score of 0.481. DT experiences a substantial increase in recall (0.440), though its precision decreases to 0.495, resulting in an F1-score of 0.465. LR maintains moderate precision (0.623), but its limited recall (0.217) leads to a lower F1-score (0.321) (see Table 6).

4.4.2. *Scenario B, threshold = 0.115*

a) Undersampled dataset

In Scenario B, with a threshold of 0.115, recall metrics increase significantly across all models, at the expense of precision. RF achieves the highest recall (0.996) and F1-score (0.517). LR shows improved recall (0.830), but its F1-score remains relatively low at 0.455. The DT model strikes a balance, with an F1-score of 0.522. The lower threshold improves the model's sensitivity to identifying SAR events. The results are presented in Table 7.

b) Undersampled dataset analytics-engineered variables

The integration of SQL-analytics-engineered variables further enhances performance metrics under Scenario B (threshold = 0.115), as shown in Table 8. RF achieves the highest recall (0.994) and the best overall F1-score (0.573), indicating strong sensitivity to SAR events, while maintaining a moderate precision of 0.403. DT also benefits from the engineered features, reaching a precision of 0.431, a recall of 0.838, and an F1-score of 0.569. LR shows consistent improvement, achieving a precision of 0.403, a recall of 0.835, and an

Table 7
Performance metrics for the undersampled datasets, scenario B.

Model/Metric	*Precision*	*Recall*	*F1*
LR	0.314	0.830	0.455
DT	0.362	0.937	0.522
RF	0.349	0.996	0.517

Table 8
Performance metrics for the undersampled dataset containing engineered variables, scenario B.

Model/Metric	*Precision*	*Recall*	*F1*
LR	0.403	0.835	0.543
DT	0.431	0.838	0.569
RF	0.403	0.994	0.573

Table 9
Performance metrics for the oversampled dataset, scenario B.

Model/Metric	*Precision*	*Recall*	*F1*
LR	0.546	0.434	0.483
DT	0.442	0.829	0.576
RF	0.477	1.000	0.645

F1-score of 0.543. Overall, the addition of engineered variables significantly strengthens model performance, particularly by enhancing the balance between precision and recall across all classifiers, with the strongest gains observed for RF and DT.

c) Oversampled dataset

Table 9 presents the performance metrics for the oversampled dataset under Scenario B (threshold $= 0.115$). In this configuration, RF achieves the highest recall (1.000) and the best F1-score (0.645), although its precision remains moderate (0.477). DT shows substantial improvement compared to the undersampled setting, achieving a precision of 0.442, a recall of 0.829, and an F1-score of 0.576. LR maintains a reasonable balance, with a precision of 0.546, a lower recall of 0.434, and an F1-score of 0.483.

d) Oversampled dataset with analytics-engineered variables

Table 10 presents the performance metrics for the oversampled dataset with SQL-analytics-engineered variables under Scenario B (threshold $= 0.115$). In this final configuration, RF achieves the highest recall (0.991) and an F1-score of 0.528, maintaining strong sensitivity to SAR events. DT shows a solid balance with a precision of 0.375, a recall of 0.847, and an F1-score of 0.519, benefiting substantially from the engineered features. LR achieves a precision of 0.532, a recall of 0.427, and an F1-score of 0.473, showing moderate but consistent performance.

Table 10
Performance metrics for the oversampled dataset with engineered features, scenario B.

Model/Metric	*Precision*	*Recall*	*F1*
LR	0.532	0.427	0.473
DT	0.375	0.847	0.519
RF	0.360	0.991	0.528

Table 11
ROC-AUC score across each dataset.

Model/Metric	*Undersampled*	*Undersampled engineered*	*Oversampled*	*Oversampled engineered*
LR	0.906	0.915	0.915	0.901
DT	0.913	0.920	0.929	0.904
RF	0.945	0.946	0.951	0.935

4.4.3. *The ROC-AUC score for each dataset*

Table 11 presents the ROC-AUC scores for the three ML models, LR, DT, and RF, across four dataset configurations: undersampled, undersampled with analytics-engineered variables, oversampled, and oversampled with analytics-engineered variables. Since ROC-AUC is threshold-independent, these scores provide a robust overall measure of the models' ability to distinguish between SAR and non-SAR events.

For the undersampled dataset, LR, DT, and RF achieve ROC-AUC scores of 0.906, 0.913, and 0.945, respectively. RF demonstrates the strongest discriminatory power even under imbalanced conditions, outperforming LR and DT.

The inclusion of SQL-analytics-engineered variables leads to slight improvements in ROC-AUC for all models on the undersampled dataset. LR improves from 0.906 to 0.915, DT from 0.913 to 0.920, and RF from 0.945 to 0.946, with RF consistently maintaining the highest score.

When oversampling is applied, ROC-AUC scores continue to remain strong across models. LR and DT both achieve a ROC-AUC of 0.915 and 0.929, respectively, while RF further improves to 0.951. This indicates that oversampling successfully enhances model sensitivity to the minority class without sacrificing the overall ability to differentiate classes.

In the final configuration, combining oversampling with analytics-engineered variables slightly lowers ROC-AUC scores compared to oversampling alone. LR drops slightly to 0.901, DT to 0.904, and RF to 0.935. Nevertheless, RF continues to demonstrate the most robust and stable performance across all configurations, confirming its ability to generalize well even when complex preprocessing techniques are applied.

5. Discussion

The results from our analysis highlight the complexity of detecting SAR in the financial domain, where imbalanced datasets and diverse feature sets pose significant challenges.

By leveraging specific ML models such as LR, DT, and RF, we systematically explored the impact of preprocessing techniques, dataset configurations, and probability thresholds on model performance.

The EDA phase revealed valuable insights into the structure and characteristics of the dataset. Numerical variables such as kycRiskScore, income, and tenureMonths demonstrated distinct distributions, shedding light on the heterogeneity of the customer base. For instance, the right-skewed distribution of income underscores the presence of outliers among high-income customers, potentially influencing model predictions. Similarly, categorical variables like state illustrated geographical disparities in data distribution, while the imbalance in the target variable SAR reinforced the need for effective oversampling strategies to address the minority class. The transformation of textual data (csrNotes) using the TF-IDF method further ensured that key information from unstructured data was incorporated into the predictive framework.

The architectures of the three models were configured to align with the study's objectives. LR was implemented with a focus on the higher-order category (SAR = 1), leveraging the logit link function and robust optimization techniques. The DT model utilized entropy-based splits to maximize information gain, coupled with cost-complexity pruning to balance model complexity and generalization. The RF model, with its ensemble approach, demonstrated superior robustness and performance, benefiting from hyperparameters such as a maximum of 500 trees, depth limitations, and a minimum leaf size, reflecting an emphasis on scalability, interpretability, and predictive accuracy across the diverse dataset configurations.

Performance metrics across scenarios and dataset configurations provided a comprehensive evaluation of the models. Under Scenario A (threshold = 0.5), RF achieved the highest precision (0.631), recall (0.374), and F1-score (0.470) on the undersampled dataset, improving further when SQL-analytics-engineered features were introduced. The addition of engineered variables led to noticeable gains in F1-scores for both DT and RF, enhancing model discrimination capabilities. In the oversampled setting, CTGAN-based synthetic data improved recall across all models, with RF attaining a recall of 0.385 and an F1-score of 0.495.

In Scenario B (threshold = 0.115), the lower threshold shifted the focus toward recall, allowing the models to capture a greater proportion of SAR events. While this came at the expense of precision, RF achieved near-perfect recall (0.996) on the undersampled dataset and maintained strong F1-scores when feature engineering was applied (0.573). In the oversampled configuration, RF maintained perfect recall (1.000) but precision dropped moderately, resulting in an F1-score of 0.645. These trends demonstrate the clear trade-off between sensitivity and specificity at different operating thresholds.

The ROC-AUC scores provided a threshold-invariant measure of model performance, with RF consistently achieving the highest scores across all dataset configurations. ROC-AUC values remained stable around 0.945 to 0.951 for RF across undersampled and oversampled datasets, confirming the model's generalization capability even after synthetic oversampling and feature engineering. This underscores the robustness of ensemble methods in handling imbalanced datasets and capturing complex relationships between features.

The combination of oversampling using CTGAN and analytics-engineered variables yielded significant improvements, although a slight decrease in ROC-AUC was observed when both were combined (RF achieving 0.935 compared to 0.951 with only oversampling). Nevertheless, this small drop is acceptable given the substantial recall gains and the operational need to detect as many SAR events as possible.

From a practical perspective, the findings have important implications for AML efforts. The CTGAN-based oversampling approach effectively addressed the class imbalance, enabling the models to better identify minority-class instances without overfitting. The analytics-engineered features, derived through SQL-based transformations, enriched the dataset with domain-specific insights, enhancing model interpretability and predictive power. The preprocessing techniques proved critical in improving model sensitivity and specificity, particularly in challenging scenarios with limited SAR instances.

Moreover, the trade-offs between precision and recall are highlighted, which are central to AML applications. While a high recall ensures that potential SAR events are not missed, precision remains crucial to minimizing false positives and maintaining operational efficiency. By applying a calibrated probability threshold (0.115) and structuring the preprocessing pipeline to prevent data leakage, the models achieved a practical balance between these competing objectives. This strategy aligns with the regulatory and operational priorities of financial institutions, offering a realistic path for deploying ML-enhanced AML systems.

6. Conclusions

Our research provides an in-depth examination of detecting SAR in the financial domain by employing various ML models, LR, DT, and RF. Through a combination of feature engineering, data balancing with CTGAN, and systematic evaluation across multiple scenarios, we highlight the complexities of handling imbalanced datasets and the importance of tailored model architectures and preprocessing techniques.

The results reveal that the RF model consistently outperforms LR and DT in predictive performance, achieving the highest ROC-AUC scores across all dataset configurations, with values ranging from 0.945 to 0.951. RF demonstrates superior robustness and generalization even after applying oversampling and feature engineering techniques. The DT model performs competitively, benefiting from entropy-based splitting and cost-complexity pruning, which ensure a balance between accuracy and generalization. Meanwhile, the LR model, despite its simplicity, shows improvement when analytics-engineered features are incorporated, particularly in scenarios involving undersampled and oversampled datasets.

The inclusion of CTGAN for oversampling proves critical in addressing the severe class imbalance present in SAR detection datasets. By generating synthetic data that closely mimics the minority class distribution, CTGAN enhances recall across all models without causing significant overfitting, as evidenced by the stable ROC-AUC scores even after oversampling. In the oversampled scenarios, RF reaches a recall of 1.000 and an F1-score of 0.645, maintaining its overall superior performance due to its ensemble nature.

In short, performance metrics evaluated across two probability thresholds (0.5 and 0.115) illustrate the trade-offs between precision and recall, which are central to AML applications. In Scenario A (threshold = 0.5), precision is emphasized, with RF achieving the best balance between precision (0.631) and recall (0.374) on the undersampled dataset. Conversely, in Scenario B (threshold = 0.115), recall becomes the priority, with RF achieving near-perfect recall (0.996) under undersampling and 1.000 under oversampling. Through targeted feature engineering, CTGAN-based data balancing, and systematic evaluation, we demonstrate the challenges and effective strategies for handling highly imbalanced datasets. A key finding is that the inclusion of SQL-analytics-engineered features substantially enhanced model performance across all configurations. This trade-off is necessary in AML contexts, where failing to identify suspicious transactions (false negatives) could have severe regulatory and operational consequences.

Despite its strengths, the research has several limitations. The use of static datasets limits the ability to capture dynamic, real-time changes in customer behaviour, which are necessary in AML applications. The dynamism of real-world financial transactions is not captured. While CTGAN effectively addresses class imbalance, the introduction of synthetic data may introduce noise, potentially affecting model generalization.

Future research could build on these findings by incorporating real-time data streams and investigating the impact of additional behavioural or external economic variables. More advanced calibration techniques and hyperparameter optimization could further enhance the models' performance, while comparative studies of alternative data balancing techniques, such as ADASYN or other generative approaches, could provide deeper insights into their relative effectiveness. Additionally, integrating interpretability techniques, such as SHAP, into the workflow could enhance trust and regulatory compliance by providing actionable insights into model predictions.

Authors Contributions. All authors contributed to the study conception and design. Material preparation, data collection and analysis were performed by Anca Andreescu and Adela Bâra. The first draft of the manuscript was written by Simona-Vasilica Oprea and Alin Gabriel Văduva commented on previous versions of the manuscript. All authors read and approved the final manuscript.

Disclosure statement. Competing Interests. The authors report there are no competing interests to declare.

Data availability statement. The data will be made available upon request.

A. Appendix

Table A1
Comparison of previous research papers.

Reference	Objective	Method	Results
(Al Badawi and Al-Haija, 2021)	To combat cryptocurrency-based money laundering using ML	Used shallow NN and decision trees on the Elliptic dataset	Shallow NN: 89.9% accuracy; Decision Tree: 93.4% accuracy
(Caglayan and Bahtiyar, 2022)	Improving AML detection with ML algorithms	Used Node2Vec for graph-based data representation and classification	Achieved better classification results than existing methods
(Liu *et al.*, 2023)	Detecting Ethereum-based money laundering	Proposed GTN2vec algorithm for money laundering detection using graph embedding	Achieved higher accuracy than advanced graph embedding methods
(Labanca *et al.*, 2022)	Reducing false positives in AML detection	Proposed Amaretto, an active learning framework, combining supervised and unsupervised techniques	Improved detection rate by 50%, reduced costs by 20%
(Huong *et al.*, 2024)	Enhancing money laundering detection through graph-based transactions	Constructed network graphs from bank transactions and applied random forest for prediction	Oversampling accuracy: 86%; Undersampling accuracy: 92%
(Zhang and Trubey, 2019)	Examining ML and sampling techniques for money laundering detection	Studied 5 ML algorithms using transaction data from a US financial institution	Highlighted advantages of ML in detecting rare events
(Hampo *et al.*, 2023)	Developing a web-based AML detection system using k-NN	Implemented k-NN algorithm with open Kaggle datasets	Achieved 98.4% accuracy
(Sheu and Li, 2022)	Addressing failure in ML detection for Panama Papers data	Developed a graph attention network with self-attention mechanism	Outperformed Naïve Bayes and SVM in detecting money laundering accounts
(Zhong *et al.*, 2022)	Detecting money laundering in cryptocurrency transactions	Designed a 4-stage AML detection system using outlier detection and cluster detection	Achieved 96.02% accuracy for abnormal transaction detection
(Drezewski *et al.*, 2015)	Supporting human analytics in AML through social network analysis	Proposed MLDS using social network analysis	Provided tools for visualizing and analysing AML networks
(Oad *et al.*, 2021)	Proposing blockchain-enabled transaction scanning (BTS) for AML	Developed a BTS method for anomaly detection and applied blockchain for transaction monitoring	Demonstrated automation in detecting suspicious transactions
(Luo *et al.*, 2022)	Developing NN for dynamic transaction pattern detection	Proposed DTPAN for learning dynamic features of transaction behaviours	Enhanced performance for AML detection compared to previous methods
(Alotibi *et al.*, 2022)	Investigating AML detection in cryptocurrency using ML techniques	Applied deep learning and ML (DNN, RF, KNN) to detect suspicious transactions in Bitcoin	Random Forest achieved an F1-score of 0.99%
(Bidabad, 2017)	Proposing a new system for AML detection using tax and banking data	Developed a MLDS comparing tax data with banking transactions	Detected and traced underground economic activities
(Lo *et al.*, 2023)	Detecting illicit cryptocurrency transactions using GNN	Proposed Inspection-L, a self-supervised GNN framework with deep learning for detecting illicit Bitcoin transactions	Outperformed state-of-the-art methods in classification
(Ketenci *et al.*, 2021)	Improving AML systems using time-frequency analysis	Applied Random Forest with time-frequency analysis for detecting suspicious transactions	Reduced false positives to 11.85%, improved F-Score to 74.06%

Acknowledgements

This work was supported by a grant of the Ministry of Research, Innovation and Digitization, CNCS/CCCDI – UEFISCDI, project number COFUND-CETP-SMART-LEM-1, within PNCDI IV.

References

Ahmad Tarmizi, M., Zolkaflil, S., Omar, N., Hasnan, S., Syed Mustapha Nazri, S.N.F. (2023). Compliance determinants of anti-money laundering regime among professional accountants in Malaysia. *Journal of Money Laundering Control*, 26(2), 361–387. https://doi.org/10.1108/JMLC-01-2022-0003.

Alotibi, J., Almutanni, B., Alsubait, T., Alhakami, H., Baz, A. (2022). Money laundering detection using machine learning and deep learning. *International Journal of Advanced Computer Science and Applications*, 13(10). https://doi.org/10.14569/IJACSA.2022.0131087.

Antwi, S., Tetteh, A.B., Armah, P., Dankwah, E.O. (2023). Anti-money laundering measures and financial sector development: empirical evidence from Africa. *Cogent Economics and Finance*, 11(1). https://doi.org/10.1080/23322039.2023.2209957.

Al Badawi, A., Al-Haija, Q.A. (2021). Detection of money laundering in bitcoin transactions. In: *IET Conference Proceedings*. https://doi.org/10.1049/icp.2022.0387.

Benzerrouk, Z.S., Alnor, N.H.A., Al-Matari, E.M., Alhebri, A., Al-Bukhrani, M.A. (2023). The effect of the banking supervision on anti-money laundering. *Humanities and Social Sciences Letters*, 11(4), 399–415. https://doi.org/10.18488/73.v11i4.3518.

Bidabad, B. (2017). Money laundering detection system (MLD) (a complementary system of rastin banking). *Journal of Money Laundering Control*, 20(4), 354–366. https://doi.org/10.1108/JMLC-04-2016-0016.

Caglayan, M., Bahtiyar, S. (2022). Money laundering detection with Node2Vec. *Gazi University Journal of Science*, 35(3), 854–873. https://doi.org/10.35378/gujs.854725.

Chen, Z., Van Khoa, L.D., Teoh, E.N., Nazir, A., Karuppiah, E.K., Lam, K.S. (2018). Machine learning techniques for anti-money laundering (AML) solutions in suspicious transaction detection: a review. *Knowledge and Information Systems*, 57, 245–285. https://doi.org/10.1007/s10115-017-1144-z.

Chen, Z., Soliman, W.M., Nazir, A., Shorfuzzaman, M. (2021). Variational autoencoders and wasserstein generative adversarial networks for improving the anti-money laundering process. *IEEE Access*, 9, 83762–83785. https://doi.org/10.1109/ACCESS.2021.3086359.

Cheng, D., Ye, Y., Xiang, S., Ma, Z., Zhang, Y., Jiang, C. (2023). Anti-money laundering by group-aware deep graph learning. *IEEE Transactions on Knowledge and Data Engineering*, 35(12), 12444–12457. https://doi.org/10.1109/TKDE.2023.3272396.

Chitimira, H., Animashaun, O. (2023). The adequacy of the legal framework for combating money laundering and terrorist financing in Nigeria. *Journal of Money Laundering Control*, 26(7), 110–126. https://doi.org/10.1108/JMLC-12-2022-0171.

Demetis, D.S. (2018). Fighting money laundering with technology: a case study of Bank X in the UK. *Decision Support Systems*, 105, 96–107. https://doi.org/10.1016/j.dss.2017.11.005.

Drezewski, R., Sepielak, J., Filipkowski, W. (2015). The application of social network analysis algorithms in a system supporting money laundering detection. *Information Sciences*, 295, 18–32. https://doi.org/10.1016/j.ins.2014.10.015.

Gilmour, P.M. (2023). Reexamining the anti-money-laundering framework: a legal critique and new approach to combating money laundering. *Journal of Financial Crime*, 30(1), 35–47. https://doi.org/10.1108/JFC-02-2022-0041.

Goecks, L.S., Korzenowski, A.L., Terra Neto, P.G., de Souza, D.L., Mareth, T. (2022). Anti-money laundering and financial fraud detection: a systematic literature review. *Intelligent Systems in Accounting, Finance and Management*, 29(2), 71–85. https://doi.org/10.1002/isaf.1509.

Hampo, J.P.A.C., Nwokorie, E.C., Odii, J.N. (2023). A web-based KNN money laundering detection system. *European Journal of Theoretical and Applied Sciences*, 1(4), 277–288. https://doi.org/10.59324/ejtas.2023.1(4).27.

Huong, H., Nguyen, X., Dang, T.K., Tran-Truong, P.T. (2024). Money laundering detection using a transaction-based graph learning approach. In: *Proceedings of the 2024 18th International Conference on Ubiquitous Information Management and Communication, IMCOM 2024*, pp. 1–8. https://doi.org/10.1109/IMCOM60618.2024.10418307.

Isolauri, E.A., Ameer, I. (2023). Money laundering as a transnational business phenomenon: a systematic review and future agenda. *Critical Perspectives on International Business*, 19(3), 426–468. https://doi.org/10.1108/cpoib-10-2021-0088.

Jensen, R.I.T., Iosifidis, A. (2023). Fighting money laundering with statistics and machine learning. *IEEE Access*, 11, 8889–8903. https://doi.org/10.1109/ACCESS.2023.3239549.

Jovicic, S., Tan, Q. (2018). Machine learning for money laundering detection in the block chain financial transaction system. *Journal of Fundamental and Applied Sciences*, 10(4S).

Kannan, S., Somasundaram, K. (2017). Autoregressive-based outlier algorithm to detect money laundering activities. *Journal of Money Laundering Control*, 20(2), 190–202. https://doi.org/10.1108/JMLC-07-2016-0031.

Ketenci, U.G., Kurt, T., Önal, S., Erbil, C., Aktürkoğlu, S., Ilhan, H.Ş. (2021). A time-frequency based suspicious activity detection for anti-money laundering. *IEEE Access*, 9, 59957–59967. https://doi.org/10.1109/ACCESS.2021.3072114.

Korejo, M.S., Rajamanickam, R., Muhamad, M.H. (2021). The concept of money laundering: a quest for legal definition. *Journal of Money Laundering Control*, 24(4), 725–736. https://doi.org/10.1108/JMLC-05-2020-0045.

Kramer, J.A., Blokland, A.A.J., Kleemans, E.R., Soudijn, M.R.J. (2023). Money laundering as a service: investigating business-like behavior in money laundering networks in the Netherlands. *Trends in Organized Crime*, 27, 314–341. https://doi.org/10.1007/s12117-022-09475-w.

Labanca, D., Primerano, L., Markland-Montgomery, M., Polino, M., Carminati, M., Zanero, S. (2022). Amaretto: an active learning framework for money laundering detection. *IEEE Access*, 10, 41720–41739. https://doi.org/10.1109/ACCESS.2022.3167699.

Liu, J., Yin, C., Wang, H., Wu, X., Lan, D., Zhou, L., Ge, C. (2023). Graph embedding-based money laundering detection for ethereum. *Electronics (Switzerland)*, 12(14), 3180. https://doi.org/10.3390/electronics12143180.

Lo, W.W., Kulatilleke, G.K., Sarhan, M., Layeghy, S., Portmann, M. (2023). Inspection-L: self-supervised GNN node embeddings for money laundering detection in bitcoin. *Applied Intelligence*, 53, 19406–19417. https://doi.org/10.1007/s10489-023-04504-9.

Luo, X., Han, X., Zuo, W., Xu, Z., Wang, Z., Wu, X. (2022). A dynamic transaction pattern aggregation neural network for money laundering detection. In: *Proceedings – 2022 IEEE 21st International Conference on Trust, Security and Privacy in Computing and Communications, TrustCom 2022*, pp. 818–826. https://doi.org/10.1109/TrustCom56396.2022.00114.

Oad, A., Razaque, A., Tolemyssov, A., Alotaibi, M., Alotaibi, B., Zhao, C. (2021). Blockchain-enabled transaction scanning method for money laundering detection. *Electronics (Switzerland)*, 10(15), 1766. https://doi.org/10.3390/electronics10151766.

Ofoeda, I., Agbloyor, E.K., Abor, J.Y., Osei, K.A. (2022). Anti-money laundering regulations and financial sector development. *International Journal of Finance and Economics*, 27(4), 4085–4104. https://doi.org/10.1002/ijfe.2360.

Ogbeide, H., Thomson, M.E., Gonul, M.S., Pollock, A.C., Bhowmick, S., Bello, A.U. (2023). The anti-money laundering risk assessment: a probabilistic approach. *Journal of Business Research*, 162, 113820. https://doi.org/10.1016/j.jbusres.2023.113820.

Pavlidis, G. (2023). Deploying artificial intelligence for anti-money laundering and asset recovery: the dawn of a new era. *Journal of Money Laundering Control*, 26(7), 155–166. https://doi.org/10.1108/JMLC-03-2023-0050.

Rocha-Salazar, J.-J., Segovia-Vargas, M.-J., Camacho-Miñano, M.-M. (2021). Money laundering and terrorism financing detection using neural networks and an abnormality indicator. *Expert Systems with Applications*, 169, 114470. https://doi.org/10.1016/j.eswa.2020.114470.

Salehi, A., Ghazanfari, M., Fathian, M. (2017). Data mining techniques for anti money laundering. *International Journal of Applied Engineering Research*, 146(12), 28–33. https://doi.org/10.5120/ijca2016910953.

Saragih, I.I.M. (2023). The needs of money laundering and tax evasion crimes prevention in the Asean Community. *International Journal of Scientific Multidisciplinary Research*, 1(5), 471–484. https://doi.org/10.55927/ijsmr.v1i5.4619.

Sheu, G.Y., Li, C.Y. (2022). On the potential of a graph attention network in money laundering detection. *Journal of Money Laundering Control*, 25(3), 594–608. https://doi.org/10.1108/JMLC-07-2021-0076.

Singh, K., Best, P. (2019). Anti-money laundering: using data visualization to identify suspicious activity. *International Journal of Accounting Information Systems*, 34, 100418. https://doi.org/10.1016/j.accinf.2019.06.001.

Teichmann, F.M.J., Falker, M.C. (2023). Money laundering – the gold method. *Journal of Money Laundering Control*, 26(3), 509–522. https://doi.org/10.1108/JMLC-07-2019-0060.

Thommandru, A., Chakka, B. (2023). Recalibrating the banking sector with blockchain technology for effective anti-money laundering compliances by banks. *Sustainable Futures*, 5, 100107. https://doi.org/10.1016/j.sftr.2023.100107.

Tiwari, M., Gepp, A., Kumar, K. (2020). A review of money laundering literature: the state of research in key areas. *Pacific Accounting Review*, 32(2), 271–303. https://doi.org/10.1108/PAR-06-2019-0065.

Wang, H.M., Hsieh, M.L. (2024). Cryptocurrency is new vogue: a reflection on money laundering prevention. *Security Journal*, 37(1), 25–46. https://doi.org/10.1057/s41284-023-00366-5.

Yang, G., Liu, X., Li, B. (2023). Anti-money laundering supervision by intelligent algorithm. *Computers and Security*, 132, 103344. https://doi.org/10.1016/j.cose.2023.103344.

Yu, L., Zhang, F., Ma, J., Yang, L., Yang, Y., Jia, W. (2023). Who are the money launderers? Money laundering detection on blockchain via mutual learning-based graph neural network. In: *Proceedings of the International Joint Conference on Neural Networks*. https://doi.org/10.1109/IJCNN54540.2023.10191217.

Zhang, Y., Trubey, P. (2019). Machine learning and sampling scheme: an empirical study of money laundering detection. *Computational Economics*, 54, 1043–1063. https://doi.org/10.1007/s10614-018-9864-z.

Zhong, Z., Zhu, C., Yang, Y., Liao, X., Wang, R., Zhao, Y., Zhou, F., Shi, R., Qin, Z. (2022). Money laundering detection for cryptocurrency transactions. *Hunan Daxue Xuebao/Journal of Hunan University Natural Sciences*.

A.I. Andreescu graduated from the Faculty of Cybernetics, Statistics and Economic Informatics of the Academy of Economic Studies in 2001. She got the title of doctor in economy in the specialty economic informatics in 2009. At present she is an associate professor in the Department of Economic Informatics and Cybernetics of the Bucharest University of Economic Studies. Her interest domains related to computer science are requirements engineering, business analytics, modelling languages, business rules approaches and software development methodologies.

S.-V. Oprea received the MSc degree through the Infrastructure Management Program from Yokohama National University, Japan, in 2007, her first PhD degree in power system engineering from the Bucharest Polytechnic University in 2009, and her second PhD degree in economic informatics from the Bucharest University of Economic Studies in 2017. She is currently a professor within the Faculty of Cybernetics, Statistics, and Economic Informatics with the Bucharest Academy of Economic Studies, involved in several research projects.

A.-G. Văduva earned his bachelor's degree in economic informatics in 2022 and his master's degree in databases – Support for Business in 2024. He is currently pursuing a PhD, focusing on the trustworthiness of artificial intelligence algorithms in business. Professionally, he works as an artificial intelligence engineer. His research interests include mathematics, machine learning, data mining, deep learning, and generative AI.

A. Bâra graduated the Faculty of Economic Cybernetics in 2002, holds a PhD diploma in economics from 2007. She is a professor at the Economic Informatics Department at the Faculty of Cybernetics, Statistics and Economic Informatics from The Bucharest University of Economic Studies and has coordinated three R&D projects. Her research interests are focused on data science, analytics, databases, IoT, big data, data mining, power systems, authoring more than 70 papers in international journals and conferences.