MAD: Multicriteria Anomaly Detection of Suspicious Financial Accounts from Billions of Cash Transactions

MAD: Multicriteria Anomaly Detection of Suspicious Financial Accounts from Billions of Cash Transactions

Giordano Paoletti*
Politecnico di Torino
Turin, Italy
giordano.paoletti@polito.it

Flavio Giobergia*
Politecnico di Torino
Turin, Italy
flavio.giobergia@polito.it

Danilo Giordano
Politecnico di Torino
Turin, Italy
danilo.giordano@polito.it

Luca Cagliero
Politecnico di Torino
Turin, Italy
luca.cagliero@polito.it

Silvia Ronchiadin
Anti Financial Crime Digital Hub
Turin, Italy
silvia.ronchiadin@intesasanpaolo.com

Dario Moncalvo
Anti Financial Crime Digital Hub,
Intesa Sanpaolo
Turin, Italy
dario.moncalvo@intesasanpaolo.com

Marco Mellia
Politecnico di Torino
Turin, Italy
marco.mellia@polito.it

Elena Baralis
Politecnico di Torino
Turin, Italy
elena.baralis@polito.it

Abstract

This paper presents a real-world deployment case study on using unsupervised anomaly detection for Anti-Money Laundering (AML). Using more than 2 billion anonymized bank transactions that Intesa Sanpaolo, a primary Italian financial institution, registered over 8 months, we developed, tuned and deployed a machine learning pipeline in production. Experts from Intesa Sanpaolo validated the performance of our approach against the institution's traditional rule-based system and checked new real-world cases the system allowed them to identify. Besides increasing both precision and recall by a factor of 6 in the detection of high-risk cases, our pipeline raises 200+ additional alerts during the 8-month period, manually identified by branch managers, but missed by the rule-based system. More importantly, a manual inspection of 100 new unseen cases revealed 28 significant previously unreported cases. The pipeline, now fully deployed in Intesa Sanpaolo's Transaction Monitoring system, highlights the advantages of machine learning over traditional approaches typically adopted in this traditionally very conservative sector.

CCS Concepts

• **Computing methodologies** → **Anomaly detection**; • **Applied computing**;

Keywords

Unsupervised Anomaly Detection, Anti-Money Laundering, Transaction Monitoring

*Equal contribution.

KDD '25, Toronto, ON, Canada

ACM ISBN 979-8-4007-1454-2/2025/08
https://doi.org/10.1145/3711896.3737244

ACM Reference Format:
Giordano Paoletti, Flavio Giobergia, Danilo Giordano, Luca Cagliero, Silvia Ronchiadin, Dario Moncalvo, Marco Mellia, and Elena Baralis. 2025. MAD: Multicriteria Anomaly Detection of Suspicious Financial Accounts from Billions of Cash Transactions. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V.2 (KDD '25), August 3–7, 2025, Toronto, ON, Canada.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3711896.3737244

1 Introduction

Individuals and organizations that engage in criminal behaviours have financial needs, ranging from payments and investments to money laundering and circumvention of sanctions and embargoes. These entities seek to meet their financial needs through the services offered by regular financial institutions, e.g. banks. As a consequence, Financial Intelligence Units (FIU) worldwide require financial institutions to report activities that are considered as suspect. To comply with this necessity, financial institutions rely on *Transaction Monitoring* (TxM) systems that process hundreds of millions of transactions looking for suspicious operations. This process is typically run every month, as alarms do not need to be raised in real-time (as it would be, for instance, for credit card transactions).

In past years, rule-based TxM systems have allowed the detection of simple cases of money laundering and circumventions of sanctions. However, individuals and organizations can easily bypass these often simplistic rules. More recently [7, 12, 22], Machine Learning (ML) started becoming a valuable alternative to traditional, rule-based systems. Although a wide variety of ML-based techniques have been proposed [5, 11, 26], to the best of our knowledge we are the first to provide a detailed account of the full end-to-end process, in a real scenario, with a deployed result. This paper presents the successful case of MAD (Multicriteria Anomaly Detection). We designed, tested, and optimised an unsupervised machine learning pipeline and effectively deployed it in the Intesa Sanpaolo TxM system, with results validated by domain experts.

MAD focuses on unsupervised anomaly detection models. We compare and test multiple algorithms, and integrate a post-processing step to enhance the detection of suspicious activity considering cash-based transactions. Domain experts validated our results by comparing them against (i) the institution's current rule-based system and (ii) manually detected fraud cases. MAD successfully identified all high-risk cases detected by the rule-based system, while reducing the overall number of alarms, a key advantage allowing the reduction of the load on the subsequent manual case-by-case verification. Additionally, MAD flagged 200+ cases deemed highly suspicious, that were missed by the rule-based system. A manual review of a sample of flagged accounts reveals that 25-30% of them correspond to never-reported but suspicious accounts. In particular, we show that MAD identified suspicious cases that were previously completely undetected, highlighting the potential of unsupervised learning in discovering previously unknown patterns. Our work demonstrates the potential of ML pipelines in fighting financial crimes, particularly in sectors typically deemed conservative.

The remainder of this paper is organized as follows. Section 2 provides the necessary background on TxM and financial regulations. Section 3 discusses related work, highlighting the strengths and weaknesses of existing AML detection methods. Section 4 defines the problem and describes the dataset used in this study. Section 5 details our proposed methodology. Section 6 presents experimental results and validation, followed by Section 7, which discusses the deployment of our approach. Finally, Section 8 concludes the paper by presenting the lessons we learned.

2 Background

In this section, we establish a common vocabulary for the rest of the paper.

A *customer* of a financial institution is defined as the entity that interacts with a financial institution. This customer can be, for instance, a physical person or a legal entity. The customer can interact with the bank through an *account* (e.g., a checking or a savings account), identified by an *accountID*, e.g., the ABA routing transit number in the US, or the International Bank Account Number (IBAN) in Europe. A customer can potentially use multiple accounts for different purposes, and an account can be jointly held by multiple customers.

Each account can be used to perform *transactions*. Transactions can involve the bank and the customer (e.g., withdrawing money from an ATM), or two parties (e.g., a wire transfer).

Financial Intelligence Units (FIUs) are government agencies tasked with the collection and investigations of suspicious or unusual financial activities. FIUs act as an intermediary between financial institutions and law enforcement agencies. Financial institutions, such as banks, credit unions, insurance companies, etc., are required to report suspicious activities to FIUs by means of a *Suspicious Activity Report* (SAR). These SARs need to contain both information on who performed the suspicious activity (the parties involved, or the *subjects* of the SAR), as well as the set of financial operations that characterize the suspicious activity (the transactions, or the *objects* of the SAR).

To fulfill this requirement, financial institutions implement *Transaction Monitoring* (TxM) activities. TxM requires sifting through hundreds of millions of transactions performed monthly by accounts and identifying potentially suspicious activity. Because of these volumes, the first step of TxM is typically automated, often with rule-based detection systems. These detections undergo multiple investigations, at different levels of detail, by multiple *human operators*, who are part of the so-called *competence centre*. This centre operates on two levels: L1 and L2. L1 performs an initial analysis to discard clearly irrelevant detections. Cases deemed interesting pass to L2, which performs additional investigations. The outcome of those investigations is that the detection is either a *false positive*, i.e. it is deemed not suspicious, or it is considered suspicious enough to generate a SAR, which is submitted to the FIU. The FIU will continue with the investigation, and act accordingly, potentially involving law enforcement agencies. The FIU does not generally provide an explicit outcome of the investigation, but only information on whether the SAR has been archived (considered not interesting) or non-archived. For the purposes of this work, we consider non-archived SARs as *true positives*.

3 Related Works

The adoption of machine learning techniques for the detection of suspicious activity is well-established in the literature. Research has explored various approaches to enhance the effectiveness of anti-money laundering (AML) systems, focusing on critical themes such as supervision, dataset imbalance, and practical deployment challenges.

Many studies in the literature focus on supervised learning techniques due to their ability to achieve high accuracy when sufficient labeled data is available. While there is a predominancy of usage of well-known machine learning algorithms, such as decision trees, Random Forest, and SVMs [12, 16, 26], some studies employ simple feedforward neural network architectures, [5, 20, 23, 26]. More recently, researchers have also explored more complex models, including LSTM, GRU, Transformers [11], and Graph Neural Networks [4, 5]. However, the intensive effort required for dataset annotation lays the limitation, often leading to small training sets that limit generalizability and may hinder data-intensive architectures. Semi-supervised approaches attempt to address these limitations by combining limited labeled data with a larger set of unlabeled transactions. Usman et al. [23] use a self-supervised GCN [13] to generate user embeddings, which are then utilized as features for a supervised model. Rouhollahi et al. [20] begin with a small labeled dataset and expand annotations using Snorkel [19]. Meanwhile, Labanca et al. [14] rank samples based on anomaly scores using an isolation forest model, then manually annotate the top anomalies for training. The rapid evolution of money laundering techniques, combined with the challenges of adapting to incomplete and evolving data in real-time, often leaves these models struggling to keep pace. Without the ability to dynamically update datasets and account for shifts in behavior or adversarial strategies, these systems risk becoming reactive, rather than proactive, in addressing emerging threats [18]. For this reason, we opt for an unsupervised approach.

Unsupervised learning techniques used in AML focused on anomaly detection and cluster analysis. Algorithms like Isolation Forest, One-Class SVM, and Local Outlier Factors [3, 7, 10, 17] detect samples deviating from the normal data distribution. However, what

is anomalous is not necessarily suspicious/fraudulent, so the false positive rate can be high. Other techniques focus on cluster analysis. Anomalous samples are grouped to form clusters representing abnormal behaviors, as demonstrated in [6]. Alternatively, clustering defines groups of similar instances, with samples far from these clusters indicating anomalous behavior, as seen in [25].

Another critical limitation in existing literature is the simplicity of experimental setups, which often fail to reflect the complexity of real-world conditions. Many studies work with datasets containing some thousands of transactions/account [1, 12, 26]. These scales are orders of magnitude smaller than the millions of accounts addressed in our study, where computational challenges become far more pronounced. Another issue is the oversimplification of the anomaly detection task through artificially balanced datasets or inflated anomaly rates. For example, some works employ down-sampling techniques on legitimate transactions to handle imbalance, such as [5], or they focus on different scenarios where they have to identify from alerts which ones are most likely to be SARs [1, 11]. In contrast, [7] provides a more realistic setup by working with datasets where anomalies represent 0.01% of the total transactions, reflecting the challenging conditions of real-world scenarios. Our work aligns with and extends these efforts by tackling the dual challenges of scale and extreme imbalance. We focus on datasets with millions of accounts and anomalies comprising only 0.01% of the transactions, emphasizing the need for robust and scalable methods capable of detecting rare events in realistic, high-complexity environments.

At last, to the best of our knowledge, no previous work reports the experience in the production system. Here, we not only design, test and tune an anomaly detection pipeline able to process hundreds of millions of transactions and accounts, but also report the experience coming from an actual deployment in the production TxM of Intesa Sanpaolo.

4 Problem definition

We refer to the set of all existing customers as $C = \{c_1, c_2, ...\}$. Similarly, we refer to each existing account as $A = \{a_1, a_2, ...\}$.

We model the relationship between customers and accounts with two functions, $owns(\cdot) : C \to \mathcal{P}(A) \setminus \emptyset$, and $owned(\cdot) : A \to \mathcal{P}(C) \setminus \emptyset$, where $\mathcal{P}(X)$ represents the power set of X. The power set is necessary because customers can own multiple accounts, and an account can be jointly held by multiple customers, the *primary* customer being identified as the owner of an account. We refer to this relationship as $powns(\cdot) : A \to C$. Each account is associated to an account balance, the amount of money available: $balance(\cdot) : A \to \mathbb{R}$.

Each customer is associated with a *customer segment*, as determined during the *Know Your Customer* (KYC) due diligence phase. There is a limited set S of known customer segments (e.g. natural person, legal entity, institution). We associate a customer to its segment via $segment(\cdot) : C \to S$.

For this work, we identify two main types of transactions: CASH transactions, i.e. transactions involving cash (withdrawals, deposits) and WIRE transactions (i.e., wire transfers, online payments, etc.) involving two parties.

A CASH transaction can be fully described as a 4-tuple $t_{CASH} = (a, m, d, t)$, where $a \in A$ is the account performing the operations, $m \in \mathbb{R}$ is the amount of the operation (positive if the amount is deposited on the account, negative otherwise), d and t represent the date and time of the operation, respectively.

A WIRE transaction is slightly more complicated, as it includes the *originator* and the *beneficiary* of the operation, i.e. two parties. As such, the transaction can be represented as a 5-tuple $t_{WIRE} = (a_d, a_s, m, d, t)$, where $a_d,\ a_s \in A$ are the parties involved in the operation [1]. If a_d is the beneficiary and a_s is the originator, the amount m is positive, otherwise it is negative. We conventionally refer to specific attributes of a transaction by means of functions, when unambiguous: for instance, we use $a_d(t)$, $m(t)$ to refer to the account and the amount associated with transaction t, respectively.

We refer to T_{CASH} and T_{WIRE} as the sets of all CASH and WIRE transactions processed by the financial institution.

To preserve the consistency of the results, as well as to limit the volumes of data involved, TxM is generally performed over batches of data that span a specific time span (e.g., on a monthly basis). In other words, we can define a window of time, delimited by a beginning d_{from} and an end d_{to}, and consider only the transactions occurring within that time frame:

$$T_{CASH}|_{d_{from}}^{d_{to}} = \{t \in T_{CASH}, d_{from} \leq d(t) < d_{to}\}$$
$$T_{WIRE}|_{d_{from}}^{d_{to}} = \{t \in T_{WIRE}, d_{from} \leq d(t) < d_{to}\}$$

For ease of notation, we use T_{CASH}, T_{WIRE} with the underlying assumption that the sets of transactions refer to a predefined time window.

Finally, we identify the set of transactions related to a specific account $a = \alpha$ as $T_{CASH,\alpha}$, $T_{WIRE,\alpha}$ and T_α defined as $T_{CASH,\alpha} = \{t \in T_{CASH}, a(t) = \alpha\}$, $T_{WIRE,\alpha} = \{t \in T_{WIRE}, a(t) = \alpha\}$, $T_\alpha = T_{CASH,\alpha} \cup T_{WIRE,\alpha}$.

Based on this notation, we set the goal of producing as a set of subjects of interest, A_{pred}, i.e. that call for further evaluation by the competence centre and eventually become part of a SAR.

The definition of a ground truth to use for the evaluation of the performance of the TxM system is tricky. The FIU investigates the received SARs and decides whether to act upon it. The ideal ground truth would be the outcome of whether the FIU effectively decided to involve law enforcement agencies, indicating an actual illicit activity. However, using this ground truth poses several problems: first, the FIU does not necessarily provide any feedback (positive or negative) on the validity of the SAR, and even when it does, the investigations that led to the outcomes are based on information that is not typically available to financial institutions. Second, the FIU investigates only reported cases, with the risk that false negatives are ignored.

4.1 Dataset

The dataset used for the development of MAD contains the anonymized transactions processed by Intesa Sanpaolo. The transactions cover a time span of 8 months, from November 2022 to June 2023. The

[1]We note that, for wire transfers, one of the parties is guaranteed to be an account owned by a financial institution customer, whereas the other account may be registered elsewhere. As such, a full characterization of those accounts may not be possible. Since the main focus of this paper is on CASH transactions, we can ignore this problem.

Table 1: Cardinalities of the available datasets, separately for each month, in terms of number of transactions and number of unique accounts involved.

Month	Total Transactions $\|T\|$	Unique Accounts $\|A\|$
Nov 2022	279,029,599	14,503,838
Dec 2022	313,481,961	14,631,119
Jan 2023	277,222,213	14,894,278
Feb 2023	262,678,554	14,417,712
Mar 2023	300,891,611	15,461,731
Apr 2023	269,555,407	14,964,225
May 2023	310,595,368	14,652,299
Jun 2023	300,568,438	14,839,518

cardinality of the dataset is reported in Table 1, separately for each month.

We observe about 300 million transactions per month performed by about 15 million accounts. The variety of transactions and activities makes it difficult to process the entire dataset with a single approach. As a consequence, in this work, we focus on a well-defined slice of all transactions: those involving cash operations. The adoption of the cash perimeter provides a homogeneous structure in the transaction: no counterparty is involved in the operations. With this assumption, we establish the set of accounts of interest as $A^{(1)} \triangleq \{\alpha \in A \ : \ \exists t \in T_{CASH},\ a(t) = \alpha\}$.

4.2 Focusing on interesting cases

Among these, a significant fraction of accounts perform limited activity either in the amount of money moved or in the number of operations. First, these accounts are, based on domain knowledge (as well as common sense), considered less interesting: for an account to have a meaningful impact we expect it to perform operations that move a sizeable amount of money. Second, accounts with limited activity are more difficult to model, as only limited information is available to describe their behaviours. For these reasons, we define two thresholds, $\tau_\# = 3$, $\tau_\Sigma = 1,000$, to define the minimum number of operations performed, and the minimum amount moved (in absolute value), respectively. We thus introduce:

$$A^{(2)} \triangleq \{\alpha \in A^{(1)} \ : \ |T_{CASH,\alpha}| \geq \tau_\#\}$$
$$A^{(3)} \triangleq \{\alpha \in A^{(2)} \ : \ \sum_{t \in T_{CASH,\alpha}} m(t) \geq \tau_\Sigma\}$$

At last, we remove those accounts that have null, or negative average account balances throughout the month. These edge cases (less than 0.1% of the accounts) produce artefacts in the normalization steps downstream. We remove those accounts: $A^{(4)} = \{\alpha \in A^{(3)} \ : \ balance(\alpha) > 0\}$.

Finally, we focus only on accounts owned by natural persons. This provides a more homogeneous type of behaviour than considering all owners: other customer segments (e.g., companies, institutions) have a wider variety of behaviours, that make the unsupervised detection of outliers a noisier task: $A^{(5)} = \{\alpha \in A^{(4)} \ : \ segment(powns(\alpha)) = PERS\}$.

Table 2: Summary of Account Cardinalities After Each Filter in May 2023, as an example.

Operation	$\|\mathbf{A^{(i)}}\|$	$\|\mathbf{A^{(i)}}\|/\|\mathbf{A^{(i-1)}}\|$	$\|\mathbf{A^{(i)}}\|/\|\mathbf{A^{(0)}}\|$
Original accounts, $A \equiv A^{(0)}$	14,652,299	100%	100%
Cash perimeter, $A^{(1)}$	6,184,570	42.2%	42.2%
# transactions ≥ 3, $A^{(2)}$	2,900,805	46.9%	19.8%
Total amount $\geq 1,000$, ($A^{(3)}$	1,399,255	48.2%	9.5%
Account balance > 0, $A^{(4)}$	1,398,387	99.9%	9.5%
Customer segment (PERS), $A^{(5)}$	995,907	71.2%	6.8%

We report, as a part of Table 2, the reduction in cardinality that occurs with each step of the definition of the perimeter, for a specific example (May 2023). It can be seen that the final set $A^{(5)}$ contains approximately 1 million accounts to analyse. Besides reducing the overall scale of the data, these reduction steps produce a more homogeneous setting, where anomalies can be better identified (and later verified).

5 Methodology

We propose a machine learning pipeline for unsupervised anomaly detection, integrating the classic steps of feature engineering, anomaly detection algorithm selection, and post-processing steps.

We first aggregate data at the account level. Then, we apply the filters that define the perimeter of interest. We extract domain-specific features to represent each account for the unsupervised anomaly detection step. The identified outliers are ranked based on their level of anomaly. At last, we introduce a post-processing step to refine the results and enhance the overall performance. Figure 1 provides an overview of MAD, including a validation framework to compare predictions with the available ground truth.

5.1 Feature Engineering

We work closely with domain experts to design features that effectively describe account activity. For each account, we extract features based on all transactions performed during the period under analysis. Instead of focusing on individual transactions, we aim to capture the overall behavioural patterns of each account. To do this, we group all transactions by account and extract features from the aggregated data.

Table 3 provides the detailed definitions of these features. We explicitly separate credit (C) and debit (D) transactions, allowing us to distinguish between in- and out-flows of money. Additionally, we compute features for the total flow (*all*), combining credit and debit operations. For monetary features, we normalize the values by dividing either by the total amount moved by the account during the month (i.e., the fraction of the total flow) or by the account balance (i.e., the fraction of the account's potential movement capacity). We summarize the value distributions of features with their *mean*, *median*, *standard deviation*, *minimum*, and *maximum*, for each feature. Finally, we define so-called *envelop* features that quantify the ratio between an account's inflows and outflows. We normalize the ratio to the $[0, 1]$ range using a saturation function defined as follows:

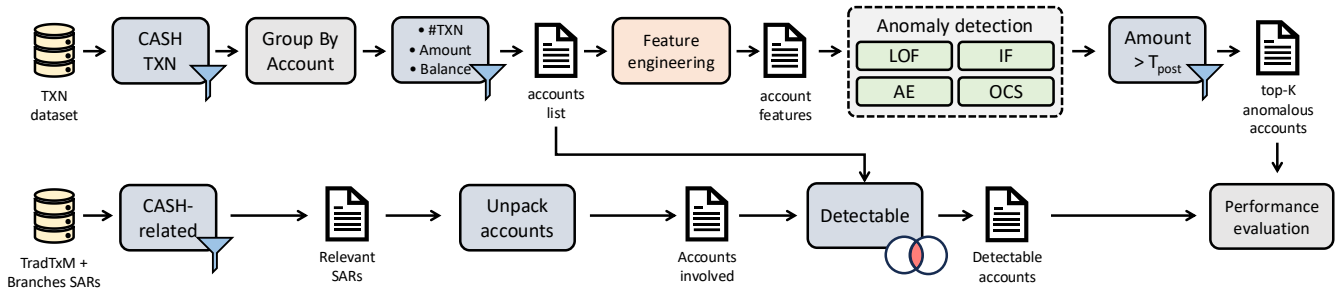

Figure 1: Architecture of the proposed MAD pipeline. The lower part represents the validation pipeline (i.e., extraction of the GT and comparison with the results).

Table 3: Features summary: We separate credit (C) and debit (D) transactions to distinguish in- and out-flows, with total flow (all) combining both. Unless otherwise specified, we only consider transactions within the CASH perimeter

Feature Name	Description	Count
	Customer/account information	
Account Balance	Account average monthly balance, from the original dataset.	1
	Transaction features	
Count (C, D, All)	Number of transactions	3
Total Amount (C, D, All)	Total amount moved in transactions unnormalized or normalized by Account Balance or by Month Volume	8
Total Amount Wire (C,D)	Total amount of money moved in transactions of the wire perimeter	2
Amount per transaction (C, D, All)	Average/median/std/min/max amount amount moved in transactions, unnormalized or normalized by Account Balance or by Month Volume	45
	Envelop features: Ratio between Tot. incoming (C) and Tot. outgoing (D) within the same or across different perimeters normalized between 0 and 1 using the saturation function described in Section 5.1.	
Cash Ratio	Total Cash-In out of Total Cash-Out	1
All Ratio	Total incoming out of Total outgoing transactions (wire+cash)	1
Cash-In Wire-Out Ratio	Total Cash-In out of Total Wire-Out	1
Wire-In Cash-Out Ratio	Total Wire-In out of Total Cash-Out	1
	Temporal features	
Num. of active days (C, D, All)	Number of days where the account has had some activity	3
Num. of transactions per day (C, D, All)	Mean/median/std/min/max number of transactions made in the days where at least one transaction has been made	15
Timeslot 0/1 frequency (C, D, All)	Fraction of transactions that have been made in either timeslot 0 (6 AM - 8 PM) or timeslot 1 (8 PM - 6 AM). Fractions sum to 1.	6
	Sequential features	
X-Y sequence	Fraction of times transaction type Y followed transaction type X ($X, Y \in \{C, D\}$)	4
Median X-Y interval	Median time elapsed between transactions of type X and following transactions of type Y ($X, Y \in \{C, D\}$)	4
	Structuring features	
Structuring(C, D)	Binary features, 1 if the sum of the transactions with amount lower than τ_1 € sum to a total greater than or equal to τ_2 €. Actual thresholds omitted for safety reasons.	2
Structuring Std (C, D)	Standard deviation of the transactions with amount lower than τ_1 €.	2

$$f(C,D) = \begin{cases} \frac{1}{k} \cdot \left[\sigma\left(a \cdot \left(\frac{C}{D} - c_1\right)\right) - \sigma\left(a \cdot \left(\frac{C}{D} - c_2\right)\right)\right], \text{if } D > 0 \\ 0, \text{otherwise} \end{cases}$$

where $\sigma(x)$ is the sigmoid function; the constants c_1 and c_2 represent the minimum and maximum threshold ratios of interest. The normalization constant $k = \sigma(a \cdot c_1) - \sigma(-a \cdot c_2)$ ensures the function outputs values between 0 and 1. Here, we set $c_1 = 1 - c$ and $c_2 = 1 + c$, with $c = 0.2$ to indicate that ratios between 0.8 and 1.2 are of interest, whereas other values will be flattened to 0. The constant $a = 10$ determines the slope of the function.

After the computation, our dataset includes features spanning several orders of magnitude, which challenges distance-based algorithms. To address this, we apply a base-10 logarithm, i.e., given the features f, we consider the transformation $f = log_{10}(f + 1)$. We apply this transformation to all features, excluding the *Envelop*, *Timeslot 0/1 frequency*, *Number of active days*, and *Structuring* features where logarithmic scaling is unnecessary. For missing features, we fill values with zero to indicate absence (e.g., no debit transactions for *Total amount D*). For the standard deviation, we use -1 instead, signalling out-of-distribution values. At last, we then standardize all features by subtracting the mean and dividing by the standard deviation.

To reduce multicollinearity, we perform correlation-based feature selection [8]. We compute the correlation matrix and remove features with correlation above a threshold θ. Exploring θ values from 0.1 to 1, we identify elbow points near 0.5 and 0.8. We test these values, along with $\theta = 1$ (no reduction), as hyperparameters. Similarly, we test the usage of Principal Component Analysis (PCA) for dimensionality reduction.

5.2 Anomaly Detection

To identify anomalies in an unsupervised setting – where no prior knowledge about what defines or characterizes an outlier is available – we select distance-based (*Isolation Forest (IF)* [15], *One-class SVM (OCS)* [21]), density-based (*Local Outlier Factor (LOF)* [2]) and reconstruction-based (*Autoencoder (AE)* [9]) techniques. For OCS, given the large dataset, we use the Stochastic Gradient Descent linear One-Class SVM solver with *Nystroem* kernel approximation [24] to reduce computational complexity while preserving nonlinearity.

For each algorithm, we perform a grid search (as detailed in Table 4) to identify the optimal configuration, validated on a partial ground truth (as detailed in Section 6.1).

All algorithms generate an anomaly score for each sample. We then rank accounts based on their scores and consider the top-K most anomalous accounts for further analysis. The output of this step is a ranked list of accounts ordered by their anomaly score. The rationale for considering top-K accounts comes from the limited processing capacity of the competence centre.

Table 4: Key hyperparameters considered for the anomaly detection algorithms.

Algorithm	Key Hyperparameters
LOF	Number of neighbors (k), Distance metric, Correlation threshold, PCA
AE	Number of layers, Neurons per layer, Batch size, Epochs
IF	Number of trees, Subsample size, Correlation threshold, PCA
OCS	ν (fraction of training errors), Tolerance, Kernel type, Correlation threshold, PCA

Post-Processing Filter. Because of the unsupervised nature of the adopted algorithms, the anomalous behaviors identified are not biased toward illicit activities: many of the identified outliers are indeed accounts acting strangely, but not of interest to AML (mainly, because of the very low volumes of money involved). Because of the limited time available to human operators, prioritization of accounts with high cash movements is needed. To this end, we introduce a minimum threshold T_{post} on the total amount moved by the account. To reduce the workload on the operators, we filter the ranking of anomalous accounts by selecting those with a total moved amount above a threshold T_{post}. We optimise this threshold along with other pipeline hyperparameters. Note that we avoid pre-filtering the dataset to ensure the algorithm retains the necessary samples to define normal behaviours and detect anomalies effectively.

6 Experimental results

This section reports the main experimental results obtained for MAD. We present in detail all of the main numerical and qualitative results, as well as the main hyperparameters that have been the subject of an exploration. However, we cannot disclose the configurations of hyperparameters that have been identified as the best performing. The source code and the anonymized data are available for research purposes upon request.[2]

6.1 Experimental setup

In this section, we provide details on the definition of a ground truth based on the available data, as well as the predictions produced by the proposed approach. As a refresher, the data used for the validation spans across 8 months.

Ground truth definition. Being able to define the complete ground truth over the available data would require complete knowledge of all illicit activities performed. This is clearly outside the realm of feasibility, as it would require the perfect human evaluation of billions of transactions. In fact, only a subset of the full ground truth is known. These include the SARs that have been filed by Intesa Sanpaolo on cash-related activities, during the period of interest. Since a majority of the SARs are archived by the FIU (as they are not considered interesting), we limit the ground truth to the non-archived SARs. Additionally, operators from the competence centre assign a risk score to each SAR. This score allows us to identify the SAR subset as *high risk* ones. These contain behaviours that are considered more troubling – it is thus a high priority that TxM detects them.

Two sources of detections generate those SARs: TradTxM and Branches. TradTxM is the rule-based commercial system currently adopted in the bank (name changed to avoid the disclosure of internal information). It encodes suspicious scenarios as rules, with specific accumulators and thresholds. Each rule involves the check of a single criterion, i.e. in a single-criterion anomaly detection system. When any of the rules triggers, TradTxM generates a detection, which the competence centre's operators manually check. For the period of interest, TradTxM triggered about 3,000 alarms per month (18,000 in 6 months), of which only 58 are SARs within the perimeter of interest (and only 4 are *high risk*). The current system is thus very prone to false positives.

[2]Please refer to Intesa Sanpaolo, with a request to AFC Digital Hub (adh@pec.afcdigitalhub.com). Researchers interested in having access to data for academic purposes will be asked to sign a non-disclosure agreement.

Branches instead refers to suspicious situations flagged by branch managers, and undergo the same analysis by the competence centre. Branches filed a total of 7,164 SARs in the 6 months of interest, most of which have not been detected by TradTxM (698 *high risk* ones).

Both SARs originating from TradTxM and from Branches generally contain transactions spanning multiple months. For this reason, they cannot be easily assigned as belonging to a single month. Consequently, the validation ground truth spans the entire 8-month period, instead of having a per-month ground truth. In our experiments, we run MAD every month, and aggregate the results over the 8 months to compare against the ground truth. We describe how we do it in the following.

Predictions extraction. MAD produces a ranking of the top K most anomalous accounts, for each month. Based on the previously defined ground truth, we need to aggregate results for comparability. *First*, we extend the prediction to the 8-month period by selecting the union of the top K predictions for each of the 8 months. In other words, an account is considered as *predicted positive* if it has been predicted as anomalous for at least one of the 8 months. With this extension, both the ground truth and the predictions refer to the same period. *Second*, the ground truth is defined in terms of *SARs*, whereas MAD works at the *account* level. We consider an account as a *true positive* if it is predicted as anomalous by MAD and it belongs to at least one SAR in the period of interest. Similarly, for SAR-level evaluations, we consider an SAR as being detected if MAD predicts as anomalous at least one account belonging to the SAR. These assumptions do not constitute a limitation: human operators can unravel the rest of the activity (and accounts) after being provided an initial account-based detection.

Baseline, non-regression and progressions. The main baseline of MAD is the previously adopted system, TradTxM: It runs in the same conditions, over the same data. The SARs coming from branches are not a TxM system *per se* – in fact the branches operate using much more information than those used by MAD and TradTxM– and are, as such, not comparable. We refer to the capability of MAD to detect SARs also identified by TradTxM as *non-regression*, whereas any additional detection is considered as a *progression*. We will evaluate progressions based on (1) the Branches data, and (2) the a posteriori manual examination of some cases revealed by MAD which are not part of the ground truth.

Evaluation metrics. We validate the results both in terms of detected SARs, as well as detected accounts. For the SARs, we consider as the two metrics of interest the total number of *undetected* SARs, and the total amount of money involved in those SARs. The second metric provides an estimate of the volume of the missed detections. For the accounts, we define *precision*, *recall* and F_1 *score* based on the previous definitions of predictions and ground truth.

6.2 Results

We present, in the following, the main results of our work.

Overall results. Table 5 represents the main result of the various Anomaly Detection algorithms used in MAD, along with the performance of TradTxM. These results are reported for $K = 550$. This value of K is the average number of monthly detections reported by TradTxM, and defines a reasonable number that can be processed (manually) by the competence centre.

Local Outlier Factor achieves the best results across almost all metrics. By definition, TradTxM has 0 undetected TradTxM SARs, but performs poorly across all other metrics. The Autoencoder performance is comparable with LOF, whereas the Isolation Forest and the One-class SVM yield the worst performance. We focus the main considerations of this section on LOF. Similar conclusions can be drawn for the Autoencoder. This overview already provides details about both the non-regression and progression properties. Although full non-regression is not achieved by any of the algorithms (i.e., none of the algorithms achieves 0% undetected TradTxM SARs), both LOF and AE only miss 1 out of 4 high risk SARs. In addition, MAD improves upon TradTxM on Branches SARs. TradTxM did not detect any of those cases. MAD instead identifies 200+ cases, for a total volume of 150M €.

Undetected SARs as a function of K. The limitation to the top 550 monthly detections is introduced to make a fair comparison against the existing system. However, a larger number of detections may be handled on a monthly basis by the operators. We study the change in undetected SARs as a function of K in Figure 2. The results confirm that the LOF produces the most satisfactory results across values of K. Interestingly, for TradTxM SARs, the Autoencoder detects a similar number of true positives. However, the gap between the *amount* curves highlights how the LOF's detections identifies SARs with larger volumes of money involved.

Results on TradTxM + Branches. We previously considered the performance on TradTxM and Branches separately. Although this separation is necessary to distinguish between non-regression and progression, we are also interested in the overall performance, without distinctions between the origin of the SARs.

We analyse, once again, model performance as a function of K. Figure 3 reports the performance of the four models and TradTxM, in terms of precision, recall and F1 score. The overall precision and recall are as expected: precision decreases as K increases (the model is less certain about anomalies further down the ranking), whereas recall increases with K (predicting more anomalous accounts increases the fraction of correct positive predictions).

The performance of TradTxM is also reported. As mentioned, TradTxM operates at an average of 550 detections per month. The gray vertical line defines the operating point, providing a fair comparison (i.e., the results in Table 5). However, we note that even for lower values of K, MAD produces generally better results. This implies that, even with less involvement of the competence centre, better overall performance can be achieved. Conversely, the same effort can be placed to identify a larger quantity of cases. We additionally note that the F1 score, which provides a trade-off between precision and recall, has a sharp increase for lower values of K, dictated by the sharp growth in precision. For larger values of K, the performance starts to plateau. Given the constraints in terms of human effort required to process the detections, we find that $K = 550$ is indeed a reasonable trade-off.

Manual inspection. The previous results are all focused on the "already known" cases. An interesting question naturally arises: can MAD also detect suspicious activities that where not previously known by the bank? To answer this question, domain experts from Intesa Sanpaolo manually inspected a subset of detections identified by MAD, that were not a part of the ground truth (neither TradTxM nor Branches). Given the manual effort required for this task, only

Table 5: Performance of Best-setting algorithms at K = 550 and TradTxM performance on Cash perimeter. Count (hr) represents the number of undetected high risk SARs.

Algorithm	Undetected TradTxM SARs (↓)			Undetected Branches SARs (↓)			Precision (↑)	Recall (↑)	F1 Score (↑)
	Frac. (%)	Frac. *hr* (%)	Amount (M €)	Frac. (%)	Frac. *hr* (%)	Amount (M €)			
LOF	44.83	25.00	13.22	**96.80**	**96.85**	**4025.91**	**0.0889**	**0.0543**	**0.0674**
AE	50.00	25.00	16.86	97.78	97.28	4065.93	0.0694	0.0424	0.0526
IF	91.38	100.00	26.89	98.23	98.57	4100.28	0.0524	0.032	0.0397
OCS	82.76	50.00	24.28	98.31	97.71	4090.18	0.049	0.0299	0.0371
TradTxM	**0.00**	**0.00**	**0.00**	100.00	100.00	4178.90	0.0132	0.008	0.01

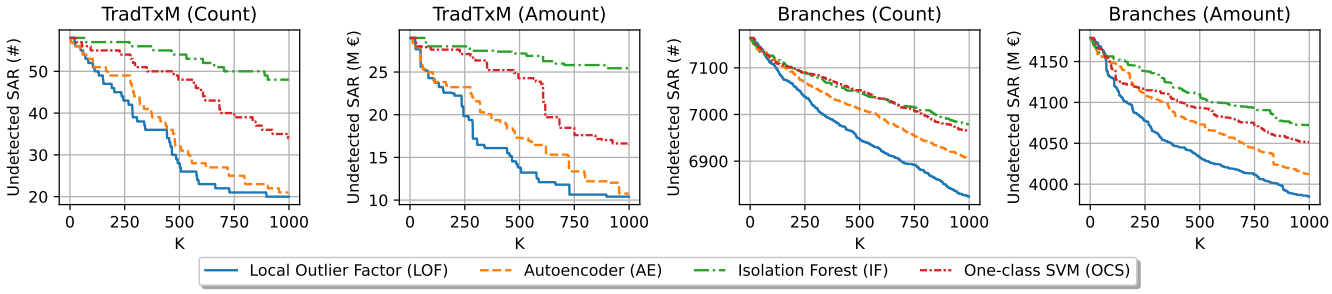

Figure 2: Number of undetected SARs and their amount, for TradTxM and Branches, as a function of K.

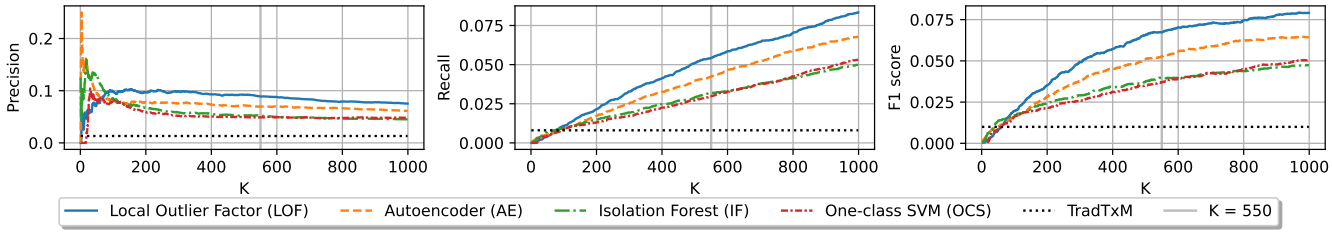

Figure 3: Precision, recall and F1 score for the various anomaly detection algorithms, as well as TradTxM.

LOF and AE results have been considered, given their promising results. For each of these two algorithms, the top 50 accounts not in the ground truth have been manually inspected, in a way that is comparable to the one carried out by the competence centre. For LOF, 13 out of the 50 accounts (26%) have been considered worth of further inspection. For the AE, the number raises to 15 of the top 50 (30%). We know from previous domain expertise that only approximately 5.7% of the cases considered interesting by the competence centre actually result in a non-archived SARs. By applying this 5.7% factor, LOF and AE have a 1.5% and 1.7% precision respectively, on cases that *were previously completely unknown*. The remaining false positives (37 and 35 accounts respectively) were considered by the domain experts as actual outlier behaviors, although not related to money laundering. Examples of these behaviors are: accounts engaging in multiple types of personal and commercial activities, gifting large lumps of money (e.g., weddings), or potential payments for off-the-books works.

7 Deployment

MAD has been deployed in November 2024 on an on-premise node within Intesa Sanpaolo's cluster. We report in this section the main deployment details.

Hardware and software. The node is characterized by 112 cores (Intel Xeon Gold 6330N CPU 2.20GHz), 2.5 TB of memory, 30 TB of storage, and 1 NVIDIA A100 GPU. The resources dedicated to MAD are 8 cores and 40 GB of memory. The solution has been implemented in Python (3.11), using FastAPI for the definition of API, BigQuery for accessing the data, and scikit-learn + PyTorch for the implementation of the full MAD pipeline. For the period November 2024 to February 2025, the average execution time for the entire pipeline has been, on average, of 8.7 hours, to process an average of 329M monthly transactions.

Post-deployment performance. LOF has clearly shown the best performance throughout the validation session. As such, a subset of 50 anomalies identified by LOF are being sent to the competence centre on a monthly basis, for a total of 200 detections: of these, 68 have already been processed, at least by the first level (L1). L1 has

promoted 36 out of the 68 detections (52.9%), whereas TradTxM only achieves a promotion rate of 12.8%. Out of those 36 cases, 4 belong to already reported SARs. L2 has processed 23 of the remaining detections, promoting 9 of them, for a promotion rate of 39.1%, whereas TradTxM achieves 9.4%. By combining the L1 and L2 promotion rates, MAD has an overall rate of 20.7%, which is an order of magnitude above TradTxM, which performs at a rate of approximately 1.2%. These results are based on a small set of detections and will therefore change in the future, but the current results appear to indicate a meaningful improvement in performance over traditional TXM techniques.

The goal for the upcoming months (at the time of writing) is to transition from TradTxM to MAD as the main TxM system in Intesa Sanpaolo for cash transaction monitoring.

8 Lessons learned

Designing, developing and deploying MAD in production has produced a variety of insights that, we believe, can be beneficial for other researchers, practitioners and stakeholders working in similar fields. We highlight the main ones in the following.

There are many low-hanging fruits. The current rule-based TxM systems only highlight a *very* small fraction of suspicious cases. The simple nature of those algorithms is an intrinsic limitation. The adoption of simple *unsupervised* anomaly detection techniques already led to significant improvements in terms of quality of results. Rule-based TxM, however, is commonly adopted in many institutions. Moving to ML techniques could be very beneficial.

Coping with real problems is a game changer. Many works in literature that address AI-based anti-money laundering present contexts that are significantly simpler than reality. For instance, the problems are often framed as balanced (or almost balanced), with similar quantities of suspicious and non-suspicious behaviours. Also, a limited number of entities is involved, in terms of customers, accounts, or transactions. In some cases, supervised problems are addressed, assuming that ground truths are exhaustively available. Finally, most of the works are validated on synthetic data, which does not reflect the many complex patterns adopted by humans. In this work we show what a real-case scenario looks like, with all of its limitations and potentials.

The importance of pre- and post-processing. We did not report hyperparameter tuning results to avoid publishing sensitive information. However, we interestingly discovered that LOF was the worst performing algorithm when no threshold T_{post} was used. The anomalies found by it were indeed outliers, but "not interesting". By applying a post-processing threshold, LOF quickly raised to be the best-performing anomaly detection approach, by some margin.

The "unknowns" were not too scary. The manual inspection phase identified suspicious cases that were not found by TradTxM, nor Branches. It is, of course, problematic that those cases were not identified elsewhere. However, while the accounts were unknown, no previously unknown money laundering *behaviours* emerged from those cases.

The results presented in this paper have been considered very important by our bank partners. However, stronger improvement can be provided by a continuing research effort on outlier detection.

Acknowledgments

This research was conducted under a cooperative agreement between Polytechnic University of Turin and AFC Digital Hub (Anti Financial Crime Digital Hub). AFC Digital Hub is a Turin-based consortium to fight digital financial crime through the use of new technologies and artificial intelligence, with the aim of becoming a national and international point of reference in the fight against money laundering and terrorism. AFC Digital Hub's members are Intesa Sanpaolo, Intesa Sanpaolo Innovation Center, the Polytechnic University of Turin, the University of Turin and CENTAI.

The authors would like to thank Aurora Costantino, Lorenzo Conti, Marco Fornasiero, Erica Lanzarini, Emma Lombardi, Maria Oliva, Carlo Prada, Paolo Racca, Valeria Ricci, Eliana Salvemini and Mauro Valorio for their useful comments. The authors would like to thank Luca Cattarossi, Valerio Cencig, Andrea Cosentini, Mario D'Almo e Laura Li Puma for supporting the researcher team.

References

[1] Mohannad Alkhalili, Mahmoud H. Qutqut, and Fadi Almasalha. 2021. Investigation of Applying Machine Learning for Watch-List Filtering in Anti-Money Laundering. *IEEE Access* 9 (2021), 18481–18496. doi:10.1109/ACCESS.2021.3052313 Conference Name: IEEE Access.

[2] Markus M Breunig, Hans-Peter Kriegel, Raymond T Ng, and Jörg Sander. 2000. LOF: identifying density-based local outliers. In *Proceedings of the 2000 ACM SIGMOD international conference on Management of data*. 93–104.

[3] Ramiro Daniel Camino, Radu State, Leandro Montero, and Petko Valtchev. 2017. Finding Suspicious Activities in Financial Transactions and Distributed Ledgers. In *2017 IEEE International Conference on Data Mining Workshops (ICDMW)*. IEEE, New Orleans, LA, 787–796. doi:10.1109/ICDMW.2017.109

[4] Xiangrui Chao, Gang Kou, Yi Peng, and Fawaz E. Alsaadi. 2019. Behavior monitoring methods for trade-based money laundering integrating macro and micro prudential regulation: a case from China. *Technological and Economic Development of Economy* 25, 6 (May 2019), 1081–1096. doi:10.3846/tede.2019.9383 Number: 6.

[5] Dawei Cheng, Yujia Ye, Sheng Xiang, Zhenwei Ma, Ying Zhang, and Changjun Jiang. 2023. Anti-Money Laundering by Group-Aware Deep Graph Learning. *IEEE Transactions on Knowledge and Data Engineering* 35, 12 (Dec. 2023), 12444–12457. doi:10.1109/TKDE.2023.3272396 Conference Name: IEEE Transactions on Knowledge and Data Engineering.

[6] Roxane Desrousseaux, Gilles Bernard, and Jean-Jacques Mariage. 2021. Profiling Money Laundering with Neural Networks: a Case Study on Environmental Crime Detection. In *2021 IEEE 33rd International Conference on Tools with Artificial Intelligence (ICTAI)*. 364–369. doi:10.1109/ICTAI52525.2021.00059 ISSN: 2375-0197.

[7] Bogdan Dumitrescu, Andra Băltoiu, and Ştefania Budulan. 2022. Anomaly Detection in Graphs of Bank Transactions for Anti Money Laundering Applications. *IEEE Access* 10 (2022), 47699–47714. doi:10.1109/ACCESS.2022.3170467 Conference Name: IEEE Access.

[8] Flavio Giobergia, Elena Baralis, Maria Camuglia, Tania Cerquitelli, Marco Mellia, Alessandra Neri, Davide Tricarico, and Alessia Tuninetti. 2018. Mining sensor data for predictive maintenance in the automotive industry. In *2018 IEEE 5th International Conference on Data Science and Advanced Analytics (DSAA)*. IEEE, 351–360.

[9] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. 2016. *Deep Learning*. MIT Press. http://www.deeplearningbook.org.

[10] Jorge Guevara, Olmer Garcia-Bedoya, and Oscar Granados. 2020. Machine learning methodologies against money laundering in non-banking correspondents. In *Applied Informatics: Third International Conference, ICAI 2020, Ota, Nigeria, October 29–31, 2020, Proceedings 3*. Springer, 72–88.

[11] Rasmus Ingemann Tuffveson Jensen and Alexandros Iosifidis. 2023. Qualifying and raising anti-money laundering alarms with deep learning. *Expert Systems with Applications* 214 (March 2023), 119037. doi:10.1016/j.eswa.2022.119037

[12] Martin Jullum, Anders Løland, Ragnar Bang Huseby, Geir Ånonsen, and Johannes Lorentzen. 2020. Detecting money laundering transactions with machine learning. *Journal of Money Laundering Control* 23, 1 (Jan. 2020), 173–186. doi:10.1108/JMLC-07-2019-0055 Publisher: Emerald Publishing Limited.

[13] Thomas N Kipf and Max Welling. 2016. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907* (2016).

[14] Danilo Labanca, Luca Primerano, Marcus Markland-Montgomery, Mario Polino, Michele Carminati, and Stefano Zanero. 2022. Amaretto: An Active Learning Framework for Money Laundering Detection. *IEEE Access* 10 (2022), 41720–41739.

doi:10.1109/ACCESS.2022.3167699

[15] Fei Tony Liu, Kai Ming Ting, and Zhi-Hua Zhou. 2008. Isolation forest. In *2008 eighth ieee international conference on data mining*. IEEE, 413–422.

[16] Jiayi Liu, Changchun Yin, Hao Wang, Xiaofei Wu, Dongwan Lan, Lu Zhou, and Chunpeng Ge. 2023. Graph Embedding-Based Money Laundering Detection for Ethereum. *Electronics* 12, 14 (Jan. 2023), 3180. doi:10.3390/electronics12143180 Number: 14 Publisher: Multidisciplinary Digital Publishing Institute.

[17] Devendra Kumar Luna, Girish Keshav Palshikar, Manoj Apte, and Arnab Bhattacharya. 2018. Finding shell company accounts using anomaly detection. In *Proceedings of the ACM India Joint International Conference on Data Science and Management of Data*. ACM, Goa India, 167–174. doi:10.1145/3152494.3152519

[18] Jack Nicholls, Aditya Kuppa, and Nhien-An Le-Khac. 2021. Financial cybercrime: A comprehensive survey of deep learning approaches to tackle the evolving financial crime landscape. *Ieee Access* 9 (2021), 163965–163986.

[19] Alexander Ratner, Stephen H Bach, Henry Ehrenberg, Jason Fries, Sen Wu, and Christopher Ré. 2017. Snorkel: Rapid training data creation with weak supervision. In *Proceedings of the VLDB endowment. International conference on very large data bases*, Vol. 11. NIH Public Access, 269.

[20] Zeinab Rouhollahi, Amin Beheshti, Salman Mousaeirad, and Srinivasa Reddy Goluguri. 2022. Towards Proactive Financial Crime and Fraud Detectionthrough Artificial Intelligence and RegTech Technologies. In *The 23rd International Conference on Information Integration and Web Intelligence (iiWAS2021)*. Association for Computing Machinery, New York, NY, USA, 538–546. doi:10.1145/3487664.3487740

[21] Bernhard Schölkopf, John C Platt, John Shawe-Taylor, Alex J Smola, and Robert C Williamson. 2001. Estimating the support of a high-dimensional distribution. *Neural computation* 13, 7 (2001), 1443–1471.

[22] Andrea Tundis, Soujanya Nemalikanti, and Max Mühlhäuser. 2021. Fighting organized crime by automatically detecting money laundering-related financial transactions. In *Proceedings of the 16th International Conference on Availability, Reliability and Security (ARES '21)*. Association for Computing Machinery, New York, NY, USA, 1–10. doi:10.1145/3465481.3469196

[23] Atif Usman, Nasir Naveed, and Saima Munawar. 2023. Intelligent Anti-Money Laundering Fraud Control Using Graph-Based Machine Learning Model for the Financial Domain. *JCIT* 25, 1 (Jan. 2023), 1–20. doi:10.4018/JCIT.316665 Publisher: IGI Global.

[24] Christopher Williams and Matthias Seeger. 2000. Using the Nyström Method to Speed Up Kernel Machines. In *Advances in Neural Information Processing Systems*, T. Leen, T. Dietterich, and V. Tresp (Eds.), Vol. 13. MIT Press. https://proceedings.neurips.cc/paper_files/paper/2000/file/19de10adbaa1b2ee13f77f679fa1483a-Paper.pdf

[25] Yimin Yang and Min Wu. 2020. Supervised and Unsupervised Learning for Fraud and Money Laundering Detection using Behavior Measuring Distance. In *2020 IEEE 18th International Conference on Industrial Informatics (INDIN)*, Vol. 1. 446–451. doi:10.1109/INDIN45582.2020.9442099 ISSN: 2378-363X.

[26] Yan Zhang and Peter Trubey. 2019. Machine Learning and Sampling Scheme: An Empirical Study of Money Laundering Detection. *Comput Econ* 54, 3 (Oct. 2019), 1043–1063. doi:10.1007/s10614-018-9864-z