IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received March 27, 2022, accepted April 8, 2022, date of publication April 18, 2022, date of current version April 25, 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3167699

Amaretto: An Active Learning Framework for Money Laundering Detection

DANILO LABANCA[1], LUCA PRIMERANO[2], MARCUS MARKLAND-MONTGOMERY[2], MARIO POLINO[1], MICHELE CARMINATI[1], AND STEFANO ZANERO[1], (Senior Member, IEEE)
[1]Dipartimento di Elettronica, Informazione e Bioingegneria, Politecnico di Milano, 20133 Milan, Italy
[2]Napier Technologies Ltd., London EC2R 8EJ, U.K.

Corresponding author: Michele Carminati (michele.carminati@polimi.it)

This work was supported by Napier Technologies Ltd.

ABSTRACT Monitoring financial transactions is a critical Anti-Money Laundering (AML) obligation for financial institutions. In recent years, machine learning-based transaction monitoring systems have successfully complemented traditional rule-based systems to reduce the high number of false positives and the effort needed to review all the alerts manually. Unfortunately, machine learning-based solutions also have disadvantages: while unsupervised models can detect novel anomalous patterns, they are usually characterized by a high number of false alarms; supervised models, instead, usually offers a higher detection rate but require a large amount of labeled data to achieve such performance. In this paper, we present Amaretto, an active learning framework for money laundering detection that combines unsupervised and supervised learning techniques to support the transaction monitoring processes by improving the detection performance and reducing the compliance management costs. Amaretto exploits novel selection strategies to target a subset of transactions for investigation, making more efficient use of the feedback provided by the analyst. We perform the experimental evaluation on a synthetic dataset provided by the industrial partner, which simulates the profiles of clients trading in international capital markets. We show that Amaretto outperforms state-of-the-art solutions by reducing money laundering risk and improving detection performance. In particular, we compare state-of-the-art unsupervised and supervised techniques commonly used in the AML domain with the ones implemented in this work. We show that the Isolation and Random Forests of Amaretto perform best in the task under analysis, with an AUROC of 0.9 for the first (20% better on average) and a detection rate of 0.793 for the second (30% better on average). In addition, they are characterized by lower investigation costs computed in terms of the daily number of transactions to be examined and the number of false positives and false negatives. Finally, we compare Amaretto against a state-of-the-art active learning fraud detection system, achieving better detection performances and lower costs in all the analyzed scenarios. Worth mentioning, Amaretto improves the detection rate up to 50% and reduces the overall cost by 20% in the most realistic scenario under analysis.

INDEX TERMS Active learning, anti-money laundering, financial systems, supervised learning, unsupervised learning, selection strategies.

I. INTRODUCTION

Money laundering encompasses any process by which the income of unlawful activities (e.g., drug trafficking, illegal arms trafficking, tax evasion) is introduced into the financial system through multiple operations that conceal their illicit origins. Nowadays, money laundering affects all worldwide economies and is responsible for generating illegal financial flows between \$1.6-2.85 trillion per year, equivalent to 2.1%-4% of the Gross World Product [1]. Transaction monitoring in AML consists of a set of activities carried out by analysts and automated systems to scrutinize customers' transactions. The aim is to detect suspicious behaviors linked to money laundering. Financial transactions include bank transfers, credit card payments, or investment banking transactions such as equity and derivative trades.

An expert system is often the first step to implement AML procedures by deploying rules that are configured

The associate editor coordinating the review of this manuscript and approving it for publication was Khin Wee Lai.

to monitor pre-determined unusual behaviors. The expert system generates alerts if rules are triggered (e.g., if *amount* $>$ 10,000,000 then *raise alert*). The benefits of a heuristic-based approach are the ease in interpreting the output of the system and the ability for subject matter experts (i.e., analysts working in the AML domain) to use that information easily. The disadvantage of expert systems is that money laundering techniques and financial crime are always evolving, so the rules need to be updated to ensure they are fit to capture these changes. Moreover, rules can only cover known anomalous behaviors, and they cannot detect unknown unusual behaviors resulting in false negatives. The fact that rules have to be configured using specific static thresholds results in a high number of false positives and, subsequently, an increase in the volume of manual investigations.

Machine learning enhances these AML techniques by overcoming some of the pitfalls in rule-based systems. Machine learning models can extract and analyze patterns and insights from data and assess unusual correlations unknown to subject matter experts. Supervised machine learning models can classify transactions as normal or anomalous. However, they require a large sample of manually reviewed transactions (labels). In order to collect a valuable set of labeled data as quickly and as efficiently as possible, these modern systems can leverage active learning. *Active learning* is a technique that uses machine learning models to select transactions for an investigation that have the highest probability of improving the performance of the supervised machine learning system.

In this paper, we present Amaretto, an active learning system that combines unsupervised and supervised models organized in an ''analyst-in-the-loop'' framework. The unsupervised model allows the system to detect unknown anomalies and new patterns that have not been seen before, while the supervised model can use labels previously classified by subject matter experts to improve the detection rate. The system preprocesses the raw transactional data, converting it to aggregated vectors; the aggregation is performed because money laundering patterns usually comprise multiple transactions executed within a period of time. The unsupervised and supervised models take the vectors as input and compute an anomaly score for each one. Subsequently, a selection strategy is applied to choose the samples that the analyst will review. Finally, the labels collected from the analyst are used as training data for the supervised model that will compute the final risk score for the aggregated vectors.

We perform the experimental evaluation on a synthetic dataset provided by the industry partners we collaborated with. The dataset includes both normal transactions and anomalous patterns that may be linked to potential money laundering activities. The dataset is modeled based on real-world investment banking scenarios. On this data, we compare state-of-art unsupervised models, and we demonstrate that Isolation Forest is the best performing for AML tasks under analysis. Then, we conduct a similar assessment amongst supervised techniques concluding that Random Forest outperforms the others. Subsequently, we prove the contributions made by the unsupervised model to complement the ability of supervised models by detecting new types of anomalies. Finally, we confirm the robustness of our design in a real-world scenario, identifying the best selection strategies from the ones proposed and showing that Amaretto outperforms state-of-the-art solutions by improving the detection rate and reducing the compliance management costs for financial institutions. Amaretto improves the True Positive Rate (TPR) up to 50% and reduces the overall cost by 20% in the most realistic scenario under analysis. It is important to highlight that, to provide a meaningful comparison between Amaretto and the other approaches under analysis, we perform the entire experimental evaluation on the same dataset. In addition, to allow the reproducibility of the results, we released the synthetic dataset of transactions at https://github.com/necst/amaretto_dataset.

In summary, the contributions are the following:

- Amaretto, an active learning system that combines unsupervised and supervised models organized in an ''analyst-in-the-loop'' framework.
- A novel selection strategy for an active learning framework that detects potential money laundering patterns. This strategy considers event diversity and prioritizes new anomalous patterns to improve the quality of the knowledge base and training set.
- The experimental evidence that demonstrates the importance of an active learning framework to achieve better detection performance and to reduce the cost of monitoring transactions for financial institutions. This analysis includes the comparison of supervised and unsupervised algorithms for detecting potential money laundering, including detailed benchmarking.

The remainder of this paper is structured as follows: In Section II, we provide some background concepts related to the money laundering detection problem, alongside an analysis of the main challenges, fundamental for understanding the choices we made in Amaretto. In Section III, we present some of the most relevant works related to money laundering and fraud detection, highlighting the main differences with Amaretto. In Section IV we describe the synthetic dataset at our disposal and the classes of anomalies considered in this work. In Section V, we provide a detailed description of Amaretto, its main components, as well as the design choices that were made to build an end-to-end active learning framework. Then, in Section VI, we show the experimental evaluation of our framework. Finally, in Section VII and VIII, we discuss the limitations, the future works, and the conclusions of our work.

II. BACKGROUND AND CHALLENGES

Detecting money laundering can be considered one of the most challenging tasks within anomaly detection. First of all, there is no common worldwide regulation that sets standards for which transactions are suspicious with respect to money laundering activities. Furthermore, money laundering processes involve multiple transactions between different counterparties using diverse monetary instruments; therefore, traditional anomaly detection techniques analyzing transactions in isolation may be ineffective. This paper focuses on detecting unusual transactional activities that may be linked to money laundering occurring in capital markets. Unusual transactions and customer behaviors are considered outliers associated with a money laundering risk. In [2], an outlier is defined as *"an observation (or subset of observations) which appears to be inconsistent with the remainder of that set of data"*. In this paper, an anomalous behavior is considered high risk for money laundering and, therefore, should be investigated. This is similar to fraud; however, money laundering often involves multiple transactions across multiple accounts, while fraud mainly occurs at a transactional level. In an ideal world with unlimited resources, an analyst would look at every transaction and then decide which one is worth investigating further as it may be linked to money laundering. Considering the large volume of transactions executed daily in global markets, this approach is not feasible because financial institutions, regulators, and enforcers have a limited amount of subject matter experts to deal with such a demand. To solve this problem, organizations employ a risk-based approach by adopting automated systems to flag and allocate transactions for review. The objective is to maximize the time spent investigating suspicious activities with a high risk of money laundering.

One of the key challenges in researching novel approaches to detect money laundering is the lack of standardized and available datasets. Financial institutions rarely share data due to confidentiality reasons and specific regulations. As part of this research, we leveraged a dataset generated by our industry partners; they specialize in AML and work with financial institutions to help them comply with AML regulations. The dataset implements different suspicious patterns similar to those defined by the Financial Action Task Force (FATF) on money laundering [3].

III. RELATED WORKS

In the last years, several systematic review papers have been published, which describe machine learning applied to the fraud and money laundering detection domains [4]–[12]. From a high level point of view, current approaches can be divided into *unsupervised*, *supervised*, and *active learning* techniques. In addition, these works describe each method's strengths and weaknesses, highlighting the need for combining them and providing further motivation to the work presented in this paper. While supervised solutions have high performance in detecting known frauds, they cannot find new fraudulent patterns and have a high rate of false positives; therefore, unsupervised techniques are needed to detect novel money laundering patterns.

A. UNSUPERVISED LEARNING

Unsupervised learning is mainly used to detect unusual correlations, and it is applied where it is expensive to obtain labels (i.e., multiple analysts need to review a significant number of data points). The main principle in money laundering detection is to quantify how a transaction (or group of transactions) deviates from the norm.

In the fraud detection domain, Ramaswamy *et al.* [13] propose a formulation for outliers in terms of the distance of a point from its neighbors.

Williams *et al.* [14] prove that a Replicator Neural Network can detect anomalies in very diverse datasets, and in some cases, it overcame issues commonly affecting Neural Networks such as training with a small dataset.

BankSealer [15], [16] works in an unsupervised setting, extracting local, global, and temporal profiles [17] for each user to capture their behaviors. The same authors also study the security of fraud detection systems against mimicry and adversarial attacks [18], [19].

In the AML domain, a recent research trend has demonstrated the effectiveness of the application of Isolation Forests and Support Vector Machines to the detection of money laundering patterns [20], [21].

Le-Khack and Kechadi [22], [23] focus on detecting anomalies in investment funds; they suggest an approach based on clustering profiles into categories and feeding a Backpropagation Neural Network with the transformed data to output an anomaly score for each transaction. This approach is specific to the problem and dataset: the entire learning process is based on two high-level features derived from the raw data. This seems to offer a limited perspective on the complexity of the underlying behaviors.

Torres and Ladeira [24] propose a hybrid approach composed of unsupervised outlier detection algorithms and the use of Visual Analytics methods to support the real-time human analysis to reduce the incidence of false positives. Similarly to Amaretto, the proposed approach targets the problem of improving the analysis of the vast daily volume of financial transactions. However, due to the exploitation of Visual Analytics techniques, the presented approach impacts human analysts' processing time, possibly increasing the investigation costs.

Paula *et al.* [25] address the problem of money laundering in Brazilian exports using Deep Learning Autoencoder demonstrating its effectiveness against PCA-based methods.

The disadvantage of unsupervised models is that in practice, an analyst will still have to verify whether all the predictions were correct, and an unsupervised model will not be able to fully leverage the output of the reviews as part of subsequent runs. Also, unsupervised techniques tend to generate a large number of false positives due to unusual data correlations that are perfectly acceptable [26]. This is an issue for institutions since reviewed false positives translate into a direct cost for the organization. Moreover, the lack of focus due to the high number of alerts could lead to potential money laundering cases not being reviewed promptly or missed.

B. SUPERVISED LEARNING

Supervised learning is used when labels are available. The main principle in money laundering detection is to quantify how a transaction (or group of transactions) is similar to known fraudulent patterns.

In the fraud detection domain, Batthacharyya *et al.* [27] demonstrate that in a real credit card fraud scenario, a Random Forest model outperforms Support Vector Machines and Linear Regression across all metrics used for the comparison. One of the first AML specific studies focused on rule-based methodologies (Decision Trees). This approach was used to create automated systems such as Financial crime law enforcement network AI System (FAIS) [28]. This system allows the analyst to follow evidence left by linked transactions and computes an anomaly score for each transaction. Simple Bayesian networks are used to update and combine evidence that a transaction or activity is illicit.

In the AML domain, the most common used techniques are Random Forests, Support Vector Machines, Decision Trees, deep neural networks, and rule-based systems [9]–[11], [29]. In the last year, also gradient bosting techniques have been successfully exploited [29]–[33].

Jullum *et al.* [30] detect money laundering at a transactional level using XGBoost and demonstrate its effectiveness against traditional rule-based systems. Alkhalili *et al.* [34] propose a watch-list filtering component applied to ML methods (i.e., Support Vector Machines, Decision Trees, and Naive Bayes) to reduce the number of false positives and to minimize analyst effort. They demonstrate that the SVM outperforms other algorithms. Similarly to Amaretto, both works [30], [34] highlight the importance of a selection strategy that takes into consideration non-reported alerts/cases. However, their works focus only on supervised learning techniques.

Tertychny *et al.* [31] address the scalability and the imbalance-resistance problem of the AML detection domain by proposing a two-layered ensembled modeling approach composed of a Logistic Regression model and gradient boosting techniques. They validate the approach using a real dataset of customer profiles and transaction histories, together with labels provided by AML experts.

Farrugia *et al.* [32] extract features from the historical transaction data of accounts marked as illegal activities by the Ethereum community and regular accounts on the Ethereum platform. The authors use XGBoost to build a classification model to detect illegal accounts.

Vassallo *et al.* [33] propose an adaptation of the XGBoost algorithm and present a comparative analysis of various offline decision tree-based ensembles. They demonstrate that decision tree-based gradient boosting algorithms outperform state-of-the-art Random Forest results at both account and transaction levels. In this work, as presented in Section VI, we compare gradient boosting and random forest techniques too, which achieve comparable performance. However, we decided to use the Random Forest algorithm for Amaretto supervised module due to its lower cost in terms of false positives and negatives.

Supervised learning directly exploits manually reviewed transactions (i.e., labeled data) and generally outperforms unsupervised learning in anomaly detection and classification tasks [35]. However, a large amount of labeled data is required to achieve adequate performance. Additionally, supervised learning is not as effective at detecting new anomalous patterns (resulting in false negatives) compared to unsupervised learning. This is where active learning plays an important role in bridging unsupervised and supervised anomaly detection.

C. ACTIVE LEARNING

Amaretto implements an active learning system combining both supervised and unsupervised learning, leveraging their strengths and mitigating their weaknesses. Active learning is a process whereby a traditional anomaly detection system is enhanced with a model that queries a subject matter expert to label a transaction or group of transactions (suspicious or genuine). This model is used to select which transactions the subject matter expert should investigate to minimize manual data reviews and, at the same time, ensure the output of the overall anomaly detection system is improved. In [36], the authors exploit analyst feedback to self-tune and improve BankSealer's detection performance using a multi-objective genetic algorithm. In [37], the authors propose an ensemble of unsupervised methods, including a Density-based model, a Matrix Decomposition-based model, and a Replicator Neural Network. By combining the anomaly scores computed by the three models, their system ranks the instances based on the most anomalous ones and then presents them to the subject matter expert for review; the feedback collected is used to train a Random Forest model. Further to this research, in [38], the authors point out the importance of selecting different types of anomalies to enhance active learning frameworks (i.e., selecting different classes of anomalies).

D. DISCUSSION

With respect to the presented works, Amaretto explicitly focuses on reducing the cost and risk for a financial institution; the cost is due to the resources involved in manually reviewing multiple transactions, whilst the risk is linked to not detecting illicit activities. To do so, we directly exploit the main insights and results of the presented research works, evaluating them in terms of the investigation costs and not only from the detection performance point of view. Amaretto also optimizes the selection strategy (i.e., the strategy used to select the transactions for the subject matter expert to investigate) in order to spot novel anomalous patterns and improve the detection rate. This strategy prioritizes which transactions should be further investigated by AML investigators by considering the ''diversity'' of the output produced by the unsupervised module.

Another approach commonly applied in fraud and money laundering detection is the analysis of graphs, which is

TABLE 1. Capital market dataset details: Number of transactions (total (T), legitimate (L), and anomalous (A)), the ratio of anomalies over the total number of transactions, number of attributes, and number of transaction originators. The dataset is highly unbalanced, with only a small portion of transactions being anomalous.

Transactions				Attributes	Originators
Total (T)	Legitimate (L)	Anomalous (A)	Ratio (A/T)		
29,704,090	29,622,822	81,268	0.27 %	12	400

out of scope for Amaretto. For instance, Alarab *et al.* [39] present a novel approach based on Graph Convolutional Network combined with MultiLayer Perceptron to predict illicit transactions in the Bitcoin transaction graph. We refer the reader to [40]–[42] for a review of the research regarding graph-based anomaly detection methods in fraud detection, intrusion detection, telecommunication, and opinion networks.

IV. CAPITAL MARKET DATASET ANALYSIS

In the AML domain, one of the major limitations is the difficulty to obtain a real dataset from financial institutions due to privacy concerns. Besides, it is even more complicated to get a labeled dataset. Therefore, we make use of a synthetic dataset generated by our industry partner using a custom-built data generator, which simulates transaction profiles of clients transacting in international capital markets. We use the same synthetic dataset for all the experiments presented in Section VI in order to provide a meaningful evaluation of the performance of Amaretto and the other methods under analysis. In addition, to allow the reproducibility of the results and a fair comparison with future works, we released our synthetic dataset of transactions at https://github.com/necst/amaretto_dataset.

The data combines more than 10,000 parameters extrapolated from real market data. The dataset consists of 29,704,090 transactions executed by 400 end clients buying and selling specific securities in a specific market. Circa 90% of the users made at least 50,000 transactions while 10% of the users performed circa 400,000, which means that 10% of the clients executed almost 50% of the transactions. 98.43% of the transactions have an amount less than 1M USD, and 40% of them have an amount less than 10K USD. Data covers 60 days divided into 12 weeks (a week is composed of 5 days. Saturdays and Sundays are not included because during the weekend markets are closed). Transactions are evenly distributed between the 12 weeks, and most of them are executed during market opening hours, while only a small percentage is executed during the early morning hours and at the end of the day. Table 1 shows a summary of the statistics of the dataset. Key fields contained in the data include the transaction amount, the product class (There are 17 different products representing the main product traded in the capital market – e.g., Equity, Fixed Income), product type (e.g., cash equity, future equity, bond), time field, currency, market. Within the data, it is not possible to identify any specific statistical distributions in any key field.

A. SYNTHETIC ANOMALIES OVERVIEW

Financial datasets are known to be extremely unbalanced, usually containing from 0.1% to 1% [18] of anomalous transactions. This information was also confirmed by the domain experts we interviewed. Therefore, to replicate real-world scenarios, we set the number of anomalies to less than 1% of the data. As part of this dataset, we generated five classes of anomalies based on examples of suspicious patterns suggested by FATF [3], an inter-governmental body that promotes effective implementation of legal, regulatory, and operational measures for combating money laundering. Anomalous transactions will follow the same trend to reinforce the concept that anomalous transactions are well hidden in the dataset. Below, we describe the classes of anomalies injected into the dataset.

Small but highly frequent transactions generated within a short timeframe: A pattern that contains multiple transactions below applicable reporting thresholds.

Transactions with rounded normalized amounts bought or sold within an account: It is unusual for transactions in capital markets to have rounded amounts (unless they occur in markets where foreign exchange conversion causes rounding errors).

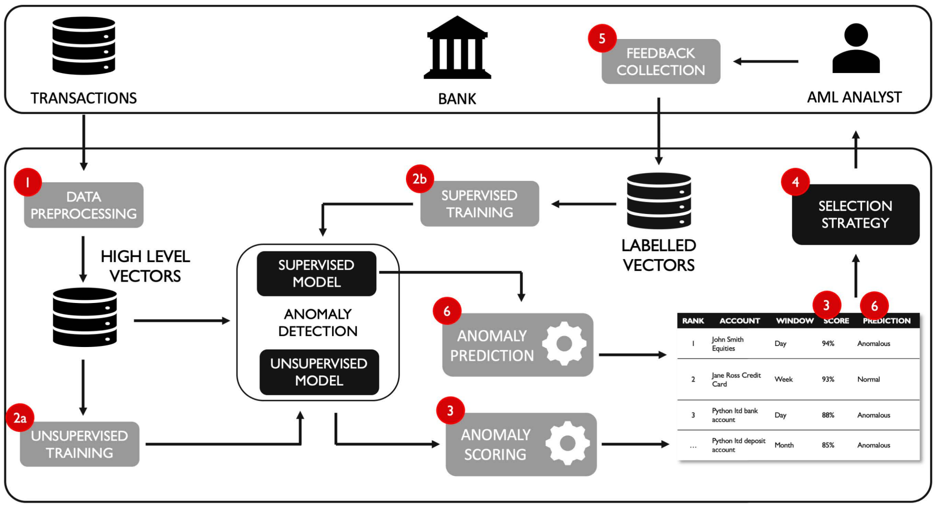
Security bought or sold at an unusual time: It is unusual for clients to trade specific securities outside of a specific timeframe (for example, outside of the opening and closing times of a stock exchange).

Large asset withdrawal: A sudden spike in transaction amount withdrawn from an account or transferred out, which deviates from the previous transactional activity and is absent of any commercial rationale or related corporate action event.

An unusually large amount of collateral transferred in and out of an account within a short period of time: This behavior is unusual as a client would not be able to invest by simply trading collateral, or at least such a strategy would be unusual.

V. AMARETTO APPROACH

Supervised detection of money laundering requires sufficient labeled data. The only way to have reliable labels is to have all transactions manually reviewed by subject matter experts, which is not feasible. For this reason, we opted for a hybrid solution, using active learning [43]. This consists of combining both unsupervised and supervised techniques in an analyst in-the-loop framework. In this active learning framework, the system uses unsupervised learning to analyze the most suspicious activities ranked by *anomaly score*. Supervised models are, then, trained on the domain experts'



RANK	ACCOUNT	WINDOW	SCORE	PREDICTION
1	John Smith Equities	Day	94%	Anomalous
2	Jane Ross Credit Card	Week	93%	Normal
3	Python ltd bank account	Day	88%	Anomalous
...	Python ltd deposit account	Month	85%	Anomalous

FIGURE 1. **Amaretto approach overview. The main steps are highlighted in red: (1.) Aggregation of historical transactions into high-level vectors; (2a.) Training of the unsupervised learning model with the high-level vectors; (3.) Anomaly scoring of the high-level vectors by the unsupervised model; (4.) Application of the selection strategies; (5.) Collection of the analyst's labels; (2b.) Training of the supervised learning model with the high-level vectors of step 1 and the feedback collected in step 4; (6) Aggregation of the supervised and unsupervised scores (i.e., prediction).**

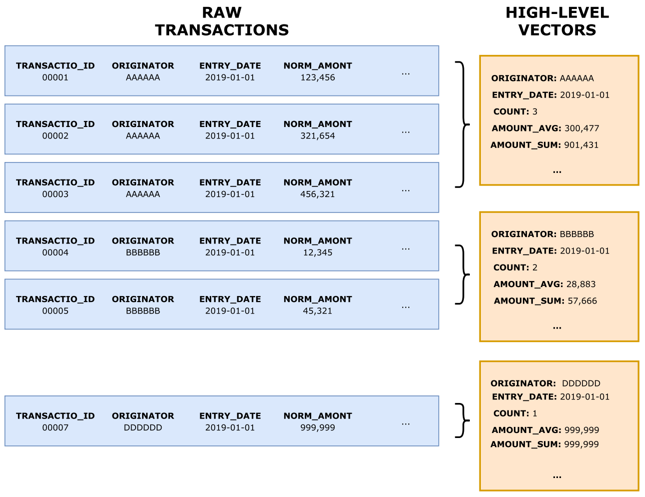
feedback (i.e., labeled dataset) to select additional data points for review.

A. APPROACH OVERVIEW

In Figure 1, we present an overview of the approach implemented in Amaretto. The first step in the Amaretto workflow is to aggregate the raw transactional data across a specific timeframe to produce features representing high-level vectors that capture the behavioral profile of a customer. The models employed in Amaretto are trained with these high-level vectors generated from historical data. After the training phase, Amaretto computes an *anomaly score* for each new vector using both unsupervised and supervised models (if enough data is available to train the latter). A specific selection strategy based on the *anomaly score* is then used to choose vectors that will be sent to the domain expert for review. The number of transactions sent each day for review (k) is a parameter of our system, based on the resources that a financial institution can allocate to this task. The domain expert analyzes these transactions to ascertain whether they are anomalous or not. This information is then saved as labels in the dataset. The reviewed labels contribute to a historical set of labeled data that is the input for the supervised component of the system. The supervised component is then used alongside the unsupervised model to continuously select the data to be reviewed by the domain expert. In Algorithm 1, we outline the pseudocode for the key steps of Amaretto.

B. DATA PREPROCESSING MODULE

Amaretto generates a set of high-level features derived from the *transactional data*. These aggregated features are based on an *aggregation window*. In particular, this window represents the time over which transactions are aggregated and is used for computing each set of aggregated features. The *aggregation windows* have the objective of capturing the short-term, mid-term, and long-term behavior of the user. We look for the most used sizes in literature [17], [37]: We use 1 hour and 1 day for the short-term, 7 days for the mid-term, and 1 month for the long-term. For example, as shown in Figure 2, if a daily window is chosen, the aggregated features' set is produced for each day by aggregating all the transactions the customer performed on that day. Also, aggregating transactions over a period of time is helpful in the AML use case since it can be used to capture correlations over time across multiple transactions. Within each period,



TRANSACTIO_ID	ORIGINATOR	ENTRY_DATE	NORM_AMONT	...
00001	AAAAAA	2019-01-01	123,456	...
00002	AAAAAA	2019-01-01	321,654	...
00003	AAAAAA	2019-01-01	456,321	...
00004	BBBBBB	2019-01-01	12,345	...
00005	BBBBBB	2019-01-01	45,321	...
00007	DDDDDD	2019-01-01	999,999	...

ORIGINATOR: AAAAAA
ENTRY_DATE: 2019-01-01
COUNT: 3
AMOUNT_AVG: 300,477
AMOUNT_SUM: 901,431
...

ORIGINATOR: BBBBBB
ENTRY_DATE: 2019-01-01
COUNT: 2
AMOUNT_AVG: 28,883
AMOUNT_SUM: 57,666
...

ORIGINATOR: DDDDDD
ENTRY_DATE: 2019-01-01
COUNT: 1
AMOUNT_AVG: 999,999
AMOUNT_SUM: 999,999
...

FIGURE 2. **Graphical representation of the aggregation of raw transactions into high-level vectors. On the left, it shows the sequence of raw transactions, while, on the right, the aggregated high-level vectors.**

Algorithm 1 Pseudocode of Amaretto's Approach: $\mathcal{L}$ Is the Set of the Feedback Received by the Fraud Analyst; $\mathcal{U}$ Is the Set of Transactions; *Mod* Is the Machine Learning Model; *sup* Stands for Supervised; *Unsup* Stands for Unsupervised; *Strat* Is the Selection Strategy; $\mathcal{K}$ Denotes the Top-$\mathcal{K}$ High-Level Vectors in the Ranking; $\mathcal{T}$ Denotes the Set of Time Windows)

```
Input: $\mathcal{L} = \emptyset, \mathcal{U}, mod_{sup}, mod_{unsup}, strat_{sup}, strat_{unsup}, \mathcal{K}, \mathcal{T}$
1: for $t \in \{0, \dots, T\}$ do
2:     if $t = 0$ then
3:         Train $mod_{unsup}$ on $\mathcal{U}^{t-1}$
4:         Compute the scores $S(x_i)$ where $x_i \in \mathcal{U}^t$
5:         Query $\mathcal{K}$ samples from $\mathcal{U}^t$ using the sampling strategy $strat_{unsup}$
6:         $sample^t_{unsup}$ = Collect analyst feedback
7:         $\mathcal{L}^t = \mathcal{L}^{t-1} \cup (x_i, y_i) \in sample^t_{unsup}$    ▷ Add the selected points to $\mathcal{L}^{t-1}$
8:     end if
9:     if $t \geq 0$ then
10:        Train $mod_{unsup}$ on $\mathcal{U}^{t-1}$
11:        Train $mod_{sup}$ on $\mathcal{L}^{t-1}$
12:        Compute the scores $S(x_i)$ where $x_i \in \mathcal{U}^t$
13:        Query $\frac{\mathcal{K}}{2}$ samples from $\mathcal{U}^t$ using the sampling strategy $strat_{unsup}$
14:        $sample^t_{unsup}$ = Collect analyst feedback
15:        $\mathcal{U}^t = \mathcal{U}_t \setminus sample^t_{unsup}$
16:        Select $\frac{\mathcal{K}}{2}$ samples from $\mathcal{U}$ using the
17:        sampling strategy $strat_{sup}$
18:        $sample^t_{sup}$=Collect analyst feedback
19:        $\mathcal{L}^t = \mathcal{L}^{t-1} \cup sample^t_{unsup} \cup sample^t_{sup}$ ▷ Add the selected points to $\mathcal{L}^{t-1}$
20:    end if
21: end for
```

multiple features are extracted and aggregated: total amount traded; average amount traded; the number of transactions; the number of transactions traded for each product class; the number of transactions traded for each currency; total amount traded for each product type for each product class; total amount traded for each product type for each currency; the average amount traded for each product type for each product class; the average amount traded for each product type for each currency and the number of transactions traded during specific times of the day. When aggregating transactions over a timeframe for a customer, the set of aggregated transactions is considered anomalous if at least one of the underlying transactions is anomalous. The result of the aggregation and feature extraction process will be referred to as *high-level vectors*. First of all, the *EntryDate* column is used to extract temporal features like *Weekday*, *Month*, *Hour*. The DataFrame containing the transactional data is grouped by using the *Originator* and the temporal features mentioned before. Then, the financial features are extracted from the *DataFrameGroupBy* object. A data frame is created for each window in which the user performed at least one transaction, collecting all the customer's activities. These records are uniquely indexed through the features used to create the *DataFrameGroupBy*. The financial features extracted in this phase are designed to model the behavioral signature of the user in each window, capturing the spending patterns. The features of interest comprise a combination between *Currency*, *Product Class*, *Product Type*, and *InputOutput* columns. These high-level features are carefully selected, exploiting the domain expertise of the Napier AI team to be able to detect all kinds of behavioral variations that might indicate anomalous activity. The first step is to transform the *Product Class*, *Product Type* into new features called *Cash* and *Collateral* that indicate if a transaction is performed through a *simple transfer* using *cash* or other types of *security*. Then, a first aggregation, called *Amount_IO_Aggregation*, is carried out extracting information about the amount of the transactions included in the aggregation window as statistical features like *mean_amount*, *sum_amount* or *code_small_amount*, *code_round_amount*. Furthermore, during this step, *InputOutput_delta* and *Collateral_delta* are determined, indicating the difference between bought and sold operations or the difference between collateral and the other securities. Afterward, another *DataFrameGroupBy* object, called *Product_Currency_Aggregation*, is created aggregating by *Currency*, *Product Class*, *Product Type*, and *InputOutput* columns and computing the *mean_amount*, *sum_amount*, and *count* for each different value of the pivot columns. Finally, the two aggregations, *Amount_IO_Aggregation* and *Product_Currency_Aggregation*, are merged and indexed using the *Originator* and the temporal columns. This is the final high-level vector used to train or analyzed by the *Anomaly Detection Module*. An example of high-level vector is shown in Table 2.

C. UNSUPERVISED MODULE

As pointed out in Section III, an unsupervised method is essential to detect new anomalous patterns never seen before. We decided to use Isolation Forest [20], [21] due to its high performance in detecting outliers even if they are present in small amounts [20], [21]. Another feature of Isolation Forest is its ability to deal with random noise. This is particularly useful in scenarios where subject matter experts may provide an incorrect label for a set of transactions.

Isolation Forest: The Isolation Forest algorithm is based on the isolation principle: it tries to separate data points from one another by recursively and randomly splitting the dataset into two partitions along its features axes. The idea is simple: if a point is an outlier, it will not be surrounded by many other points, and therefore it will be easier to isolate it from the rest of the dataset with random partitioning. The algorithm uses the training set to build a series of isolation trees, which, when combined, form the Isolation Forest; each isolation tree is built upon a subset of the original data, randomly sampled.

TABLE 2. Features (and example of values) of the high-level vector resulting from the aggregation of raw transactions.

Feature name	Value
Originator	Client_304
EntryDate	2019-01-01 00:00:00
Weekday	1
Hour	7
Night	1
Morning	0
Evening	0
Anomaly	1
Transactions_count	73
Normalized Amount_sum	20886919.32000001
Transactions_count_Small_Amount	15.0
Transactions_count_Round_Amount	0.0
InputOutput_delta	17
Collateral_delta	-39
Transactions_count_Buy_Collateral_Cash_Currency1	0.0
Transactions_count_Buy_Collateral_Security_Currency1	3.0
Transactions_count_Sell_Collateral_Cash_Currency1	1.0
Transactions_count_Sell_Collateral_Security_Currency1	2.0
Normalized Amount_sum_Buy_Collateral_Security_Currency1	34888.65
Normalized Amount_sum_Sell_Collateral_Cash_Currency1	16782.43
Normalized Amount_sum_Sell_Collateral_Security_Currency1	27584.75
Normalized Amount_mean_Buy_Collateral_Cash_Currency1	0.0
Normalized Amount_mean_Buy_Collateral_Security_Currency1	11629.550000000001
Normalized Amount_mean_Sell_Collateral_Cash_Currency1	16782.43
Normalized Amount_mean_Sell_Collateral_Security_Currency1	13792.375

The splitting is performed along a random feature axis, using a random split value that lies between the minimum and maximum values for that feature amongst the data points in that partition. This split process is performed recursively until a single point has been isolated from the others. The number of splits required to isolate an outlier is likely to be much smaller than the one needed by a regular point due to the lower density of points in the surrounding feature space. Isolation Forest leverages an ensemble of isolation trees, with anomalies exhibiting a closer distance to the root tree. The anomaly score can be derived from path length $h(x)$ of a point x, which is defined as the average number of splits required to isolate the point across all the trees in the forest. The anomaly score s of an instance x is defined as $s(x, n) = 2^{-\frac{E(h(x))}{c(n)}}$ and $c(n) = 2H(n-1) - \frac{2(n-1)}{n}$, where $E(h(x))$ is the average of $h(x)$ from a collection of isolation trees. Furthermore, $c(n)$ is the average path length of unsuccessful searches in binary search trees.

The system extracts high-level vectors related to historical transactions for each customer and uses an Isolation Forest to generate an anomaly score per high-level vector. We built a model for each customer to capture variations in individual behaviors and, at the same time, use the score to compare different behaviors. Subsequently, the score generated by the Isolation Forests is used as part of the selection strategy of the system to select the transactions for the subject matter expert to investigate.

D. SUPERVISED MODULE

A supervised model, used alongside an unsupervised model, improves the ability of the system to make future predictions. Supervised models usually yield more accurate predictions than unsupervised ones. Therefore the combination of these approaches leads to more robust results. In Amaretto, we adopt Random Forest [44] because it exhibited the best performance when compared to other supervised algorithms (as highlighted in Section III).

Random Forest: The basic component of a Random Forest is a Decision Tree [44]. It is a structure that allows the categorization of data points into different classes. Starting from the root node, each data point traverses through different branches of the tree, depending on conditions set out for each node, until a leaf node is reached. The node rules are simple conditions verified by a given feature of the data point (e.g., is feature $a_1 \geq K$? Or, for categorical features, is feature a_2 equal to C_0?). In the leaf node, the class of the data point is determined by looking at the most common / majority class present in that leaf node. A major advantage of this technique is the possibility to explain the outcome of the algorithm by following the route of the datapoint through the tree to determine which conditions were met / not met in order to classify the point. One of the key challenges in using decision trees is overfitting. To deal with this problem, an ensemble of multiple decision trees can be utilized to form a Random Forest [45], which consists of multiple weak learners characterized by low bias and high variance. The bagging ensemble of these weak learners will be a robust model since the overall prediction is made by averaging the prediction of each individual tree. Initially, when the system is bootstrapped, no labeled data is available. In this situation, the unsupervised model is used for anomaly detection. When enough feedback from the subject matter expert is collected, it is possible to train the Random Forest model. We train a single Random Forest model using high-level vectors across

Algorithm 2 First Stage: *SELECT-TOP* Strategy. High-Level Vectors Are Ranked in Decreasing Order Based on the Anomaly Score, and the Topmost Anomalous Vectors Are Selected

```
Input: $\mathcal{U}^t$, $ratio_{unsup}$
Output: $sample^t_{unsup}$
1: Sort $\mathcal{U}^t$ by unsupervised score
2: $\mathcal{C}$ = Select $ratio_{unsup}$ most anomalous aggregations.
3: $sample^t_{unsup} = sample^t_{unsup} \cup \mathcal{C}$        ▷ Append $\mathcal{C}$ to $sample^t_{unsup}$
4: Return $sample^t_{unsup}$
```

all customers. The supervised model outputs an *anomaly score* by computing the probability that a high-level vector is anomalous or not. As new labels are obtained from the subject matter expert, the Random Forest is re-trained accordingly, and predictions are run against the remaining unlabeled vectors (unlabeled data).

E. SELECTION STRATEGIES MODULE

Amaretto combines supervised and unsupervised learning in three stages, each one with a different selection strategy.

1) FIRST STAGE: NEW ANOMALOUS PATTERNS DETECTION

The purpose of the first stage is to detect new anomalous patterns as well as common anomalous patterns. As previously mentioned, the *anomaly score* computed by the Isolation Forest is fundamental to detecting new anomalous behaviors. Two possible active learning selection strategies are available for this stage: the SELECT-TOP and SELECT-DIVERSE strategies. In the SELECT-TOP strategy, high-level vectors are ranked in decreasing order based on the *anomaly score* generated by the Isolation Forest. The system then selects the topmost anomalous vectors. However, this approach may not guarantee that all types of anomalies are covered (i.e., the top anomalies by *anomaly score* may all belong to the same anomaly type). In Algorithm 2 we present the pseudocode of the SELECT-TOP strategy.

As previously evidenced in Section III by [38], it is important to diversify the type of unusual patterns that are selected. For this reason, the SELECT-DIVERSE strategy uses clustering to group similar high-level vectors and draw samples from each cluster based on the *anomaly score*. Samples are drawn from each cluster, starting from the least dense cluster until the desired number of samples has been reached. The decision of starting from the least dense cluster is motivated by the following assumption: given that the number of non-anomalous high-level vectors is greater than the number of anomalous vectors, the latter should form less dense clusters. In Algorithm 3 we present the pseudocode of the SELECT-DIVERSE strategy.

The clustering algorithm used for this strategy is HDBSCAN [46], [47]. This algorithm is based on the work by [46] and [47]. The first step of the algorithm is to build a weighted graph, where each data point

Algorithm 3 First Stage: *SELECT-DIVERSE* Strategy. It Clusters Similar High-Level Vectors and Draws Samples From Each Cluster Based on the *Anomaly Score* Until the Desired Number of Samples Has Been Reached

```
Input: $\mathcal{U}^t$, $ratio_{unsup}$, $p_{unsup}$
Output: $sample^t_{unsup}$
1: Sort $\mathcal{U}^t$ by unsupervised score
2: $x_p$ = Find $p_{unsup}$ percentile
3: $anoms_{unsup}$ = Select $x_i \subset \mathcal{U}^t$ with score $s_i > x_p$
4: Run HDBSCAN algorithm on $anoms_{unsup}$
5: Define $cluster_{ratio} = \max\{1, \frac{ratio_{unsup}}{n_{clusters}}\}$
6: Sort cluster by $cluster_{density}$
7: for $i \in \{0, \ldots, n_{clusters}\}$ do
8:     $\mathcal{C}_i$ = Select $cluster_{ratio}$ samples from $cluster_i$
9:     Append $\mathcal{C}_i$ to $sample^t_{unsup}$
                ▷ i.e. $sample^t_{unsup} = sample^t_{unsup} \cup \mathcal{C}_i$
10:    if $|sample^t_{unsup}| \geq ratio_{unsup}$ then
11:        Break
12:    end if
13: end for
14: Return $sample^t_{unsup}$
```

represents a node of the graph. The weights of this graph are computed using a metric called *mutual reachability distance* between two points, defined as: $d(a, b) = \max\{core_k(a), core_k(b), d(a, b)\}$. $core_k(x)$ is the core distance for a point x which is the distance between that point and its k-th farthest neighbor. The *mutual reachability distance* defines the density of the areas around each point, and it is used for spreading apart isolated points. A minimum spanning tree is constructed from the resulting graph using Prim's algorithm, which aims to connect every point in the graph whilst minimizing the total weight of the edges in the resulting graph. The next part of the algorithm focuses on building a hierarchy of clusters. This is achieved by removing all edges sorted by decreasing weight. This split process is recursively performed, starting with the edges of the tree that have the lowest weight. This is defined by a parameter "minimum cluster size". The first step in cluster extraction is condensing down the large and complicated cluster hierarchy into a smaller tree. The key point is to consider points that are split close to a cluster belonging to this single persistent cluster. To do so, the notion of minimum cluster size is applied. Again, a different measure than distance is defined to measure the persistence of clusters: $\lambda = \frac{1}{\text{distance}}$. For a given cluster, values λ_{birth} and λ_{death} represent the value when the cluster split off and became its cluster and the lambda value (if any) when the cluster split into smaller clusters respectively. In turn, for a given cluster, for each point p in that cluster, we can define the value λ_p as the lambda value at which that point "fell out of the cluster", which is a value somewhere between λ_{birth} and λ_{death}. This is because the point either falls out of the cluster at some point in the cluster's lifetime or leaves the cluster when the cluster splits into two smaller clusters. For each cluster we compute the

Algorithm 4 Third Stage: *SELECT-ENTROPY* Strategy. It Uses the Probability Scores Generated by the Supervised Model. Samples Whose Probability Is Close to 0.5 Have a High Chance of Being Selected Due to Their High Entropy and, Hence, Uncertainty

```
Input: $\mathcal{U}^t$, $ratio_{sup}$, $p_{center}$
Output: $\mathcal{C}^{center}$
1: $dist_i = |s_i - 0.5|$   ▷ Compute the distance of the score for each aggregation $i$ to the center = 0.5
2: Sort $dist_i$ in ascending order
3: $\mathcal{C}^{center}$=Select $ratio_{sup} \times p_{center}$ least distant aggregations
4: Return $\mathcal{C}^{center}$
```

stability as $\sum_{p\in\text{cluster}}(\lambda_p - \lambda_{\text{birth}})$. If the sum of the stabilities of the child clusters is greater than the stability of the cluster, then we set the cluster stability to be the sum of the child stabilities. If, on the other hand, the stability of the cluster is greater than the sum of its children, then we declare the cluster to be selected and unselect all its descendants. Once we reach the root node, we call the current set of selected clusters our flat clustering and return that.

2) SECOND STAGE: RANDOM FOREST SELECTION

The second stage of the selection phase relies on the probability score generated by the Random Forest classifier. This process comprises two steps: the first one is the selection of the most anomalous vectors; the second one is the selection of the least anomalous aggregated transactions. The purpose of this stage is to take advantage of the higher accuracy of the Random Forest classifier to reinforce the information contained in the labeled dataset by automatically scoring all transactions in the dataset.

3) THIRD STAGE: UNCERTAIN DATAPOINTS SELECTION

In the third and last stage of the selection phase, the system selects the high-level vectors for which the two models show the most uncertainty. To assess the level of uncertainty, two active learning strategies can be followed: the SELECT-ENTROPY and SELECT-CONFLICT strategies. The SELECT-ENTROPY uses the probability scores generated by the supervised model. Those samples whose probability is close to 0.5 have a high chance of being selected due to their high entropy and hence the uncertainty. In Algorithm 4 we present the pseudocode of the SELECT-ENTROPY strategy.

The SELECT-CONFLICT strategy takes into account the difference between the scores generated by the supervised and unsupervised model. A discrepancy in the score for each set of high-level vectors indicates that the outputs of the two models disagree. For this reason, the samples with a score discrepancy close to 1.0 are selected (i.e., the models disagree on whether the vectors are anomalous or not). In Algorithm 5 we present the pseudocode of the SELECT-CONFLICT strategy.

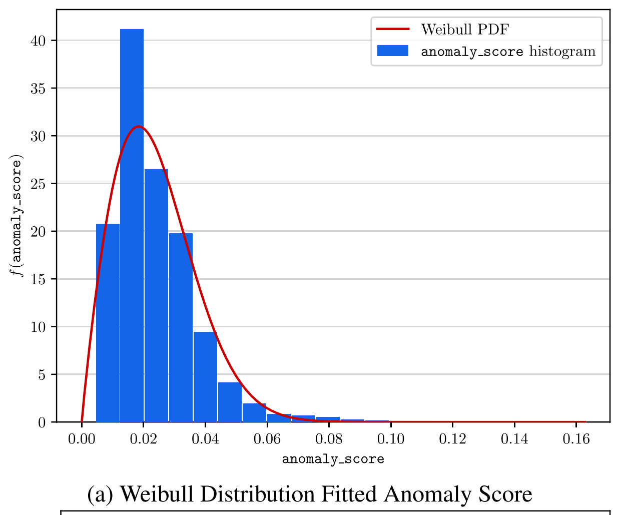
Algorithm 5 Third Stage: *SELECT-CONFLICT* Strategy. It Takes Into Account the Difference Between the Scores Generated by the Supervised and Unsupervised Model. A Discrepancy in the Score for Each Set of High-Level Vectors Indicates That the Outputs of the Two Models Disagree and, Therefore, They Are Selected

```
Input: $\mathcal{U}^t$, $ratio_{sup}$, $p_{center}$
Output: $\mathcal{C}^{center}$
1: $unc_i$ = Compute the difference between the supervised score and the unsupervised score for each aggregation $i$
2: Sort $unc_i$ in descending order
3: $\mathcal{C}^{center}$=Select $ratio_{sup} \times p_{center}$ most uncertain aggregations.
4: Return $\mathcal{C}^{center}$
```

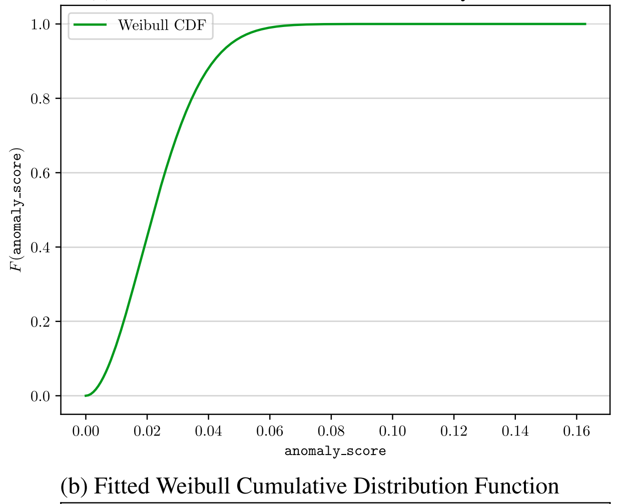
Standardization and Ensembling: Amaretto exploits the power of Random Forest, which outputs class probabilities $\in [0, 1]$, and Isolation Forest, which outputs an anomaly score $\in [-1, +1)$. Even if the models yield outputs in the same range (e.g., probabilities in $[0, 1]$), their prediction distribution could be significantly different, so the sum of the predictions could be misleading. For this reason, we combine the supervised and unsupervised *anomaly scores* using an ensemble technique based on the Weibull distribution (see Figure 3). We selected the Weibull probability distribution because of its shape and flexibility, which fits the anomaly score distribution and allows us to better discern between normal and anomalous instances (i.e., it amplifies the distance between these two classes). By doing this, we transform the *anomaly scores* produced by each model into probabilities in the interval [0, 1]. The following procedure is carried out to perform the ensembling: we fit a Weibull probability distribution function to the *anomaly scores* produced by each model (see Figure 3a). Then, we compute the corresponding cumulative distribution function through integration (see Figure 3b). Finally, for each new prediction $s_{\vec{x}}$, we redefine the *anomaly scores* as $F(s_{\vec{x}}) = P(\vec{x} \leq s_{\vec{x}})$. This is performed by plugging the old *anomaly scores* into the Weibull cumulative distribution function (see Figure 3c).

VI. EXPERIMENTAL EVALUATION

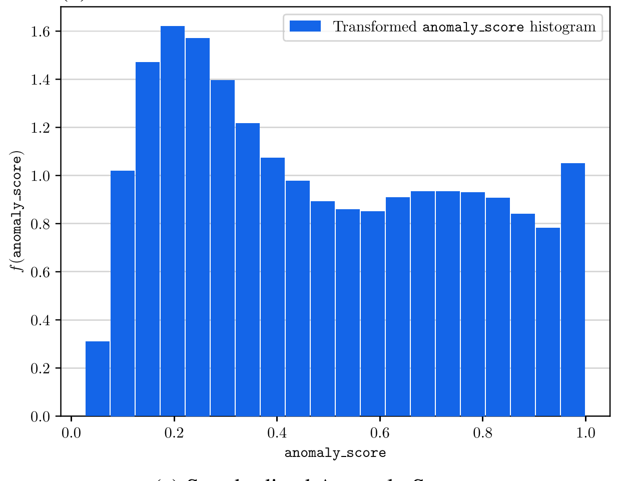
In this section, we describe the experiments conducted to assess the performance and effectiveness of Amaretto. First, we compare the Isolation Forest used in Amaretto with state-of-the-art unsupervised solutions to confirm that our choice is the best for money laundering detection (Section VI-B). Then, we test the unsupervised techniques to assess their prediction ability with different daily budgets (Section VI-C). Afterward, we evaluate Random Forest against other supervised solutions to prove that they perform the best in our domain (Section VI-D). Furthermore, we prove the importance of an unsupervised model in combination with a supervised one in detecting new anomalous patterns (Section VI-E). Then, we compare the different selection



(a) Weibull Distribution Fitted Anomaly Score



(b) Fitted Weibull Cumulative Distribution Function



(c) Standardized Anomaly Score

FIGURE 3. Anomaly score standardization and ensemble. In Figure 3a a Weibull probability distribution function is fit to the *anomaly scores* of each model. Then, in Figure 3b the corresponding cumulative distribution function through integration is computed. Finally, in Figure 3c the old *anomaly scores* are plug into the Weibull cumulative distribution function to obtain the standardized score.

strategies of Amaretto (Section VI-F). Finally, we compare Amaretto with AI^2, a state of the art active learning framework, in an AML scenario (Section VI-G).

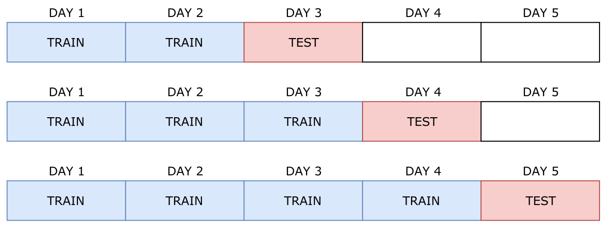


FIGURE 4. Graphical representation of the walk-forward approach used for evaluating Amaretto and its active learning functionality. Each day, K samples from the unsupervised ranking are provided to the analyst for labeling; based on the labeled received, both supervised and unsupervised systems are trained (represented in blue). The trained model are used for scoring the following day (represented in red).

A. EVALUATION APPROACH AND METRICS

The data contained in our dataset can be considered as time-series data. We split the dataset into two sets. The first one, which is used for training the models and for hyper-parameter optimization, contains the first 7 weeks of data, corresponding to 17,327,387 transactions. The second one, which is used to evaluate the model performance by running tests, includes the subsequent 5 weeks of data, corresponding to 12,376,703 transactions. It is important to highlight that we perform the entire experimental evaluation on the same dataset to provide a meaningful comparison between Amaretto and the other approaches under analysis.

Given the temporal link of the data, for the experiments in which Amaretto was evaluated in a realistic setting (Section VI-C and VI-G), we used a walk-forward testing approach [48], as shown in Figure 4. This allows us to fully test the system on a daily working routine, like the real-world scenario in which a subject matter expert has to investigate a set of anomalous cases each day. We split the testing data on a daily basis: Each "simulated" day, K samples from the unsupervised ranking are provided to the analyst for investigation (i.e., labeling). Based on the assigned labels, Amaretto will train both the supervised and unsupervised learning models and will use them for ranking the samples of the subsequent day. In Section VI-C we provide an analysis of daily budget K that allows the system to achieve a suitable detection rate whilst minimizing the effort of the subject matter expert in reviewing the samples.

For the hyper-parameter optimization, we used *Bayesian Optimization* [49] due to its ability to achieve accurate parameter selection within a reasonable amount of time. *Bayesian Optimization* is a probabilistic model-based approach for finding an input value or a set of values to an objective function that yields the lowest loss.

1) METRICS

To evaluate Amaretto, we adapt common evaluation metrics to our context. A *True Positive* (TP) is an anomalous high level vector correctly classified as anomalous, *False Positive* (FP) is a legitimate high level vector wrongly ranked as anomalous, a *False Negative* (FN) is an anomalous vector wrongly ranked as legitimate, and a *True Negative* (TN) is a legitimate vector correctly ranked as non-anomalous. On the basis of these definitions, we assess the system performance by computing:

Accuracy: Percentage of high-level vectors correctly classified:

$$Accuracy = \frac{TN + TP}{TN + FP + FN + TP}$$

Precision: Proportion of *TP* over the vectors considered as anomalies:

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

True Positive Rate or Recall: Percentage of correctly identified anomalous vectors:

$$TPR = \frac{TP}{TP + FN}$$

False Positive Rate: Percentage of legitimate vectors that are wrongly identified as anomalous:

$$FPR = \frac{FP}{TN + FP}$$

F-Score: Harmonic mean between the Recall and the Precision:

$$FScore = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall}$$

Matthews Correlation Coefficient: Quality of the detection rate in terms of the correlation coefficient between the observed and predicted classifications; a coefficient of +1 represents a perfect ranking, 0 no better than a random prediction, and −1 indicates total disagreement between prediction and observation:

$$\text{MCC} = \frac{TP \cdot TN - FP \cdot FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}}$$

Area Under the Receiver Operating Characteristic (ROC) Curve: This is the area under the *ROC curve*, obtained by plotting the *TPR* against the corresponding *FPR* at various threshold settings. The *AUROC* gives a measure of the solution performance, where a perfect model has an *AUROC* of 1.

The test data is very imbalanced (0.27% of anomalous transactions), so metrics like accuracy are not very meaningful. However, to make a fair comparison with the state-of-the-art solution, all metrics are included as a reference.

The *AUROC* is a useful indicator for benchmarking algorithms; if the *ROC curve* of a model is consistently higher than the curve of other estimators, this indicates the former achieves better performance. For these reasons, we use the *AUROC* and the *ROC curve* to assess the performance of various unsupervised models and the metrics described above for assessing the performance of the supervised models.

We also considered additional metrics to account for class imbalance and different classification costs: the **Precision-Recall Curve** and the **Cost Metric**. The **Precision-Recall Curve** shows the tradeoff between Precision and Recall for different thresholds. A high area under the curve represents both high Recall and high Precision, where high Recall relates to a low false negative rate, and high Precision relates to a low false positive rate. High scores for both reflect that the classifier is returning accurate results (high Precision), as well as returning a majority of all positive results (high Recall). The **Cost Metric** is described in [50] as:

$$Cost = FP + C\_R \times FN$$

A normalization process can be applied to obtain a value that is independent of the number of transactions:

$$Norm\_Cost = \frac{FP + C\_R \times FN}{TN + FP + C\_R \times (TP + FN)}$$

As suggested in [50], 100 is a reasonable estimation of C_R, that is the *cost ratio* between *FN* and *FP*. 100 was the value used to assess the optimal operating condition of our system. However, it could be set to reflect the real costs of anomalous transactions based on different scenarios. This metric takes into account the cost of false positives for an institution. A unit cost is applied to a *FP*, whilst a higher cost is applied for a *FN* since the cost of allowing a money launderer in the system is hundreds of times higher than the cost of false positives and may result in fines for the institution.

B. EXPERIMENT 1: UNSUPERVISED ALGORITHM COMPARISON

In this experiment we compare state-of-the-art unsupervised solutions with the Isolation Forest used in Amaretto. In particular, we take into consideration Autoencoders [25], Variational Autoencoders [51], Extended Isolation Forests [52]. In addition, we also test the unsupervised techniques described in [37], which exploits a Matrix-decomposition model, a Density-based model, and an Autoencoder, using PCA as a Matrix decomposition model [53] and using a Copula distribution as a Density-based model. We also tested a threshold-based model [15] that uses mean and standard deviation computed for each feature of the high-level vector. Given these descriptive statistics, we compute a one-sided threshold as the sum of the mean and standard deviation. In order to score new samples, all features that exceed their respective threshold add the surplus to the risk score, while features below the threshold yield a risk score of 0. For this experiment, we train and tune all the models on the training dataset composed of 7 weeks of data (17,327,387 transactions) and evaluate the performance on the subsequent five weeks of data (12,376,703 transactions). As shown in Figure 5, Isolation Forest exhibits the best performance with an *AUROC* of 0.9. Surprisingly, the threshold-based model and the matrix decomposition-based model outperformed the Auto Encoder, which is considered one of the best models for outlier detection.

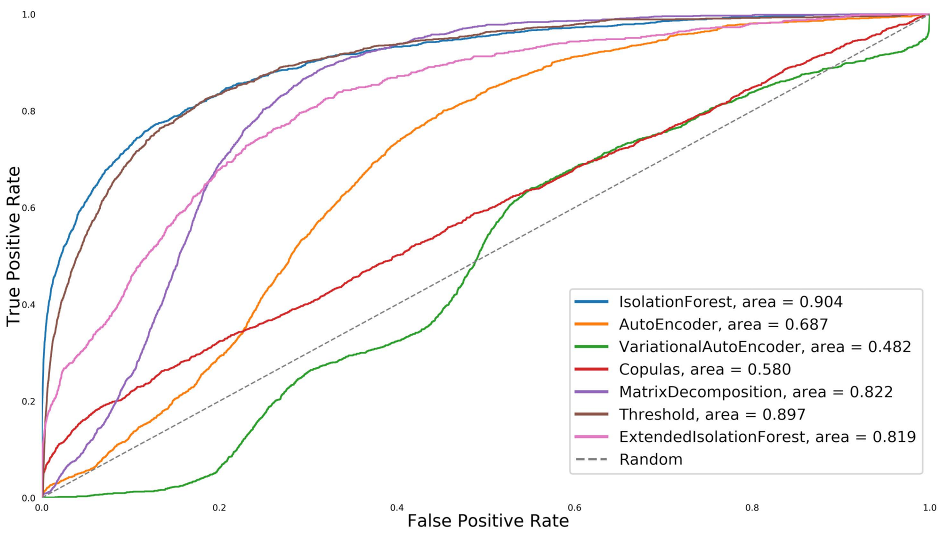


FIGURE 5. Experiment 1—Comparison of the ROCs of the unsupervised algorithms. Isolation Forest exhibits the best performance with an AUROC of 0.904, while variational autoencoders perform worst.

C. EXPERIMENT 2: DAILY BUDGET K ESTIMATION FOR UNSUPERVISED RANKING

For this experiment, we benchmark the performances of the unsupervised models analyzed in the previous experiments when varying the number of samples reviewed each day by the analyst. For every day existing in the test set, each model computes the *anomaly score* for the high-level vectors, which is then used to rank the vectors. Then the K top anomalous vectors are considered anomalous, e.g., for $K = 10$, the first 10 vectors with the highest score are selected for the review. The purpose of this experiment is to assess the best *daily budget* that allows the system to achieve a suitable detection rate whilst minimizing the effort of the subject matter expert in reviewing the high-level vectors. The metrics presented in Table 3 are the average metrics computed for each technique and budget. As shown in Table 3, the Isolation Forest is the model that achieves the best results for every budget K, achieving an average *Precision* of 0.904 and an average *FPR* of less than 0.01 (for $K = 10$). This means that the Isolation Forest allows the subject matter expert to focus only on the most anomalous vectors. The matrix decomposition-based model achieves comparable performance only with a higher budget K, whilst for lower values, the Isolation Forest is better. The daily budget values considered in this experiment represent a small percentage of the daily vectors that are generated. For this reason, the *FNR* is high for a small daily budget, and it reduces as the budget increases. It is important to highlight that the financial company with which we collaborated considers a percentage around the 1% and the 2% of the data received daily as a reasonable number of transactions that they can manually inspect with its specialized analysts. In our dataset, this percentage corresponds to $K = 5$ (approximately 6,000 transactions) and $K = 10$ (approximately 12,000 transactions).

D. EXPERIMENT 3: SUPERVISED ALGORITHMS COMPARISON

With this test, we compare the Random Forest model of Amaretto with state-of-the-art supervised models based on Gradient and Category Boosting techniques. Gradient Boosting [54] is considered one of the best algorithms for classification tasks. Category Boosting [55] is an alternative boosting algorithm based on decision trees. It offers computational and efficiency improvements compared to Gradient Boosting-based models. For this experiment, we train and tune all the models on the training dataset composed of 7 weeks of data (17,327,387 transactions) and evaluate the performance on the subsequent five weeks of data (12,376,703 transactions), running predictions on a daily basis.

Table 4 presents the average metrics for each technique. In line with the results obtained by state-of-the-art works [30] and as shown in Table 4, the metrics are quite similar between Random Forest, Category Boosting (CatBoost), and Gradient Boosting (LGBM) models, whilst other supervised methods do not perform as well. The CatBoost model exhibits the highest *Precision*, although Random Forest achieves the highest *TPR* and the lowest *FNR*. Furthermore, if we take into account cost-related metrics, we can conclude that Random Forest is better suited in this context compared to the other models.

E. EXPERIMENT 4: DETECTING NOVEL ANOMALOUS PATTERNS

The goal of this experiment is to assess the performance of the supervised and unsupervised techniques employed in Amaretto to detect new anomalous patterns. We execute several runs of this experiment in order to test each combination of the classes of anomalies existing in the dataset. For every run, a class of anomalies is withheld from the

TABLE 3. Experiment 2 - Daily budget K estimation. Performance metrics of the unsupervised models are shown varying the number of samples reviewed by the analyst each day (K). Isolation Forest is the model that achieves the best performance for every budget K, achieving an average Precision of 0.904 and an average FPR of less than 0.01.

K	Performance Metric	Autoencoder	Copula	Isolation Forest	Matrix Decomposition	Threshold-based
10	**TPR**	0.003	0.032	**0.142**	0.021	0.028
	FPR	0.001	0.001	**0.0**	0.001	0.001
	FNR	0.997	0.968	**0.858**	0.979	0.972
	AUC	0.612	0.558	**0.897**	0.676	0.898
	Accuracy	0.991	0.991	**0.993**	0.991	0.991
	Precision	0.015	0.208	**0.904**	0.131	0.181
	FScore	0.005	0.055	**0.246**	0.037	0.049
	MCC	0.003	0.078	**0.357**	0.05	0.068
	Norm. Cost	0.448	0.435	**0.385**	0.44	0.437
50	**TPR**	0.015	0.063	**0.279**	0.051	0.205
	FPR	0.006	0.006	**0.004**	0.006	0.005
	FNR	0.985	0.937	**0.721**	0.949	0.795
	AUC	0.612	0.558	0.897	0.676	**0.898**
	Accuracy	0.986	0.987	**0.99**	0.986	0.989
	Precision	0.018	0.081	**0.355**	0.065	0.261
	FScore	0.017	0.07	**0.312**	0.057	0.229
	MCC	0.01	0.064	**0.309**	0.05	0.225
	Norm. Cost	0.446	0.424	**0.326**	0.429	0.36
100	**TPR**	0.021	0.086	**0.366**	0.065	0.304
	FPR	0.013	0.012	**0.01**	0.012	0.01
	FNR	0.979	0.914	**0.634**	0.935	0.696
	Accuracy	0.98	0.981	**0.985**	0.98	0.984
	Precision	0.013	0.055	**0.233**	0.041	0.195
	AUC	0.612	0.558	0.897	0.676	**0.898**
	FScore	0.016	0.067	**0.284**	0.05	0.237
	MCC	0.007	0.059	**0.284**	0.042	0.236
	Norm. Cost	0.446	0.417	**0.29**	0.427	0.318
200	**TPR**	0.032	0.112	**0.463**	0.092	0.407
	FPR	0.025	0.025	**0.022**	0.025	**0.022**
	FNR	0.968	0.888	**0.537**	0.908	0.593
	Accuracy	0.967	0.968	**0.974**	0.968	0.973
	Precision	0.01	0.036	**0.147**	0.029	0.13
	AUC	0.612	0.558	0.897	0.676	**0.898**
	FScore	0.015	0.055	**0.223**	0.044	0.197
	MCC	0.004	0.05	**0.251**	0.038	0.219
	Norm. Cost	0.449	0.412	**0.253**	0.421	0.278

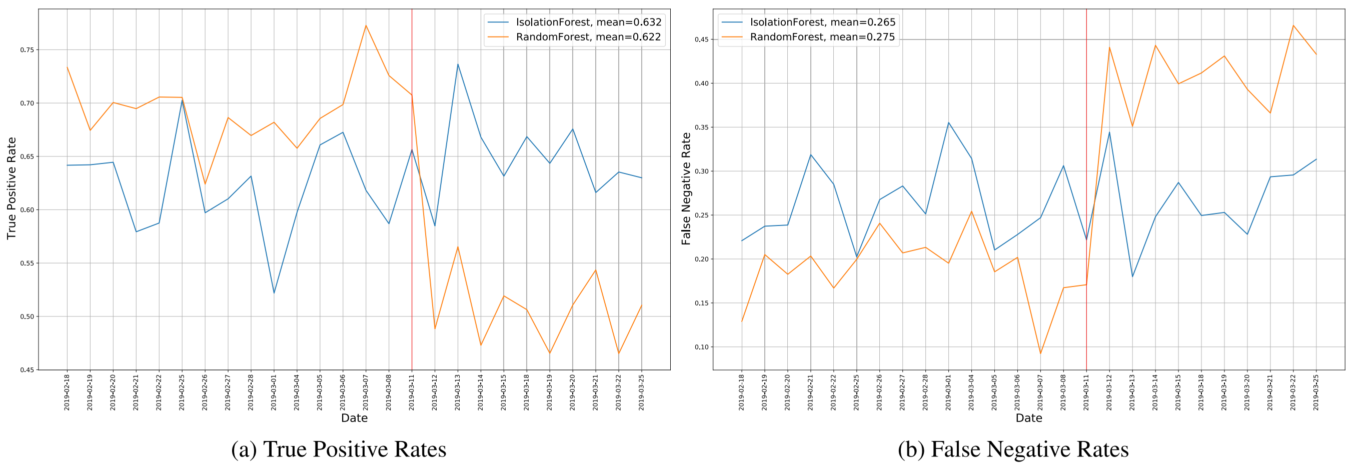
TABLE 4. Experiment 3—Supervised algorithms comparison. All algorithms show similar performances from the point of view of all metrics under analysis, with the exception of the cost, where Random Forest performs best.

	Random Forest	Cat Boost	Decision Tree	LGBM	SVM	Naive Bayes
TPR	**0.793**	0.781	0.771	0.752	0.3	0.135
FPR	0.001	0.001	0.002	0.001	**0**	0.03
FNR	**0.207**	0.219	0.229	0.248	0.7	0.865
Accuracy	**0.998**	0.998	0.996	0.997	0.994	0.963
Precision	0.899	**0.907**	0.77	0.879	0.855	0.036
FScore	**0.842**	0.838	0.77	0.809	0.44	0.056
MCC	**0.843**	0.84	0.768	0.811	0.501	0.054
Norm. Cost	**0.094**	0.099	0.104	0.112	0.314	0.405

training set and only introduced in the test set for evaluation purposes. During the run, the models are trained using the high-level vectors obtained from the training set that contains the remaining class of anomalies, excluding the withheld anomalies. After several iterations of the system (precisely after the 15th day of the experiment), the withheld pattern is introduced in the test data to assess the behaviors of the two models. The results of each run are then averaged on a daily basis and shown in Figure 6a and Figure 6b. As expected, the Isolation Forest model performance is consistent, while Random Forest exhibits a decay in performance when new anomalies are introduced. The TPR of the Random Forests model halved, while the False Negative Rate (FNR) tripled. This proves that Random Forest performs poorly at detecting the new anomalous patterns introduced in the dataset. On the other hand, the performance of the Isolation Forest is not negatively affected by the new anomalies, showing its capability of detecting the new anomalous pattern introduced.

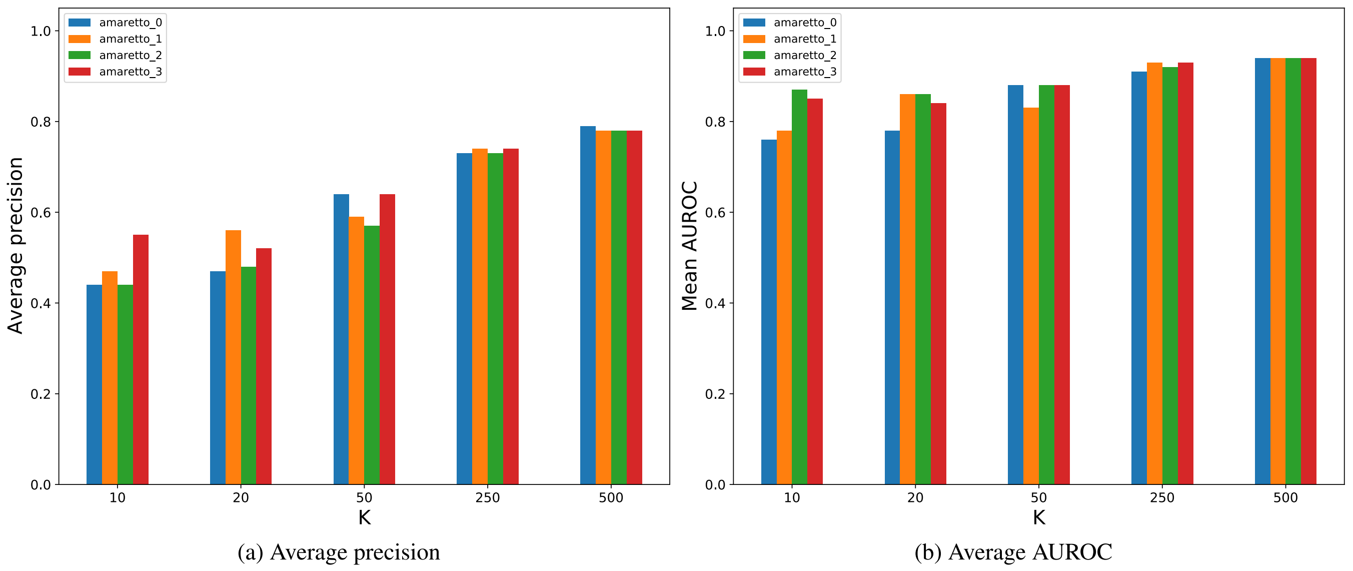
F. EXPERIMENT 5: AMARETTO CONFIGURATIONS

The purpose of this experiment is to test which of the strategies is the most suitable for our dataset. The experiment works as follows: on the first day, the labeled dataset is empty.



(a) True Positive Rates (b) False Negative Rates

FIGURE 6. Experiment 4—Performance of the supervised and unsupervised models in detecting novel anomalous patterns (i.e., not present in the training set) averaged on a daily basis. The Isolation Forest model (in blue) performance is consistent, while Random Forest (in orange) exhibits a decay in performance when new anomalies are introduced. The red line represent the moment in which new fraudulent patterns are introduced.



(a) Average precision (b) Average AUROC

FIGURE 7. Experiment 5—Performance comparison, varying K, of the four different Amaretto configurations (i.e., with different selection strategies. Overall, the performance are similar. Amaretto_3 (red bar) provides the best trade-off in performance and costs, especially for lower K, since it achieves a comparable detection performance but a lower amount of transactions to investigate.

Therefore the supervised model is not used. After the first day, the labeled dataset contains the samples selected by the Isolation Forest, which have been reviewed by a subject matter expert. From this point onwards, the entire selection strategy can be employed (first stage, second stage, and third stage). The mapping between the approaches adopted in the first and third stages, as well as the names of the configurations, are outlined in Table 5. Figure 7a and 7b show the average *Precision* and the average *AUROC* of the score generated by Random Forest. The performance of the 4 configurations is similar. We decided to focus on the system that employs *SELECT-DIVERSE* and *SELECT-ENTROPY* (*Amaretto _3*, red bar in the experiment 5 Figure 7b) because it provides the best overall average *Precision* and AUC, reducing the cost of manually reviewing multiple transactions and risk for a financial institution of not detecting illicit activities. In fact, with these two strategies selected, even with a daily budget of K=10, the average Precision is higher than the one obtained with K=20 and comparable with the one obtained with K=50.

G. EXPERIMENT 6: COMPARISON WITH THE STATE OF THE ART

In this final experiment, we compare Amaretto with AI^2 [37] since it represents, to the best of our knowledge, the state-of-the-art active-learning framework for anomaly detection. AI^2 comprises an ensemble of three unsupervised techniques, including a density-based model, using a *Copula-based multivariate* distribution, a matrix decomposition-based

TABLE 5. Experiment 5—Mapping between the selection strategies and Amaretto configurations. 1st and 3rd represents the stages in which such strategies are deployed.

Amaretto	TOP	ENTROPY	SELECT-DIVERSE	CONFLICT
0	1st	3rd		
1			1st	3rd
2	1st			3rd
3		3rd	1st	

model, using a *PCA-based* model, and an *Autoencoder*. The combination of the anomaly scores computed by the three models is used to rank the most anomalous high-level vectors for review by a subject matter expert. The feedback collected is then used to train a Random Forest that additionally analyzes the high-level vectors.

The experiment is divided into three parts: *I - Static Scenario*, *II - Real-world Scenario*, and *III - Real-world Scenario with Risk Profiles*. In the first part, we compare the frameworks in a static scenario, i.e., we collect 10 samples per day over a period of 10 days from each framework and then use this labeled data to train the Random Forest and predict all remaining high-level vectors. In the second part, we compare the frameworks in a real-world scenario, studying the effective support to the daily routine of a subject matter expert and the performances of the frameworks. In the third part, we compare the frameworks' performance in a real-world scenario by taking into account different risk profiles for a financial institution.

1) STATIC SCENARIO

The purpose of the first part of this experiment is to verify the framework's performance with a minimum amount of training data. During this part, we also assess the active learning inner modules, i.e., the components of the framework in charge of computing the *anomaly score* and selecting the samples to be shown to the subject matter expert. For the first 10 days of the test set, only the inner module is employed with a minimum daily budget ($K = 10$), collecting 100 samples. This labeled dataset is then used to train a Random Forest. Subsequently, the trained Random Forest model computes the *probability score* and the *prediction* for all the remaining high-level vectors. Figure 8a and Figure 8b show the comparison of our system against AI^2 using the *probability score*. As exhibited by the *ROC curve* plotted in Figure 8a, Amaretto achieves an *AUROC* of 0.93, whereas AI^2 obtained 0.89. Furthermore, as shown in the Precision/Recall curve in Figure 8b, Amaretto reaches an *average precision* of 0.61, which is 31% better than AI^2.

2) REAL-WORLD SCENARIO

In the second part of the experiment we assess how applying this framework can decrease the workload of the subject matter expert in a real-world scenario. Initially, only the unsupervised machine learning techniques could be employed since no feedback was collected. After the first day, the Random Forest works alongside the unsupervised models in the active learning loop and the prediction phase. For this test, we consider the worst-case scenario with a minimum daily budget of ($K = 10$). Figure 8c shows the *average precision* computed using the *probability score*. Amaretto doubled its Precision in approximately ten days, i.e., with a dataset of 100 high-level vectors, constantly increasing its performance. During the tests, Amaretto reaches a maximum average precision of 0.78, while AI^2 0.57. As shown in Figure 8d, Amaretto achieves better performances also considering the AUROC. As in the previous test, the maximum *AUROC* is 0.94, and the average *AUROC* is 0.847, improving AI^2 by circa 14%.

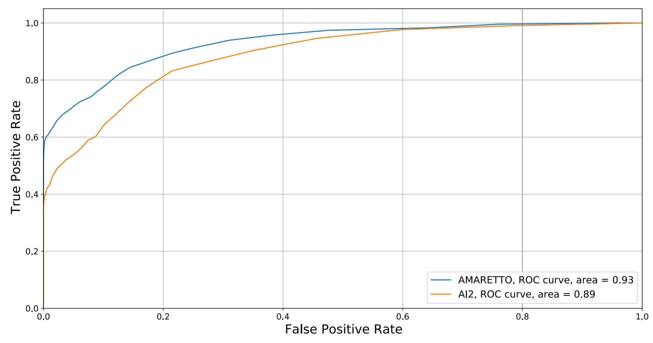
3) REAL-WORLD SCENARIO WITH RISK PROFILES

In the last part of the experiment, we test the frameworks considering different risk profiles that a financial institution can adopt. This is done considering different threshold values for the *probability score* corresponding to different use cases. For example, a lower threshold (corresponding to a lower anomaly probability score) could be used where high financial risk is estimated. By doing this, more transactions will be considered as candidates for review, hence reducing the false negatives but increasing false positives. As shown in Figure 8e, Figure 8f, Figure 8g, and Figure 8h, Amaretto outperforms AI^2 across all thresholds. In the low-risk use case, Amaretto achieves a TPR of 0.428%, while in the high-risk use case, a TPR of 0.596% represents an improvement of circa 50% w.r.t AI^2. Only in a low-risk case does AI2 achieve a better FPR than Amaretto. On the other hand, Amaretto achieves a higher TPR, a lower FNR, and cost, balancing the overall performance. In addition, in every scenario, the cost of Amaretto is always lower than AI^2.

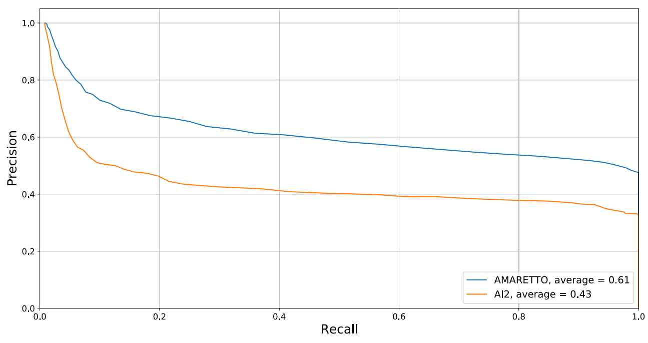
VII. LIMITATIONS AND FUTURE WORKS

The main limitation of this research work is the lack of an intuitive explanation for the anomaly score returned by Amaretto that is used to rank the high-level vectors. The two composing algorithms (i.e., Random Forest and Isolation Forest) are used in a black box fashion and function as rule-based trees whose prediction can be explained following the path that led to the given classification. However, since we are considering a "forest," we are averaging the output of several trees. Therefore, the resulting probability does not provide helpful insight and cannot help the human analyst better understand the result. Future work may focus on integrating into Amaretto a library and implementing the explainability processes. For example, SHAP [56] is one of the most interesting approaches for explaining the model output.

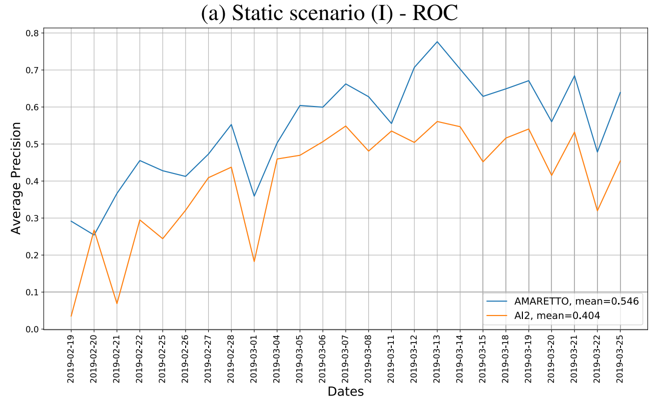
The available dataset represents another limitation. Even though it was synthetically forged from a real-world dataset, it is very limited in timespan, containing 60 days of transactions. A broader dataset would have allowed us to analyze the system's evolution over time deeply. For example,



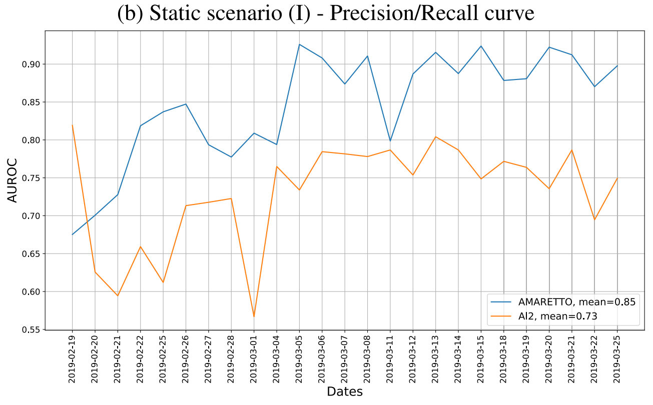
(a) Static scenario (I) - ROC



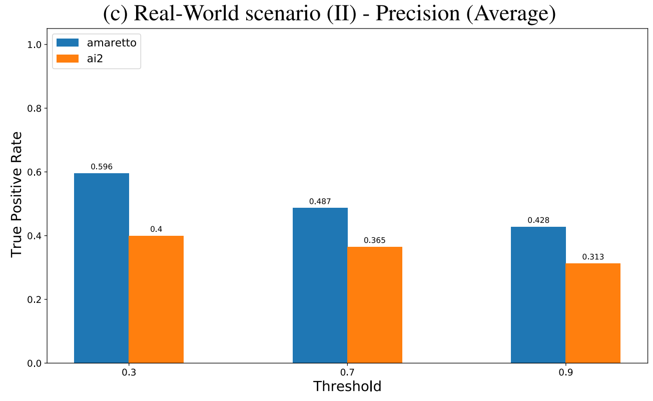
(b) Static scenario (I) - Precision/Recall curve



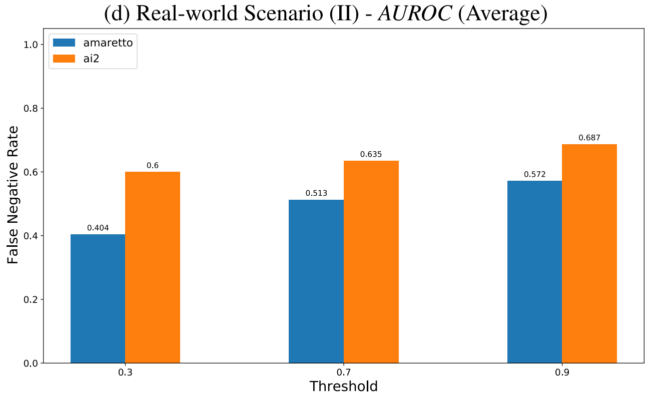
(c) Real-World scenario (II) - Precision (Average)



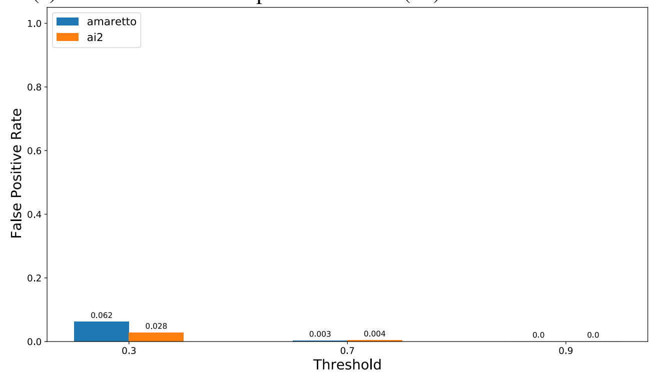
(d) Real-world Scenario (II) - *AUROC* (Average)



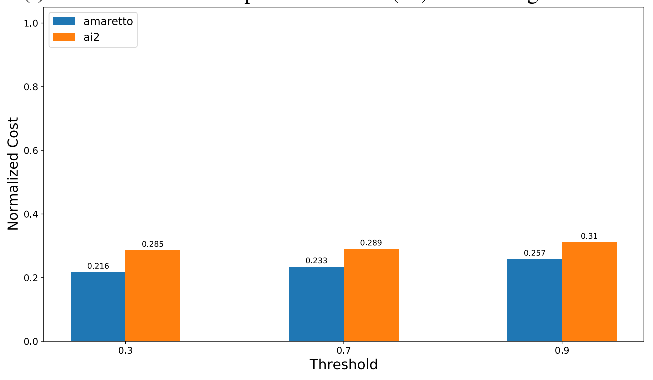
(e) Real-world with risk profile scenario (III) - True Positive Rate



(f) Real-world with risk profile scenario (III) - False Negative Rate



(g) Real-world with risk profile scenario (III) - False Positive Rate



(h) Real-world with risk profile scenario (III) - Norm. Cost

FIGURE 8. Experiment 6—Comparison of Amaretto (blue) versus AI^2 (orange) with $K = 10$ for the three experimental settings: *Scenario I–Static*, *Scenario II–Real-world*, and *Scenario III–Real-world with risk profile*. Amaretto outperforms AI^2 in almost all the metrics and scenarios under analysis at the cost of a slightly higher average FPR but lower overall cost.

seasonal models and models based on specific anomalous patterns could be developed.

It could be interesting to see how Amaretto works on more complex scenarios and further analyze the anomalies—for example, investigating the relationships among these anomalies or which anomalies occur more frequently. We assumed that the subject matter expert always provided correct labels when reviewing the data; further research should be conducted to assess the impact of incorrect labeling on the system and ensure that the models accommodate such errors. Another avenue for future research includes automatically tuning Amaretto parameters to improve performance and reduce the number of samples required to achieve satisfactory results. Furthermore, alternative unsupervised and supervised models should be tested.

In this work, we did not consider graph-based or deep learning models since they usually require higher computational requirements with respect to standard machine learning approaches. In fact, we aimed to build the simplest and lightest active learning system to outperform state-of-the-art solutions. In addition, even if the available dataset seems to offer enough data for applying such categories of algorithms, the experimental evaluation performed in this work seems to suggest the opposite. In fact, the deep learning model tested in this work (i.e., Autoencoder and Variational Autoencoders) does not perform well when evaluated in a real-world scenario in which an analyst can review only a limited number of samples. Future works may explore applying deep learning algorithms (e.g., models that overcome the shortcomings of random forests)to complement Amaretto on larger datasets that span over a greater period. For example, LSTM-based neural networks could be employed due to the temporal correlations in money laundering patterns.

Finally, it would be interesting to evaluate Amaretto on detecting other kinds of financial crime, like credit card fraud detection, to investigate its flexibility.

VIII. CONCLUSION

In this paper, we presented Amaretto, an active learning system for anomaly detection applied to transaction monitoring for money laundering detection in capital markets. Amaretto comprises an unsupervised model for detecting known and unknown anomalous patterns, including four strategies to optimally sample the data for a subject matter expert to review. This data is fed into a supervised learning model to continuously improve the system's performance. Amaretto was able to process over 29 million transactions, extracting aggregated features and highlighting customer behavioral patterns over time to detect unusual correlations. We then presented the experimental results conducted on a synthetic dataset generated to resemble genuine, as well as potential money laundering patterns. We compared unsupervised techniques, commonly used in anomaly detection tasks and state-of-the-art solutions. We demonstrated that Isolation Forest is the best algorithm in the AML domain. we also compared supervised techniques, and we determined that Random Forest outperforms the others. Subsequently, we proved how important it is for the unsupervised component of the system to detect novel patterns, since the supervised models could not accurately detect these. Finally, we conducted experiments to confirm the robustness of our design in a real-world scenario. We demonstrated the best selection strategies amongst the ones proposed and proved that Amaretto achieved state-of-the-art performance within a short time frame and with minimal manual input from a subject matter expert.

REFERENCES

[1] United Nations Office on Drugs and Crime. (2011). *Estimating Illicit Financial Flows Resulting From Drug Trafficking and Other Transnational Organized Crimes*. [Online]. Available: https://www.unodc.org/documents/data-and-analysis/Studies/Illicit_financial_flows_2011_web.pdf

[2] T. L. V. Barnett, *Outliers in Statistical Data*, 3rd ed. Hoboken, NJ, USA: Wiley, 1994.

[3] FATF. (2018). *Guidance for a Risk-Based Approach: Securities Sector*. [Online]. Available: https://www.fatf-gafi.org/media/fatf/documents/recommendations/pdfs/RBA-Securities-Sector.pdf

[4] M. Tiwari, A. Gepp, and K. Kumar, "A review of money laundering literature: The state of research in key areas," *Pacific Accounting Rev.*, vol. 32, no. 2, pp. 271–303, Mar. 2020.

[5] Z. Chen, L. D. V. Khoa, E. N. Teoh, A. Nazir, E. K. Karuppiah, and K. S. Lam, "Machine learning techniques for anti-money laundering (AML) solutions in suspicious transaction detection: A review," *Knowl. Inf. Syst.*, vol. 57, no. 2, pp. 245–285, 2018, doi: 10.1007/s10115-017-1144-z.

[6] N. M. Labib, M. A. Rizka, and A. E. M. Shokry, "Survey of machine learning approaches of anti-money laundering techniques to counter terrorism finance," in *Internet of Things—Applications and Future*. Singapore: Springer, 2020, pp. 73–87.

[7] A. A. S. Alsuwailem and A. K. J. Saudagar, "Anti-money laundering systems: A systematic literature review," *J. Money Laundering Control*, vol. 23, no. 4, pp. 833–848, May 2020.

[8] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: A survey," *ACM Comput. Surv.*, vol. 41, no. 3, pp. 15:1–15:58, Jul. 2009, doi: 10.1145/1541880.1541882.

[9] E. W. T. Ngai, Y. Hu, Y. H. Wong, Y. Chen, and X. Sun, "The application of data mining techniques in financial fraud detection: A classification framework and an academic review of literature," *Decision Support Syst.*, vol. 50, no. 3, pp. 559–569, 2011, doi: 10.1016/j.dss.2010.08.006.

[10] J. West and M. Bhattacharya, "Intelligent financial fraud detection: A comprehensive review," *Comput. Secur.*, vol. 57, pp. 47–66, Mar. 2016, doi: 10.1016/j.cose.2015.09.005.

[11] D. Yue, X. Wu, Y. Wang, Y. Li, and C.-H. Chu, "A review of data mining-based financial fraud detection research," in *Proc. Int. Conf. Wireless Commun., Netw. Mobile Comput.*, Sep. 2007, pp. 5519–5522.

[12] D. V. Kute, B. Pradhan, N. Shukla, and A. Alamri, "Deep learning and explainable artificial intelligence techniques applied for detecting money laundering—A critical review," *IEEE Access*, vol. 9, pp. 82300–82317, 2021, doi: 10.1109/ACCESS.2021.3086230.

[13] S. Ramaswamy, R. Rastogi, and K. Shim, "Efficient algorithms for mining outliers from large data sets," in *Proc. ACM SIGMOD Int. Conf. Manage. Data*, Dallas, TX, USA, W. Chen, J. F. Naughton, and P. A. Bernstein, Eds., May 2000, pp. 427–438, doi: 10.1145/342009.335437.

[14] G. Williams, R. Baxter, H. He, S. Hawkins, and L. Gu, "A comparative study of RNN for outlier detection in data mining," in *Proc. ICDM*, 2002, p. 709.

[15] M. Carminati, R. Caron, F. Maggi, I. Epifani, and S. Zanero, "BankSealer: A decision support system for online banking fraud analysis and investigation," *Comput. Secur.*, vol. 53, pp. 175–186, Sep. 2015, doi: 10.1016/j.cose.2015.04.002.

[16] M. Carminati, R. Caron, F. Maggi, I. Epifani, and S. Zanero, "BANKSEALER: An online banking fraud analysis and decision support system," in *ICT Systems Security and Privacy Protection* (IFIP Advances in Information and Communication Technology), vol. 428, N. Cuppens-Boulahia, F. Cuppens, S. Jajodia, A. A. E. Kalam, and T. Sans, Eds. Marrakech, Morocco: Springer, Jun. 2014, pp. 380–394, doi: 10.1007/978-3-642-55415-5_32.

[17] M. Carminati, A. Baggio, F. Maggi, U. Spagnolini, and S. Zanero, ''Fraudbuster: Temporal analysis and detection of advanced financial frauds,'' in *Detection of Intrusions and Malware, and Vulnerability Assessment* (Lecture Notes in Computer Science), vol. 10885, C. Giuffrida, S. Bardin, and G. Blanc, Eds. Saclay, France: Springer, Jun. 2018, pp. 211–233, doi: 10.1007/978-3-319-93411-2_10.

[18] M. Carminati, M. Polino, A. Continella, A. Lanzi, F. Maggi, and S. Zanero, ''Security evaluation of a banking fraud analysis system,'' *ACM Trans. Priv. Secur.*, vol. 21, no. 3, pp. 11:1–11:31, 2018, doi: 10.1145/3178370.

[19] M. Carminati, L. Santini, M. Polino, and S. Zanero, ''Evasion attacks against banking fraud detection systems,'' in *Proc. 23rd Int. Symp. Res. Attacks, Intrusions Defenses (RAID)*, M. Egele and L. Bilge, Eds. San Sebastian, Spain: USENIX Association, Oct. 2020, pp. 285–300. [Online]. Available: https://www.usenix.org/conference/raid2020/presentation/carminati

[20] J. Guevara, O. Garcia-Bedoya, and O. M. Granados, ''Machine learning methodologies against money laundering in non-banking correspondents,'' in *Applied Informatics* (Communications in Computer and Information Science), vol. 1277, H. Florez and S. Misra, Eds. Ota, Nigeria: Springer, Oct. 2020, pp. 72–88, doi: 10.1007/978-3-030-61702-8_6.

[21] A. E. M. Shokry, M. A. Rizka, and N. M. Labib, ''Counter terrorism finance by detecting money laundering hidden networks using unsupervised machine learning algorithm,'' in *Proc. Int. Conf. ICT, Soc., Hum. Beings*, 2020, pp. 89–97.

[22] N. A. L. Khac and M.-T. Kechadi, ''Application of data mining for anti-money laundering detection: A case study,'' in *Proc. 10th IEEE Int. Conf. Data Mining Workshops (ICDMW)*, W. Fan, W. Hsu, G. I. Webb, B. Liu, C. Zhang, D. Gunopulos, and X. Wu, Eds. Sydney, NSW, Australia: IEEE Computer Society, Dec. 2010, pp. 577–584, doi: 10.1109/ICDMW.2010.66.

[23] N. A. L. Khac, S. Markos, and M.-T. Kechadi, ''A data mining-based solution for detecting suspicious money laundering cases in an investment bank,'' in *Proc. 2nd Int. Conf. Adv. Databases, Knowl., Data Appl.*, F. Laux and L. Strömbäck, Eds. Menuires, France: IEEE Computer Society, Apr. 2010, pp. 235–240, doi: 10.1109/DBKDA.2010.27.

[24] R. A. L. Torres and M. Ladeira, ''A proposal for online analysis and identification of fraudulent financial transactions,'' in *Proc. 19th IEEE Int. Conf. Mach. Learn. Appl. (ICMLA)*, M. A. Wani, F. Luo, X. A. Li, D. Dou, and F. Bonchi, Eds., Miami, FL, USA, Dec. 2020, pp. 240–245, doi: 10.1109/ICMLA51294.2020.00047.

[25] E. L. Paula, M. Ladeira, R. N. Carvalho, and T. Marzagao, ''Deep learning anomaly detection as support fraud investigation in Brazilian exports and anti-money laundering,'' in *Proc. 15th IEEE Int. Conf. Mach. Learn. Appl. (ICMLA)*. Anaheim, CA, USA: IEEE Computer Society, Dec. 2016, pp. 954–960, doi: 10.1109/ICMLA.2016.0172.

[26] A. D. Pozzolo and G. Bontempi, ''Adaptive machine learning for credit card fraud detection,'' Ph.D. dissertation, Dept. Comput. Sci., Mach. Learn. Group, Université Libre de Bruxelles, Brussels, Belgium, 2015. [Online]. Available: http://di.ulb.ac.be/map/adalpozz/pdf/Dalpozzolo2015PhD.pdf

[27] S. Bhattacharyya, S. Jha, K. Tharakunnel, and J. C. Westland, ''Data mining for credit card fraud: A comparative study,'' *Decision Support Syst.*, vol. 50, no. 3, pp. 602–613, 2011, doi: 10.1016/j.dss.2010.08.008.

[28] T. E. Senator, H. G. Goldberg, J. Wooton, M. A. Cottini, A. F. U. Khan, C. D. Klinger, W. M. Llamas, M. P. Marrone, and R. W. H. Wong, ''Financial crimes enforcement network AI system (FAIS) identifying potential money laundering from reports of large cash transactions,'' *AI Mag.*, vol. 16, no. 4, pp. 21–39, 1995, doi: 10.1609/aimag.v16i4.1169.

[29] O. Garcia-Bedoya, O. Granados, and J. C. Burgos, ''AI against money laundering networks: The Colombian case,'' *J. Money Laundering Control*, vol. 24, no. 1, pp. 49–62, May 2021.

[30] M. Jullum, A. Løland, R. B. Huseby, G. Ånonsen, and J. Lorentzen, ''Detecting money laundering transactions with machine learning,'' *J. Money Laundering Control*, vol. 23, no. 1, pp. 173–186, Jan. 2020.

[31] P. Tertychnyi, I. Slobozhan, M. Ollikainen, and M. Dumas, ''Scalable and imbalance-resistant machine learning models for anti-money laundering: A two-layered approach,'' in *Enterprise Applications, Markets and Services in the Finance Industry* (Lecture Notes in Business Information Processing), vol. 401, B. Clapham and J. Koch, Eds. Helsinki, Finland: Springer, 2020, pp. 43–58, doi: 10.1007/978-3-030-64466-6_3.

[32] S. Farrugia, J. Ellul, and G. Azzopardi, ''Detection of illicit accounts over the Ethereum blockchain,'' *Expert Syst. Appl.*, vol. 150, Jul. 2020, Art. no. 113318, doi: 10.1016/j.eswa.2020.113318.

[33] D. Vassallo, V. Vella, and J. Ellul, ''Application of gradient boosting algorithms for anti-money laundering in cryptocurrencies,'' *Social Netw. Comput. Sci.*, vol. 2, no. 3, p. 143, May 2021, doi: 10.1007/s42979-021-00558-z.

[34] M. Alkhalili, M. H. Qutqut, and F. Almasalha, ''Investigation of applying machine learning for watch-list filtering in anti-money laundering,'' *IEEE Access*, vol. 9, pp. 18481–18496, 2021, doi: 10.1109/ACCESS.2021.3052313.

[35] N. Göernitz, M. M. Kloft, K. Rieck, and U. Brefeld, ''Toward supervised anomaly detection,'' 2014, *arXiv:1401.6424*.

[36] M. Carminati, L. Valentini, and S. Zanero, ''A supervised auto-tuning approach for a banking fraud detection system,'' in *Cyber Security Cryptography and Machine Learning* (Lecture Notes in Computer Science), vol. 10332, S. Dolev and S. Lodha, Eds. Be'er Sheva, Israel: Springer, Jun. 2017, pp. 215–233, doi: 10.1007/978-3-319-60080-2_17.

[37] K. Veeramachaneni, I. Arnaldo, V. Korrapati, C. Bassias, and K. Li, ''Ai^2: Training a big data machine to defend,'' in *Proc. 2nd IEEE Int. Conf. Big Data Secur. Cloud (BigDataSecurity), IEEE Int. Conf. High Perform. Smart Comput. (HPSC), IEEE Int. Conf. Intell. Data Secur. (IDS)*, New York, NY, USA, Apr. 2016, pp. 49–54, doi: 10.1109/BigDataSecurity-HPSC-IDS.2016.79.

[38] S. Das, M. R. Islam, N. K. Jayakodi, and J. R. Doppa, ''Active anomaly detection via ensembles: Insights, algorithms, and interpretability,'' 2019, *arXiv:1901.08930*.

[39] I. Alarab, S. Prakoonwit, and M. I. Nacer, ''Competence of graph convolutional networks for anti-money laundering in bitcoin blockchain,'' in *Proc. 5th Int. Conf. Mach. Learn. Technol.*, Jun. 2020, pp. 23–27.

[40] L. Akoglu, H. Tong, and D. Koutra, ''Graph based anomaly detection and description: A survey,'' *Data Mining Knowl. Discovery*, vol. 29, no. 3, pp. 626–688, 2015, doi: 10.1007/s10618-014-0365-y.

[41] W. Eberle and L. Holder, ''Anomaly detection in data represented as graphs,'' *Intell. Data Anal.*, vol. 11, no. 6, pp. 663–689, 2007. [Online]. Available: http://content.iospress.com/articles/intelligent-data-analysis/ida00309

[42] C. C. Noble and D. J. Cook, ''Graph-based anomaly detection,'' in *Proc. 9th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, L. Getoor, T. E. Senator, P. M. Domingos, and C. Faloutsos, Eds., Washington, DC, USA, Aug. 2003, pp. 631–636, doi: 10.1145/956750.956831.

[43] H. S. Seung, H. Sompolinsky, and N. Tishby, ''Statistical mechanics of learning from examples,'' *Phys. Rev. A, Gen. Phys.*, vol. 45, pp. 6056–6091, Apr. 1992. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevA.45.6056

[44] J. R. Quinlan, ''Induction of decision trees,'' *Mach. Learn.*, vol. 1, no. 1, pp. 81–106, 1986, doi: 10.1023/A:1022643204877.

[45] L. Breiman, ''Random forests,'' *Mach. Learn.*, vol. 45, no. 1, pp. 5–32, 2001, doi: 10.1023/A:1010933404324.

[46] R. J. G. B. Campello, D. Moulavi, and J. Sander, ''Density-based clustering based on hierarchical density estimates,'' in *Advances in Knowledge Discovery and Data Mining* (Lecture Notes in Computer Science), vol. 7819, J. Pei, V. S. Tseng, L. Cao, H. Motoda, and G. Xu, Eds. Gold Coast, QLD, Australia: Springer, Apr. 2013, pp. 160–172, doi: 10.1007/978-3-642-37456-2_14.

[47] L. McInnes and J. Healy, ''Accelerated hierarchical density based clustering,'' in *Proc. IEEE Int. Conf. Data Mining Workshops (ICDMW)*, Nov. 2017, pp. 33–42, doi: 10.1109/ICDMW.2017.12.

[48] I. Kaastra and M. Boyd, ''Designing a neural network for forecasting financial and economic time series,'' *Neurocomputing*, vol. 10, no. 3, pp. 215–236, 1996, doi: 10.1016/0925-2312(95)00039-9.

[49] J. Snoek, H. Larochelle, and R. P. Adams, ''Practical Bayesian optimization of machine learning algorithms,'' in *Proc. 26th Annu. Conf. Neural Inf. Process. Syst.*, P. L. Bartlett, F. C. N. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, Eds., Lake Tahoe, NV, USA, 2012, pp. 2960–2968. [Online]. Available: https://proceedings.neurips.cc/paper/2012/hash/05311655a15b75fab86956663e1819cd-Abstract.html

[50] C. Whitrow, D. J. Hand, P. Juszczak, D. J. Weston, and N. M. Adams, ''Transaction aggregation as a strategy for credit card fraud detection,'' *Data Mining Knowl. Discovery*, vol. 18, no. 1, pp. 30–55, 2009, doi: 10.1007/s10618-008-0116-z.

[51] D. P. Kingma and M. Welling, ''Auto-encoding variational Bayes,'' in *Proc. 2nd Int. Conf. Learn. Represent. (ICLR)*, Y. Bengio and Y. LeCun, Eds., Banff, AB, Canada, Apr. 2014, pp. 1–14.

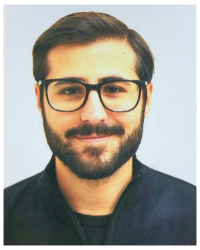
[52] S. Hariri, M. C. Kind, and R. J. Brunner, ''Extended isolation forest,'' *IEEE Trans. Knowl. Data Eng.*, vol. 33, no. 4, pp. 1479–1489, Apr. 2021, doi: 10.1109/TKDE.2019.2947676.

[53] M. L. Shyu, S. C. Chen, K. Sarinnapakorn, and L. Chang, "A novel anomaly detection scheme based on principal component classifier," in *Proc. IEEE Found. New Directions Data Mining Workshop, Conjunction 3rd IEEE Int. Conf. Data Mining (ICDM*, Nov. 2003, pp. 172–179.

[54] T. Chen and C. Guestrin, "XGBoost: A scalable tree boosting system," in *Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Aug. 2016, pp. 785–794, doi: 10.1145/2939672.2939785.

[55] S. M. Lundberg and S. Lee, "A unified approach to interpreting model predictions," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 4765–4774. [Online]. Available: https://proceedings.neurips.cc/paper/2017/hash/8a20a8621978632d76c43dfd28b67767-Abstract.html

[56] S. M. Lundberg and S.-I. Lee, "A unified approach to interpreting model predictions," in *Advances in Neural Information Processing Systems*, vol. 30, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds. Red Hook, NY, USA: Curran Associates, 2017, pp. 4765–4774. [Online]. Available: http://papers.nips.cc/paper/7062-a-unified-approach-to-interpreting-model-predictions.pdf



DANILO LABANCA received the master's degree in computer security and machine learning from the Politecnico di Milano, in 2019. He was a master student with the Politecnico di Milano during the research. Currently, he works as a Data Scientist with the AML and Compliance Department, Paysafe, a financial institution that deals with online payments.

LUCA PRIMERANO received the degree and the M.Sc. degree in telecommunications engineering from the Politecnico di Milano. He was, during the research, the Chief AI Officer and the Founder of Napier Technologies Ltd., a London-Based anti-money laundering software company. He has extensive technology experience that he gained at Goldman Sachs, Deutsche Bank, and Deloitte.

MARCUS MARKLAND-MONTGOMERY received the M.Sc. degree in statistics from Imperial College London. He currently works with Napier Technologies Ltd., applying data science to AML and trade compliance problems. In particular cross-lingual client screening and using active learning to detect money laundering patterns.

MARIO POLINO received the Ph.D. degree in computer security from the Politecnico di Milano, in 2017. His Ph.D. was mainly based on malware analysis and automation of behavior extraction. Besides malware analysis, he is interested in several aspects of computer security and his works range from the study of cyber-physical systems to banking fraud detection going through Android security. He is also interested in binary analysis and exploitation techniques. Currently, he is an Assistant Professor with the Politecnico di Milano. He spends most of his free time playing Capture the Flag Competition with "Tower fo Hanoi" and "mHACKeroni."

MICHELE CARMINATI received the Ph.D. degree in information technology from the Politecnico di Milano, Italy. He is currently an Assistant Professor (RTDa) working as part of the Cybersecurity Group, Politecnico di Milano. His research interests include the application of machine learning methods in various security-related fields, ranging from cyber-physical and automotive systems to binary analysis (going through anomaly and intrusion detection). In particular, he is actively involved in the financial fraud detection domain, where he works on the analysis of advanced financial threats, such as banking Trojans and on the design of novel frameworks to identify and timely detect fraudulent transactions.

STEFANO ZANERO (Senior Member, IEEE) received the Ph.D. degree in computer engineering from the Politecnico di Milano. He is currently an Associate Professor with the Dipartimento di Elettronica, Informazione e Bioingegneria, Politecnico di Milano. His research interests include malware analysis, cyber-physical security, and cybersecurity in general. He is on the Board of Governors of the IEEE Computer Society. He is a Lifetime Senior Member of the ACM and has been named a fellow of the Information System Security Association (ISSA).