Enhancing Anti-Money Laundering with Machine Learning: Feature Engineering in Synthetic Transaction Data

August 7, 2024

Abstract

Money laundering is a growing concern in the financial sector due to its widespread prevalence and its role in facilitating criminal activities. Benefiting from advancements in data science, machine learning methods have become powerful tools for combating this issue by effectively detecting and analyzing anomalous activities. Feature engineering further enhances these methods by obtaining valuable insights from raw data. This study primarily focuses on feature engineering using the AMLSim synthetic dataset, which simulates realistic financial transactions and fraud patterns. By calculating transaction characteristics at the user level, the research evaluates how feature engineering influences the effective classification models - Random Forest and XGBoost. The study finds that incorporating engineered features enhances both the accuracy and interpretability of model predictions, enabling these models to better understand complex transaction behaviors and identify suspicious activities with greater precision. These findings underscore the contribution of feature engineering techniques in enhancing the performance of machine learning models in anti-money laundering efforts.

Contents

1 Introduction

Money laundering is the process of concealing the origins of funds obtained through illegal activities, making them appear legitimate to integrate them into the legal financial system. The United Nations Office on Drugs and Crime (UNODC) estimates that the amount of money laundered globally is approximately USD 2 trillion annually, which accounts for 2% to 5% of global GDP (Deprez et al., 2024). This staggering figure has raised significant concerns within the financial sector, highlighting the urgent need for effective anti-money laundering measures to combat this pervasive issue. In the digital era, convenient and efficient electronic financial systems have provided money laundering with more diverse and rapid methods. Meanwhile, electronic transactions create traceable digital footprints that aid in combating money laundering. The ability to detect suspicious transactions is a decisive factor in the success of anti-money laundering efforts.

The development of computational systems has enabled digital fraud detection systems to adapt to evolving fraudulent tactics by leveraging machine learning techniques to recognize complex patterns and anomalies in transaction data (West & Bhattacharya, 2016). Machine learning, with its learning capability, can analyze historical fraud patterns and other relevant data to effectively detect ongoing money laundering activities within vast amounts of transaction information. Its learning strategies enable it to identify features in fraudulent transactions that are difficult to detect using conventional rules. Machine learning techniques rely heavily on large, well-labeled datasets as the foundation for their predictions (Menon & Pottenger, 2009). However, obtaining real transaction data is challenging due to legal and privacy protections. Even anonymized real-world data often lacks effective labeling or fails to capture undetected fraudulent transactions, making it difficult to apply effectively to machine learning (Oztas et al., 2023).

To address the problem of data scarcity, researchers utilize synthetic data as an alternative to fraud detection. This data is generated using statistical models, machine learning algorithms, or simulation techniques to mimic the statistical properties of real-world data (Barse et al., 2003). Nonetheless, the simulation process often struggles to replicate the complexity of real-world scenarios. The data variety is typically limited, and the rules governing its generation may introduce biases or hidden patterns, impacting its validity for research purposes (Lopez-Rojas & Axelsson, 2012).

The feature limitations of synthetic data may fail to provide the useful features necessary for effective model prediction. To investigate this hypothesis, this study utilizes state-of-the-art fraud detection models applied to a set of synthetic bank transaction data, enhanced through feature engineering, to evaluate how feature diversity impacts prediction accuracy.

2 Review of Related Work

This section provides a comprehensive review of insights into recent studies on synthetic data, efficient modeling techniques, and the role of feature engineering as an enhancement method.

2.1 Background of Synthetic Data

In 2016, the PaySim dataset was created using agent-based simulations of a real confidential dataset (LopezRojas et al., 2016). It has limitations in providing account information because transactions are simulated using unique accounts. To address this shortcoming, IBM enhanced the Multi-Agent-Based Simulation to produce the AMLSim dataset (Ankul, 2024). The Simulation creates nodes representing accounts in a graph using NetworkX (2024) and generates transactions between these accounts. The simulation process produces semi-realistic suspicious activities with time series data using PaySim (Weber et al., 2018). The dataset has been applied in several studies utilizing tree-based classification models, among which Random Forest and Extreme Gradient Boosting (XGBoost) have shown outstanding performance. The XGBoost classifier achieved an accuracy of 99.9% when incorporated with an over-sampling method (Frumerie, 2021) In the study by Oztas et al. (2023), Random Forest achieved an 89.14% true positive rate and a 99.92% true negative rate.

2.2 Supervised Machine Learning Approach

In addition to its high performance on the synthetic dataset, Random Forest has distinguished its effectiveness and robustness among many classifier models, making it one of the most accurate fraud detection algorithms in the financial sector (Afriyie et al., 2023). It is effective for credit card fraud detection by differentiating between normal and fraudulent transactions by leveraging historical transaction data to identify patterns indicative of fraud (Xuan et al., 2018). Random Forest is particularly effective in financial fraud detection when the class distribution is imbalanced because its hierarchical structure enables learning patterns from both classes (Nami & Shajari, 2018). A study by Oztas et al. (2023) on enhancing anti-money laundering applied Random Forest as the classifier for analyzing synthetic datasets without using re-sampling methods, achieving an accuracy level of over 90%. Additionally, using techniques like the Synthetic Minority Over-sampling Technique (SMOTE) in conjunction with Random Forest further enhances the model's performance by addressing class imbalances in transaction data (Aburbeian & Ashqar, 2023).

XGBoost is also a widely used prediction model in various types of fraud detection due to its strong performance. It achieved a 0.99% AUC in credit card fraud detection research by Afriyie et al. (2023). As a scalable tree-boosting system, XGBoost is designed for fast computations on large, sparse datasets, making it highly efficient for big data applications (Chen & Guestrin, 2016). Research indicates that XGBoost outperformed both Random Forest and

a bank's current rule-based method in detecting money laundering, demonstrating its superior accuracy and adaptability (Frumerie, 2021). Its low execution time makes it particularly effective for fraud detection, as it enables rapid analysis and decision-making in real-time applications. Furthermore, integrating XGBoost with an under-sampling method has been proposed to significantly improve detection accuracy by addressing class imbalances and enhancing the model's sensitivity to minority classes (Hajek et al., 2022).

2.3 Feature engineering

Effective fraud detection with machine learning relies on thorough data pre-processing, including critical techniques such as feature engineering (Deprez et al., 2024). In general, feature engineering provides machine learning models with enhanced input data by transforming raw data into more informative features. This capability was utilized in many research studies on fraud detection. Bahnsen et al. (2016) focus on enhancing credit card fraud detection by employing advanced feature engineering techniques that leverage the temporal patterns in transaction data. They propose a transaction aggregation strategy that captures the periodic behavior of cardholders using the von Mises distribution to model transaction times. This innovative approach distinguishes between normal and fraudulent patterns by analyzing spending periodicity, resulting in an average savings increase of 13% through reduced false positives and improved detection accuracy.

Xie et al. (2019) introduce a rule-based feature engineering method that accounts for both individual and group behaviors to better capture the nuances of fraudulent transactions. They argue that frequency-based features alone are insufficient as they do not fully consider the distinct characteristics of fraudulent activities. By integrating behavioral insights at both the individual and collective levels, their approach enhances the differentiation between legitimate and fraudulent transactions, leading to more effective fraud detection models.

3 Research Objectives

This study aims to explore the impact of feature engineering on model prediction. The target data used in this study is the AMLSim dataset. As explained in previous sections, this data is generated through simulations that infer user behavior and classify fraud to create fraudulent transaction data. The generation process involves simulating user links within a graphical network to produce transactions. Then, a hypothesis can be made that transactions are based on user behavior, suggesting that there might be an implicit rule that can be captured by machine learning methods but is not noticeable to human observers. Therefore, predicting user characteristics should be integrated into the model to determine if the predictions offer insights into the entire fraud network.
The impact of feature engineering can be assessed by evaluating whether models can perform high-level feature extraction on simple data and interpret the characteristics of the entire fraud network based on these features. To provide a quantitative evaluation and address the study's goals, the following research questions are proposed:

1. How do different supervised classifiers perform in predicting fraudulent transactions using synthetic data?

2. Does feature engineering improve machine learning classifier accuracy?

3. Does the sampling method affect the accuracy?

4 Data and Methodology

4.1 Data

A comprehensive data exploration is essential for conducting meaningful feature engineering. The AMLsim synthetic dataset provides simulated banking transaction data that replicates real-world fraud behavior in accounts and transactions (Weber et al., 2018). It consists of three CSV files: *Account*, *Alert*, and *Transaction*, comprising 10,000 accounts, 1,048,575 transaction rows, and 1,719 detected fraudulent transactions.

4.1.1 Account

The dataset includes 10,000 unique accounts, all with the country "US" and account type "I", shown in Table 4.1. 1,685 accounts are identified as fraudulent. The `TX_ID` column contains five unique values, which are not explained by the original data source. All transactions occur within the provided accounts, serving as either the sender or receiver (Figure 7.1).

Column	**Count**	**Unique**	**Data Type**
ACCOUNT_ID	10,000	10,000	int64
CUSTOMER_ID	10,000	10,000	object
INIT_BALANCE	10,000	8,818	float64
COUNTRY	10,000	1	object
ACCOUNT_TYPE	10,000	1	object
IS_FRAUD	10,000	2	bool
TX_BEHAVIOR_ID	10,000	5	int64

Table 4.1: Column details of the *Account*

4.1.2 Alert

The alert file contains 1,719 detected fraudulent transactions, of which 1,329 also appear in the transaction file. The alert file includes all the columns present in the transaction file, along with a additional column that reveal the fraudulent typologies (Table 4.2). By adding 390 additional transaction rows from the alert file, the total number of transactions increases to 1,048,965.
Fraudulent transactions in the alert file are classified into two primary categories: Fan-In and Cycle. These classifications are distributed relatively evenly across the fraudulent transaction (Figure 4.1). The Fan-In typology is characterized by numerous incoming transactions converging into a single account, forming a pattern similar to a tree structure (Figure 7.3a). In this dataset, Fan-In transactions, grouped by a single alert ID, include 190 rows with transactions received from four accounts, 6 rows from three accounts, and 3 rows from fewer than three accounts (Figure 7.2). This indicates that the number of received transactions could be a relevant factor in detecting account fraud.

Column in *Alert*	Column in *Transaction*	Count	Unique	Data Type
ALERT_ID	ALERT_ID	1,719	391	int64
ALERT_TYPE		1,719	2	object
IS_FRAUD	IS_FRAUD	1,719	1	bool
TX_ID	TX_ID	1,719	1,719	int64
SENDER_ACCOUNT_ID	SENDER_ACCOUNT_ID	1,719	1,484	int64
SENDER_ACCOUNT_ID	RECEIVER_ACCOUNT_ID	1,719	1,030	int64
TX_TYPE	TX_TYPE	1,719	1	object
TX_AMOUNT	TX_AMOUNT	1,719	392	float64
TIMESTAMP	TIMESTAMP	1,719	200	int64

Table 4.2: Column details of the transaction and alert dataset

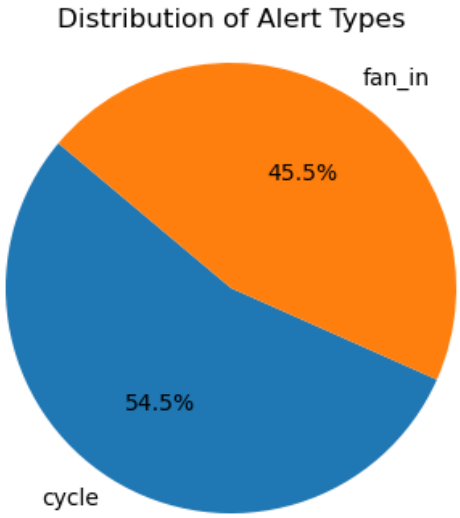


Figure 4.1: Fraudulent Typologies Distribution

The Cycle typology describes a pattern in which multiple fraudulent transactions form a loop, where money is initially transferred out of an account and eventually returned to the same account (Figure 7.3b). The depth-first search (DFS) algorithm was used to search for a path between two nodes in a graph structure (Riansanti et al., 2018). By converting each account into a node in the graph, the DFS method traced the path of money transfers in Cycle transactions. The analysis indicated that 915 out of 939 (97%) Cycle transactions formed a complete loop, while the remaining transactions were detected in a non-continuous sequence. This suggests that a graph analytic algorithm can effectively analyze Cycle transactions and contribute to detecting fraud typologies. However, the algorithm requires substantial memory resources to process ten thousand accounts within a million transactions. Due to the limitations of the machine used in this study, this study failed to retrieve supplementary information about whether all transactions were part of a complete loop.

4.1.3 Transaction

Combining the Account and Alert tables with the Transaction table using `ACCOUNT_ID` and `TX_ID` enriches the Transaction table with supplemental information. This process builds a joint dataset containing 13 columns of account and transaction behavior information, with 1,048,965 rows. Figure 4.2 shows that the joint dataset is highly skewed, with only 0.164% of transactions flagged as fraudulent. This indicates a significant data imbalance, as the vast majority of the dataset consists of normal transactions. In this case, fraudulent transactions, as the minority class, may not be effectively learned by machine learning models. This suggests the need for a data sampling method to establish a control group to evaluate whether the model can learn sufficient feature information from imbalanced data.

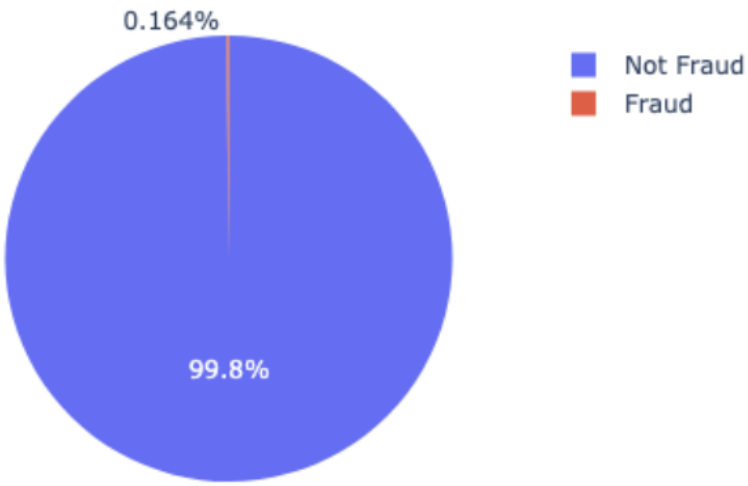


Figure 4.2: Transaction Distribution

4.1.4 Observations on Aggregated Data

Aggregating transactions at the account level provides valuable statistical insights. Additional account-related information can be obtained by calculating the cumulative debit from outgoing transactions and credit from incoming transactions, which together sum to determine the total transaction amount and final balance for each account. Figure 4.3 displays a heatmap of the correlations among these newly created variables at the account level. The observed low correlation between these variables indicates that they may provide valuable additional features for enhancing model predictions.

The Aggregated data also reveal a logical relationship between account behavior and transaction activity: A transaction is detected as fraudulent only if both the sender and the receiver are identified as fraudulent accounts. The mere presence of a fraudulent account as either the sender or the receiver does not necessarily imply that the transaction itself is fraudulent. This suggests a strong correlation between account activity and transaction behavior, which will be

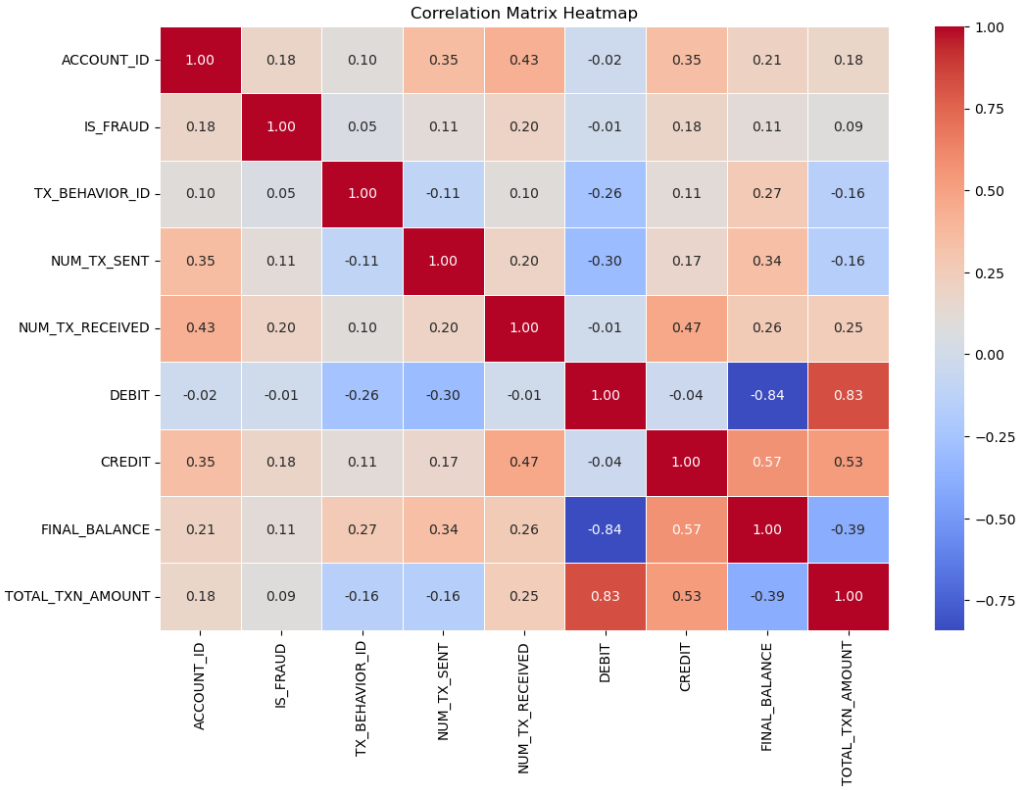


Figure 4.3: Correlation Heatmap of Accumulated Account Metrics

analyzed in a later section to determine whether it can be effectively detected by the machine learning model.

4.2 Methodology

The methodology of this study encompasses a comprehensive procedure that includes data preparation, feature engineering, model selection, and validation phases.

4.2.1 Data Preprocessing

Data Cleaning

To address the research questions, Two comparable datasets need to be constructed for the analysis: Data 1 comprises all the original features derived from the integration of the initial files. Data 2 consists of both these original features and additional supplementary features created through feature engineering.

Three files are joined on shared information to create Data 1 which contains all provided information. Consequently, a data-cleaning process is implemented to eliminate extraneous variables. ID columns and columns with singular values are considered less valuable for predicting outcomes because of their systematic and uniform nature and are therefore removed from the dataset. Following this data-cleaning process, the refined dataset is referred to as Data 1 and serves as the raw dataset for this study.

Data 1 Columns	Data 2 Columns
SENDER_ACCOUNT_ID	SENDER_ACCOUNT_ID
SENDER_TX_BEHAVIOR_ID	SENDER_TX_BEHAVIOR_ID
RECEIVER_ACCOUNT_ID	SENDER_NUM_TX_SENT
RECEIVER_TX_BEHAVIOR_ID	SENDER_DEBIT
TX_AMOUNT	SENDER_CREDIT
TIMESTAMP	SENDER_FINAL_BALANCE
SENDER_IS_FRAUD	SENDER_TOTAL_TXN_AMOUNT
RECEIVER_IS_FRAUD	RECEIVER_ACCOUNT_ID
IS_FRAUD	RECEIVER_TX_BEHAVIOR_ID
	RECEIVER_NUM_TX_SENT
	RECEIVER_DEBIT
	RECEIVER_CREDIT
	RECEIVER_FINAL_BALANCE
	RECEIVER_TOTAL_TXN_AMOUNT
	TX_AMOUNT
	TIMESTAMP
	SENDER_IS_FRAUD
	RECEIVER_IS_FRAUD
	IS_FRAUD

Table 4.3: Features in Data 1 and Data 2

The exploratory analysis in the Data section suggests that aggregating transactions can conduct feature engineering at the account level, which is used to create supplementary features in Data 2. This process involves generating cumulative information at the account level by summing the number of transactions, the total value of money sent or received, the volume of total transactions, and the ending balance. According to the observation in the data section, there is a low correlation between most variables, indicating that these variables may provide additional inferential insights.

The composition of these datasets is shown in Table 4.3. To evaluate the performance of the prediction models, both datasets were divided into training and testing subsets, with 80% of the data allocated for training and the remaining 20% reserved for testing.

Data Over-sampling

The Synthetic Minority Over-sampling Technique (SMOTE) is utilized to mitigate the unevaluated impact of data imbalance by oversampling fraudulent transactions. Data imbalance is prevalent in fraud detection as fraudulent transactions are rare compared to legitimate ones. This imbalance can lead to biased models that struggle to accurately predict the minority class, often resulting in a high number of false negatives (Ramprakash, 2021). This method aims to balance the majority class and amplify the feature significance of the minority class. SMOTE works by interpolating between existing minority class instances, creating new synthetic samples along the line segment joining two instances using a k-nearest neighbor approach (Chawla et al., 2002). The SMOTE function in this study is provided by the Python package imbalanced-learn

(Lemaitre et al., 2016). A comparison of different sampling methods is shown in Figure 4.4. This technique is applied to both datasets to increase the minority class to an even number with the majority class.

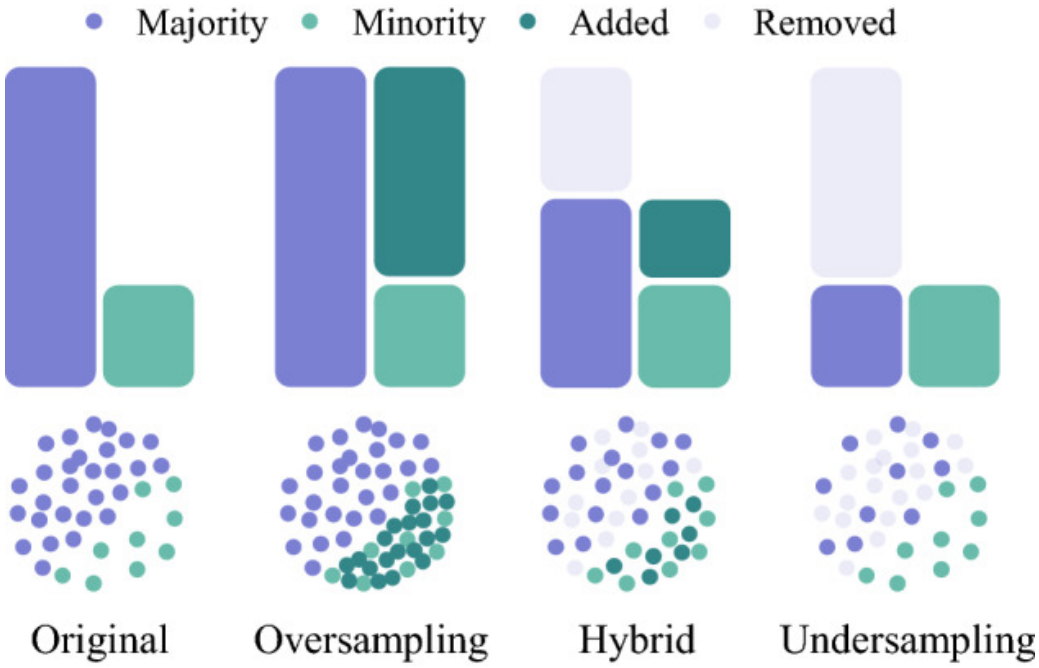


Figure 4.4: Imbalanced Processing Sampling Types (Werner de Vargas et al., 2022)

Table 4.4 illustrates the shape of the data before and after the application of the SMOTE method.

Class	**Count**	**Percentage**
Original Data		
Class = False	1,047,246	99.84%
Class = True	1,719	0.16%
Upsampled Data		
Class = False	1,047,246	99.84%
Class = True	1,047,246	99.84%

Table 4.4: Data distribution before and after over–sampling

4.2.2 Model

Following the comprehensive exploratory data analysis and data preprocessing outlined above, Random Forest and XGBoost are used to evaluate their performance on fraud transaction detection.

Random Forest

Random Forest is a supervised machine learning algorithm that employs an ensemble method to make predictions by aggregating results from multiple decision tree models (Breiman, 2001).

It divides the data into different samples using decision trees and evaluates features by randomly resampling them to assess their contribution to predictions. This approach does not require a separate feature selection procedure, which helps avoid false predictions of variables with significant variability and many possible values (Afriyie et al., 2023). Thus, it is appropriate to examine the impact of feature engineering in this study. The Python machine learning package `'sklearn'` is used to set up the Random Forest Classifier model. This package provides tools to measure a feature's importance by analyzing how much the tree nodes utilize that feature, adding more explainability to the result analysis. The model is configured with default parameters, except that the `'n_estimators'` parameter is reduced to 50 to decrease running time without significantly affecting accuracy compared to the default setting. Random Forest has demonstrated strong performance on AMLsim (Oztas et al., 2023), establishing it as a valuable benchmark for comparison with the XGBoost algorithm.

Extreme Gradient Boosting (XGBoost)

Extreme Gradient Boosting (XGBoost) is an efficient and scalable implementation of gradient-boosted decision trees, designed to enhance the model's predictions by learning from the errors of preceding trees (Hajek et al., 2022). Based on the principle of gradient boosting, it combines multiple weak learners to form a robust predictive model. XGBoost incorporates regularization techniques to prevent overfitting and improve model generalization. Additionally, it supports parallel and distributed computing, making it highly scalable for large datasets and significantly reducing training time (Chen & Guestrin, 2016). Hajek (2022) proposed that XGBoost is suitable for fraud detection when combined with an under-sampling method. The effect of incorporating an over-sampling method, such as SMOTE, remains to be tested. The XGBoost model in this research is configured using the XGBoost package in Python with default parameters. The `'use_label_encoder'` parameter is set to `'False'`, as recommended by the official documentation (XGBoost, 2021).

4.2.3 Experimental Setup

Three binary variables: `'IS_FRAUD'`, `'SENDER_IS_FRAUD'`, and `'RECEIVER_IS_FRAUD'` are set as the prediction targets. This approach evaluates the models' ability to understand and extract the interconnections among these output values. To enable Random Forest and XGBoost to provide multiple classification outputs, the MultiOutputClassifier from the `'sklearn'` package is used. The MultiOutputClassifier enables each applied model to the extraction of results for each target prediction in a single execution. It indicates the prediction result for each prediction column is consistent for an overall accuracy evaluation. The models are trained using the following data configurations:

1. Training on the original Data 1
2. Training on Data 1 with SMOTE oversampling to balance class distribution

3. Training on the original Data 2
4. Training on Data 2 with SMOTE oversampling to balance class distribution

4.3 Evaluation

The output results are evaluated separately and collectively using the following evaluation methods. Confusion matrices are used to illustrate the classification results, as they are widely used in the AML literature and provide valuable insights into the performance of classification models. Each classification result is displayed in a confusion matrix (Table 4.5) with four entries:

- **False Positive (FP)**: The number of non-fraudulent transactions incorrectly classified as fraudulent.
- **False Negative (FN)**: The number of fraudulent transactions incorrectly predicted as non-fraudulent.
- **True Positive (TP)**: The number of fraudulent transactions correctly predicted as fraudulent.
- **True Negative (TN)**: The number of non-fraudulent transactions correctly identified as non-fraudulent.

	Prediction Negative	**Prediction Positive**
Actual Negative	True Negative (TN)	False Positive (FP)
Actual Positive	False Negative (FN)	True Positive (TP)

Table 4.5: Confusion matrix

FP and FN represent the transactions that the models misclassify, which could result in significant consequences in a financial context. Improving the predictive model involves minimizing predictions in these two categories, which can be represented by the metrics Precision, Recall, and the F1-score.

Precision

Precision measures the proportion of correctly identified fraudulent transactions out of all transactions predicted as fraudulent. It is calculated as:

$$Precision = \frac{TP}{TP + FP} \tag{4.1}$$

Recall

Recall measures the proportion of actual fraudulent transactions that were correctly identified by the model. It is calculated as:

$$Recall = \frac{TP}{TP + FN} \tag{4.2}$$

F1 Score

The F1 score is the harmonic mean of precision and recall, providing a balance between the two metrics. It suggests that the model effectively identifies fraudulent activities while minimizing incorrect predictions. The F1 score is particularly useful when the class distribution is imbalanced, offering a single metric that balances precision and recall. It is used as the main metric to evaluate model performance in this study and is computed using the formula:

$$F1\ score = 2 \times \frac{Precision \times Recall}{Precision + Recall} \tag{4.3}$$

All Match Accuracy

In addition to the metrics above, an **All Match Accuracy** is defined to calculate the accuracy of matching all three prediction outputs, where a correct prediction on all three is considered true, and others are false. This metric is used to evaluate the extent of a model's understanding of prediction data. If a model has a high score, it indicates greater accuracy in predicting multiple outcomes, suggesting that the model better understands the relationships within the dataset. This implies that the model has better interpretability regarding the features. It is expressed mathematically as:

$$All\ Match\ Accuracy = \frac{Number\ of\ Correct\ All\ Three\ Outputs}{Total\ Predictions} \tag{4.4}$$

However, due to the limitations of the `imblearn` package, the resampling method does not support multiple outputs resampling, thus the overall accuracy cannot be calculated across separate prediction runs.

Run-time

Run-time is another important metric for evaluating a model's performance. A model that provides better predictions but takes significantly longer to run is not necessarily more effective. Run-time is measured during both the model training and prediction phases. In this study, all tests are conducted on a MacBook Pro equipped with an Apple M3 chip and 8GB of memory. The run-time alongside prediction accuracy provides a comprehensive assessment of the model's overall efficiency and practical applicability.

5 Results

This section presents the experimental results and highlights key observations. Table 5.1 summarizes the results of all prediction settings.

Model	Dataset	Target	Precision	Recall	F1 Score
Random Forest	1	IS_FRAUD	0.9915	0.9330	0.9613
	1	SENDER_IS_FRAUD	0.9984	0.9765	0.9873
	1	RECEIVER_IS_FRAUD	0.9965	0.9793	0.9879
	2	IS_FRAUD	0.9942	0.9249	0.9583
	2	SENDER_IS_FRAUD	0.9999	0.9987	0.9993
	2	RECEIVER_IS_FRAUD	1.0000	0.9972	0.9986
	1 SMOTE	IS_FRAUD	0.9611	0.9276	0.9441
	1 SMOTE	SENDER_IS_FRAUD	0.9852	0.9770	0.9811
	1 SMOTE	RECEIVER_IS_FRAUD	0.9871	0.9744	0.9807
	2 SMOTE	IS_FRAUD	0.9943	0.9276	0.9598
	2 SMOTE	SENDER_IS_FRAUD	0.9996	0.9960	0.9978
	2 SMOTE	RECEIVER_IS_FRAUD	0.9999	0.9974	0.9987
XGBoost	1	IS_FRAUD	0.9943	0.9303	0.9612
	1	SENDER_IS_FRAUD	0.9663	0.6559	0.7814
	1	RECEIVER_IS_FRAUD	0.8790	0.5332	0.6637
	2	IS_FRAUD	0.9943	0.9357	0.9641
	2	SENDER_IS_FRAUD	0.9956	0.8947	0.9425
	2	RECEIVER_IS_FRAUD	0.9937	0.8579	0.9208
	1 SMOTE	IS_FRAUD	0.4133	0.9517	0.5763
	1 SMOTE	SENDER_IS_FRAUD	0.6671	0.8585	0.7508
	1 SMOTE	RECEIVER_IS_FRAUD	0.6716	0.7265	0.6980
	2 SMOTE	IS_FRAUD	0.9915	0.9410	0.9656
	2 SMOTE	SENDER_IS_FRAUD	0.9403	0.9626	0.9513
	2 SMOTE	RECEIVER_IS_FRAUD	0.9323	0.9235	0.9279

Table 5.1: Evaluation Metrics for Random Forest and XGBoost Models on Different Datasets and Targets

Both models achieved an F1 score of 96% for the `‘IS_FRAUD‘` prediction, demonstrating their strong performance in fraud detection on AMLSim dataset, consistent with previous research findings. In predicting sender and receiver behavior, Random Forest outperformed XGBoost, as evidenced by higher precision, recall, and F1 scores. The lower recall of XGBoost shows that it classified more normal accounts as fraudulent compared to Random Forest. Both Random Forest and XGBoost experienced a decrease in accuracy after applying SMOTE. Random Forest showed a slight decline in F1 scores across predictions, with the `‘IS_FRAUD‘` score decreasing by 0.0172, a more significant drop than other outcome types. The impact of SMOTE on XGBoost predictions was more variable; The `‘IS_FRAUD‘` experienced the most substantial F1 score decrease, dropping by 0.3849. However, there was an improvement in recall for `‘IS_FRAUD‘` predictions, suggesting that SMOTE helped reduce false negatives in identifying fraudulent transactions, which balanced the significant decrease in precision. The feature-engineered dataset enhanced both models' performance, improving most metrics, except for a

slight decrease of 0.003 in the F1 score for Random Forest's `‘IS_FRAUD‘` prediction. XGBoost saw significant improvements with feature engineering, achieving a precision of 0.99 across all predictions, and the F1 score improved by 0.1611 and 0.2571 in predicting account behaviors. Random Forest achieved a precision score of 1.0 for `‘RECEIVER_IS_FRAUD‘`, indicating that all fraudulent receiver accounts were correctly detected.

The effect of SMOTE on the feature-engineered dataset (Data 2) was minor, leading to only slight increases or decreases in the F1 scores for Random Forest and slight increases in the F1 scores for XGBoost.

Figures **??** to **??** illustrate the importance of each feature in contributing to the prediction results. In Random Forest predictions, the transaction amount is identified as the most important feature across both datasets. The sender transaction behavior ID, which is the second most important feature in Random Forest, is deemed the most important feature in XGBoost predictions. Furthermore, some features derived from feature engineering are observed to be more effective than the original variables for XGBoost.

Table 5.2 demonstrates that XGBoost has a significantly shorter runtime compared to Random Forest. However, this advantage is offset by a lower overall accuracy. The all-match accuracy increases for both models after applying feature engineering. Due to its lower initial values, XGBoost shows a particularly noticeable improvement, although Random Forest also experiences growth from a higher baseline. Nevertheless, this improvement is accompanied by a notable increase in runtime for each model.

Model	**Dataset**	**All-Match-Accuracy**	**Execution Time (seconds)**
Random Forest	Dataset 1	0.9893	94.00
	Dataset 2	0.9988	156.31
XGBoost	Dataset 1	0.7915	2.74
	Dataset 2	0.9377	3.72

Table 5.2: Models All-Match-Accuracy and Execution Time

6 Findings and Discussion

The result presented in the previous section provides valuable insights to address research questions. The result presented in the previous section provides valuable insights to address the research question:

RQ: How do different supervised classifiers perform in predicting fraudulent transactions using synthetic data?

Both models demonstrated similar efficacy in predicting transaction fraud, each achieving an F1 score of 96%, which underscores their robustness in fraud detection. Random Forest achieved over 95% accuracy in various predictions on Data 1, demonstrating strong feature abstraction capabilities within the original dataset. In contrast, the lower accuracy of user behavior prediction by XGBoost indicates difficulties in linking data comprehension to user actions. The lower all-match accuracy observed in XGBoost suggests a less comprehensive understanding of feature interactions compared to Random Forest. Moreover, the feature importance charts reveal significant differences in feature preferences between the two models. Random Forest emphasizes transaction amounts, which appear to be more critical for understanding the dataset. Although the features selected by XGBoost are also significant in Random Forest's analysis, they do not effectively aid in predicting user behavior. Nonetheless, XGBoost required only 2% of the prediction time needed by Random Forest. In terms of transaction prediction alone, it is far more efficient than Random Forest.

RQ: Does feature engineering improve machine learning classifier accuracy?

The results indicate that simple feature engineering, such as data aggregation, improves model accuracy. This suggests that models can extract more useful information from the additional features, enhancing prediction results. The significant improvement on XGBoost hints the feature adequacy could be essential to gradient boosting algorithm. The change in feature importances landscape implies the feature engineer have provide the characteristic that the model are unable to learning. However, it may also have introduced noise into the predictions, as evidenced by the decreased accuracy of Random Forest in detecting fraudulent transactions. Arguably, this outcome could also be positive, as it might reduce the model's overfitting to the dataset, thereby enhancing its ability to generalize to other data samples. Since this study did not include additional datasets for validation, the issue of overfitting cannot be fully assessed

RQ: Does the sampling method affect the accuracy?

The impact of the over-sampling strategy on Random Forest is minor. Because the over-sampling strategy differs from the method used to generate the synthetic data, it is difficult to effectively validate whether the generated data aligns with the simulator. However, the stability of the prediction results demonstrates Random Forest's consistency in its predictions,

further indicating its ability to make judgments based on a solid understanding of the data. The impact of the over-sampling strategy on datasets with more features is minimal. However, it significantly affects XGBoost's predictions on the original data, leading to both improvements and declines. This may be due to XGBoost's tendency to heavily weigh certain features in its predictions, with the SMOTE strategy introducing additional data that amplifies these features. As a result, highly correlated features are also enhanced, while those with less correlation may decline. This demonstrates XGBoost's high sensitivity to features, where a small number of features can significantly impact predictions.

Overall, based on the current data results, Random Forest demonstrates a superior ability to understand the data compared to XGBoost. Feature engineering effectively enhances model prediction performance by uncovering additional hidden features, which help the model achieve a better understanding of the data. This conclusion is yet to be proved since this study is limited by its validation results involving only two models. The current feature engineering is based on statistical results. More complex features are yet to be included and investigated in future studies. Moreover, the prediction data is derived from the AMLSim project's synthetic data, which is relatively limited in feature variety compared to real-world transaction data. Therefore, the applicability of the method in this study remains to be tested in real-world scenarios. Future research will include incorporating additional deep-learning models for more complex feature extraction and conducting experimental comparisons with more cutting-edge models. Future work could explore more sophisticated data augmentation techniques and investigate the potential for prediction improvement through data simulation based on generative adversarial algorithms.

7 Conclusion

In this study, an experiment is conducted to examine the effect of feature engineering along with the over-sampling method on machine learning performance in fraud detection. Two machine learning models, Random Forest and XGBoost are built to perform prediction on a synthetic dataset, AMLSim. The results suggest that statistical feature engineering enhances model performance by effectively extracting critical information, leading to improved model accuracy and interpretability in detecting fraudulent transactions. Additionally, the study reveals that the over-sampling method does not significantly improve prediction results for imbalanced data. In some cases, it has a noticeable negative impact on specific predictions. This may be due to the introduction of noise or bias during the up-sampling process, which can affect the models' ability to accurately identify minority class instances. The experimental results of this study contribute valuable insights into the role of data engineering in the application of machine learning models for anti-money laundering purposes. By highlighting the benefits and limitations of feature engineering, this research can guide future efforts in improving machine learning methods and data generation for fraud detection.

Appendix

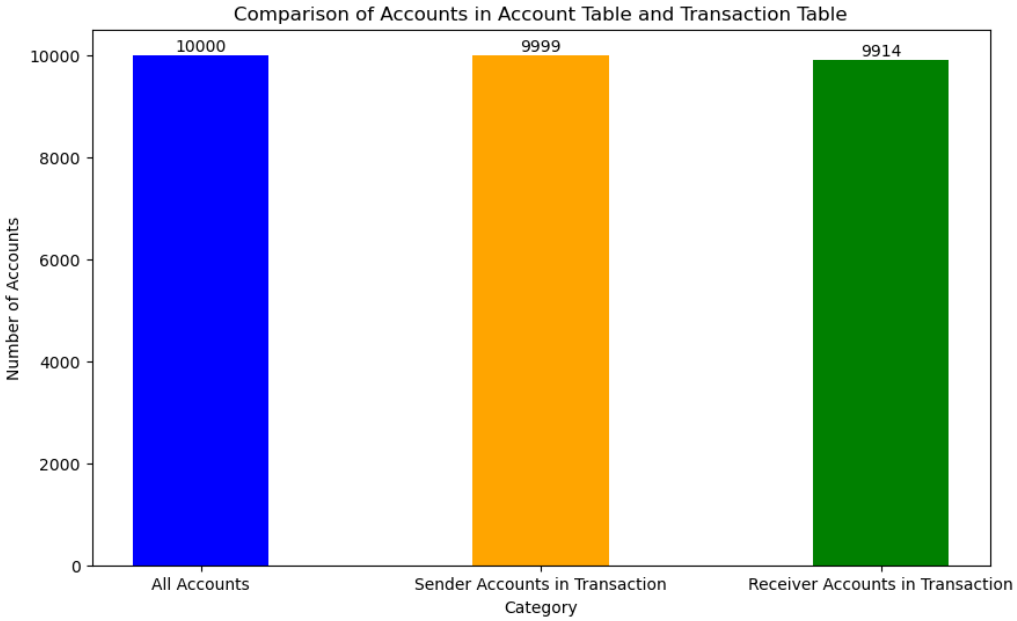


Figure 7.1: Distribution of Accounts as Senders and Receivers

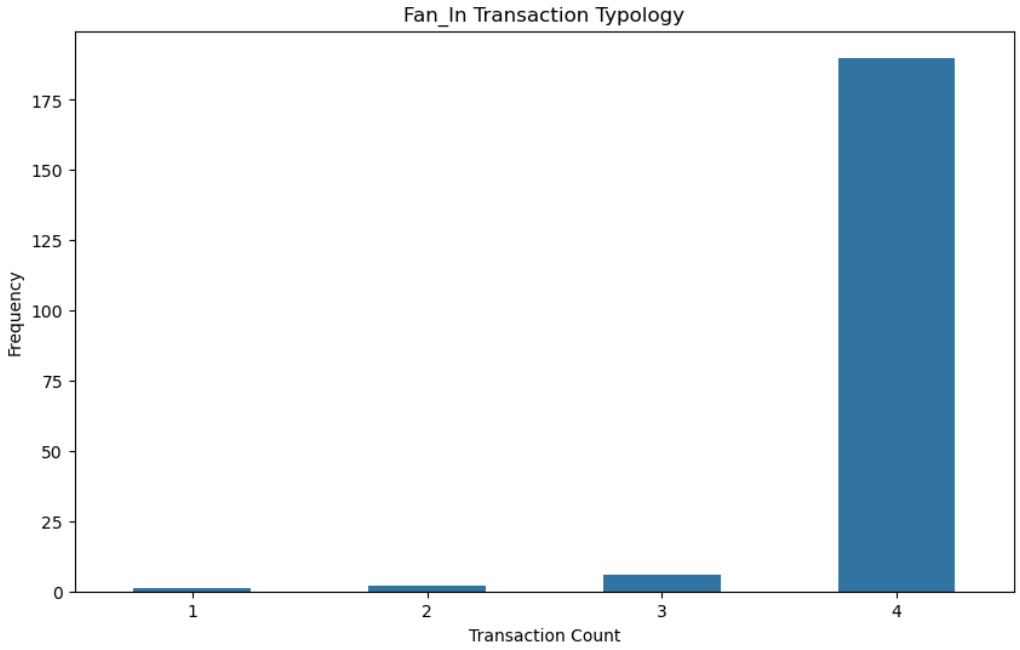
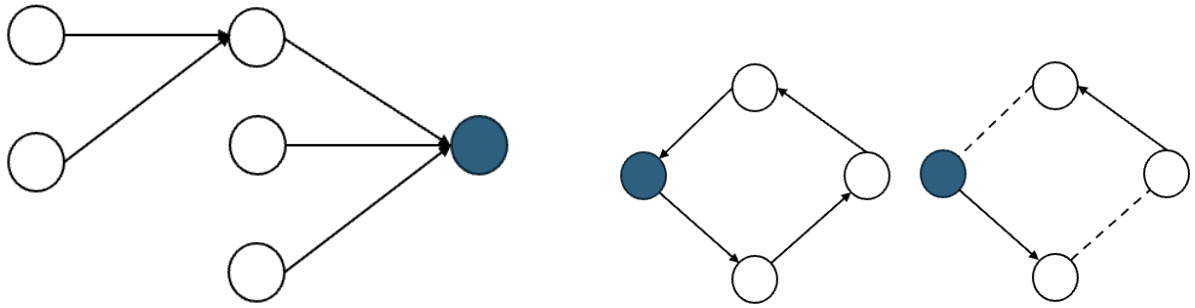


Figure 7.2: Number of Transaction in Fan_In



(a) Fan_in (b) Cycle

Figure 7.3: Fraudulent Typologies

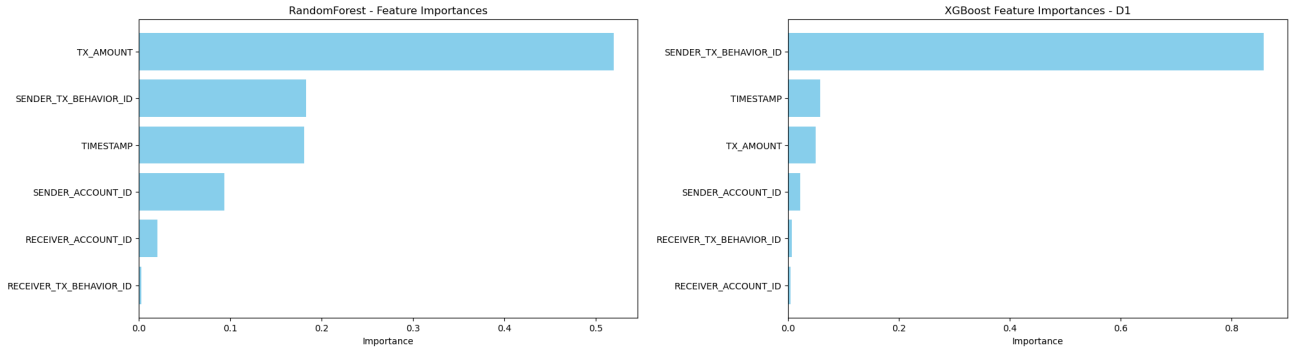


Figure 7.4: Feature Importance on Data 1

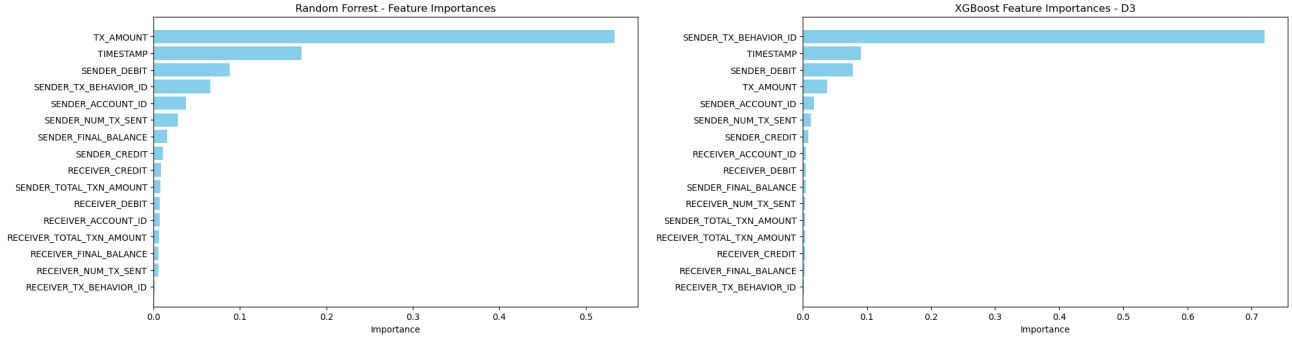


Figure 7.5: Feature Importance on Data 2

Bibliography

Aburbeian, A. M. & Ashqar, H. (2023), 'Credit card fraud detection using enhanced random forest classifier for imbalanced data', *arXiv (Cornell University)* .

Afriyie, J. K., Tawiah, K., Pels, W. A., Addai-Henne, S., Dwamena, H. A., Owiredu, E. O., Ayeh, S. A. & Eshun, J. (2023), 'A supervised machine learning algorithm for detecting and predicting fraud in credit card transactions', *Decision Analytics Journal* **6**, 100163.

Ankul, A. (2024), 'Ibm amlsim example dataset'.
URL: *https://www.kaggle.com/datasets/anshankul/ibm-amlsim-example-dataset/data*

Barse, E. L., Kvarnström, H. & Jonsson, E. (2003), 'Synthesizing test data for fraud detection systems'.
URL: *https://www.acsac.org/2003/papers/74.pdf*

Breiman, L. (2001), 'Random forests', *Machine Learning* **45**, 5–32.
URL: *https://link.springer.com/article/10.1023/A:1010933404324*

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

Chen, T. & Guestrin, C. (2016), 'Xgboost: a scalable tree boosting system', *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining - KDD '16* p. 785–794.

Correa Bahnsen, A., Aouada, D., Stojanovic, A. & Ottersten, B. (2016), 'Feature engineering strategies for credit card fraud detection', *Expert Systems with Applications* **51**, 134–142.

Deprez, B., Vanderschueren, T., Baesens, B., Verdonck, T. & Verbeke, W. (2024), 'Network analytics for anti-money laundering – a systematic literature review and experimental evaluation'.
URL: *https://arxiv.org/abs/2405.19383*

Frumerie, R. (2021), 'Money laundering detection using tree boosting and graph learning algorithms'.
URL: *https://www.diva-portal.org/smash/get/diva2:1663255/FULLTEXT01.pdf*

Hajek, P., Abedin, M. Z. & Sivarajah, U. (2022), 'Fraud detection in mobile payment systems using an xgboost-based framework', *Information Systems Frontiers* .

Lemaitre, G., Nogueira, F. & Aridas, C. K. (2016), 'Imbalanced-learn: A python toolbox to tackle the curse of imbalanced datasets in machine learning'.

Lopez-Rojas, E. A. & Axelsson, S. (2012), 'Money laundering detection using synthetic data'.
URL: *https://ep.liu.se/ecp/071/005/ecp12071005.pdf*

LopezRojas, E. A., Elmir, A. & Axelsson, S. (2016), *PAYSIM: A FINANCIAL MOBILE MONEY SIMULATOR FOR FRAUD DETECTION*.

Menon, V. & Pottenger, W. M. (2009), 'A higher order collective classifier for detecting and classifying network events', *CiteSeer X (The Pennsylvania State University)* .

Nami, S. & Shajari, M. (2018), 'Cost-sensitive payment card fraud detection based on dynamic random forest and k -nearest neighbors', *Expert Systems with Applications* **110**, 381–392.

NetworkX (2024), 'Networkx — networkx documentation'.
URL: *https://networkx.org/*

Oztas, B., Cetinkaya, D., Deniz Adedoyin, D., Budka, M., Dogan, H. & Aksu, G. (2023), 'Enhancing anti-money laundering: Development of a synthetic transaction monitoring dataset'.

Ramprakash, A. (2021), 'Implementation of xgboost ensemble learning model for detecting money laundering', *International Journal for Research in Applied Science and Engineering Technology* **9**, 312–316.

Riansanti, O., Ihsan, M. & Suhaimi, D. (2018), 'Connectivity algorithm with depth first search (dfs) on simple graphs', *Journal of Physics: Conference Series* **948**, 012065.

Weber, M., Chen, J., Suzumura, T., Pareja, A., Ma, T., Kanezashi, H., Kaler, T., Leiserson, C. & Schardl, T. (2018), 'Scalable graph learning for anti-money laundering: A first look'.
URL: *https://arxiv.org/pdf/1812.00076*

Werner de Vargas, V., Schneider Aranda, J. A., dos Santos Costa, R., Ricardo da Silva Pereira, R. & Victória Barbosa, J. L. (2022), 'Imbalanced data preprocessing techniques for machine learning: a systematic mapping study', **65**, 31–57.
URL: *https://www.ncbi.nlm.nih.gov/pmc/articles/PMC9645765/*

West, J. & Bhattacharya, M. (2016), 'Intelligent financial fraud detection: A comprehensive review', *Computers Security* **57**, 47–66.

XGBoost (2021), 'Python api reference — xgboost 1.3.3 documentation'.
URL: *https://xgboost.readthedocs.io/en/release*$_1$*.3.0/python/python*$_a$*pi.html*

Xie, Y., Liu, G., Cao, R., Li, Z., Yan, C. & Jiang, C. (2019), 'A feature extraction method for credit card fraud detection'.
URL: *https://ieeexplore.ieee.org/abstract/document/8782457/*

Xuan, S., Liu, G., Li, Z., Zheng, L., Wang, S. & Jiang, C. (2018), 'Random forest for credit card fraud detection'.
URL: *https://ieeexplore.ieee.org/abstract/document/8361343/figures*