Amaretto: An Active Learning Framework for Money Laundering Detection

Danilo Labanca\textsuperscript{a}, Luca Primerano\textsuperscript{b}, Marcus Markland-Montgomery\textsuperscript{b}, Mario Polino\textsuperscript{a}, Michele Carminati\textsuperscript{a,}\textsuperscript{*}, Stefano Zanero\textsuperscript{a}

\textsuperscript{a}Dipartimento di Elettronica, Informazione e Bioingegneria, Politecnico di Milano, Milan
\textsuperscript{b}Napier Technologies Limited, London

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 been successfully used to complement traditional rule-based systems. This is done to reduce the high number of false positives and the effort needed to manually review all the generated alerts. Unfortunately, these solutions also have disadvantages: unsupervised models lead to low performance, resulting in a high number of false positives, while supervised models require a large amount of labeled data to achieve a high detection rate.

In this paper, we present Amaretto, an active learning framework for money laundering detection that combines supervised and unsupervised learning techniques to take advantage of their strengths. Amaretto improves transaction monitoring processes by targeting a subset of transactions for investigation and making more efficient use of the feedback provided by the analyst. We evaluated Amaretto on a synthetic dataset simulating a real-world scenario. We show that our approach outperforms the state-of-the-art solutions reducing money laundering risk and improving the detection performance of the system.

Keywords: Active Learning, Anti-Money Laundering, Supervised Learning, Unsupervised Learning, Financial Systems

1. 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 that may be 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 to monitor pre-determined unusual behaviors. The expert system generates alerts if rules are triggered (e.g., if \textit{amount} > 10,000,000 then \textit{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 and 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 can not 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 traditional AML techniques by overcoming some of the pitfalls in rule-based systems. Machine learning models are not limited to a set of pre-defined rules. Instead, they can extract and analyze patterns and insights from data and assess unusual correlations that may be 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 \textit{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 of 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, the selection strategy is applied to choose the samples that will be reviewed by the analyst. Finally, the labels collected from the analyst are used as training data for the su-
pervised model that will compute the final risk score for the aggregated vectors - then the cycle repeats.

Finally, we present the experimental evaluation conducted on a synthetic dataset provided by the industry partners we collaborated with. The dataset includes both normal transactions as well as anomalous patterns that may be linked to potential money laundering activities; the dataset was 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. Then, we conduct a similar assessment amongst supervised techniques concluding that Random Forest outperforms the others. Subsequently, we prove the contributions made by an unsupervised model complement the ability of supervised models by being able to detect new types of anomalies. In the end, we conduct these experiments to confirm the robustness of our design in a real-world scenario, identifying the best selection strategies from the ones proposed and finally showing that Amaretto outperforms the state-of-the-art solutions by improving the detection rate and reducing the fraud management costs for financial institutions. To allow the reproducibility of the results, we will release the synthetic dataset of transactions at url.to.be.released In summary, the contributions are the following:

- A novel selection strategy for an active learning framework that detects potential money laundering patterns. This strategy takes into account event diversity and prioritizes new anomalous patterns to improve the quality of the knowledge base and training set.
- The comparison of supervised and unsupervised algorithms for detecting potential money laundering, including detailed benchmarking.
- 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.

2. Background and Challenges

Detecting money laundering can be considered as one of the most challenging tasks within anomaly detection for several reasons. First of all, there is no a 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. The focus of this paper is on detecting unusual transactional activities that may be linked to money laundering occurring in capital markets. Unusual transactions and customer behaviors are considered outliers that may have an associated 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 transaction 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, whilst fraud mostly occurs at a transactional level. In an ideal world, where there is an unlimited amount of 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 that have a high risk of money laundering.

One of the key challenges around researching novel approaches to detecting money laundering is the lack of standardized and available datasets. Financial institutions rarely share data due to confidentiality reasons and due to 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].

3. Related Works

Anomaly detection is a widely researched topic. In this section, we focus on money laundering detection, but we also highlight the fraud detection context that shares many commonalities with it. For a comprehensive overview of anomaly detection research, we refer the reader to [4, 5, 6, 7].

**Unsupervised learning.** Unsupervised learning is mainly used to detect unusual correlations and 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 [8], the authors directed their work along this way, proposing a formulation for outliers, according to the distance of a point from its neighbors. In [9], the authors proved 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.

Another work concerning fraud detection is BankSealer [10, 11], where the authors work in a semi-supervised setting, extracting local, global, and temporal profiles for each user to capture their behaviors. Then, they use various techniques in combination, such as Histogram-based Outlier Score (HBOS) [12] and Density-based Spatial Clustering of Applications with Noise (DBSCAN) [13], to rank new transactions. In [14] the security and robustness of BankSealer is evaluated against mimicry attacks. In a subsequent work [15], the temporal profiles is improved with the application of signal processing techniques to exploit
the end user’s recurrent vs. non-recurrent spending pattern. Regarding money laundering, the authors of [16] and [17] 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 extremely specific to the particular problem and dataset: the entire learning process is carried out on just two high-level features derived from the raw data. This seems to offer a limited perspective on the complexity of the underlying behaviors. In [18], the authors address the problem of money laundering in Brazilian exports using a type of Replicator Neural Network called Auto Encoder, which is considered a state-of-the-art machine learning algorithm for anomaly detection.

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 [19]. This is an issue for institutions, as false positives being reviewed 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 being missed.

Supervised learning. Supervised techniques are reviewed by [5], [6], and [7]; the most commonly used are Linear Regression, Bayesian Networks, Decision Trees, Neural Networks. Additionally, Random Forest and Support Vector Machines have started to gain more attention. In [20], the authors 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) [21]. This system allows the analyst to follow evidence left by linked transactions in a sequential manner 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.

Supervised learning can utilize manually reviewed transactions (i.e., labeled data) and generally outperform unsupervised learning in anomaly detection and classification tasks [22]. However, a large amount of labeled data is required to ensure adequate performance is achieved. Additionally, supervised learning is not as effective at detecting new anomalous patterns (resulting in false negatives) when compared to unsupervised learning. This is where active learning plays an important role in bridging unsupervised and supervised anomaly detection.

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 mat-
ter expert for the label of a transaction or group of transactions (suspicious or
genuine). This model is used to select which transactions the subject matter ex-
pert should investigate to minimize manual data reviews and, at the same time,
sure the output of the overall anomaly detection system is improved. In [23],
the authors exploit analyst feedback to self-tune and improve BankSealer’s de-
tection performance by means of a multi-objective genetic algorithm. In [24],
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 [25], the
authors point out the importance of selecting different types of anomalies to en-
hance active learning frameworks (i.e., selecting different classes of anomalies).

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.
Amaretto also optimizes the selection strategy in order to spot new anomalous
patterns and improve the detection rate of the system.

Graph analysis. Another approach commonly applied in fraud and money
laundering detection is the analysis of graphs, which is out of scope for Amaretto.
We refer the reader to [26, 27, 28] for a review of the research regarding graph-
based anomaly detection methods in fraud detection, intrusion detection, telecom-
munication, and opinion networks.

4. 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 be able to use a labelled dataset. Therefore,
we generated a synthetic dataset that simulate transaction profiles of clients
transacting in international capital markets. The data was generated by our
industry partner using a custom-built data generator, that combined more than
10,000 parameters extrapolated from real market data. The dataset consists of
29,704,090 transactions executed by 400 end-clients that are 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% the transactions. 98.43% of the
transactions have an amount that is 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 done during the early hours of the morning and at the
end of the day. Table 1 shows a summary of the statistics of the dataset. Key
Table 1: Input Data Characteristics: Number of transactions, anomalies, attributes, and ratio of anomalies over the number of transactions.

<table>
<thead>
<tr>
<th>Sample size</th>
<th>Transactions</th>
<th>Anomalies</th>
<th>Ratio</th>
<th>Attributes</th>
<th>Originators</th>
</tr>
</thead>
<tbody>
<tr>
<td>29,704,090</td>
<td>29,622,822</td>
<td>81,268</td>
<td>0.27 %</td>
<td>12</td>
<td>400</td>
</tr>
</tbody>
</table>

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.

4.1. Synthetic Anomalies Overview

Financial datasets are known to be extremely unbalanced, usually containing from 0.1% to 1% \[14\] of anomalous transactions. This information was also confirmed by 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 5 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 in the dataset.

**Small but highly frequent transactions generated within a short time-frame**: 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 previous transactional activity and is absent 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.

5. *Amaretto* Approach

To enable the detection of money laundering through supervised models, there is a requirement for sufficient labeled data. The only way to have re-
liable 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 [29]. This consists of using both unsupervised and supervised techniques combined 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. In parallel, the supervised models are trained on the domain experts’ feedback (i.e., labeled dataset), then select additional data points for review.

5.1. 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 pseudo-code for the key steps of *Amaretto*.

**Algorithm 1: Amaretto**

**Input:** \( L = \emptyset, U, mod_{sup}, mod_{unsup}, strat_{sup}, strat_{unsup}, K, T \)

1. for \( t = 0, \ldots, T \) do
2.   if \( t = 0 \) then
3.     Train \( mod_{unsup} \) on \( U^{t-1} \)
4.     Compute the scores \( S(x_i) \) where \( x_i \in U^t \)
5.     Query \( K \) samples from \( U^t \) using the sampling strategy \( strat_{unsup} \)
6.     Collect analyst feedback. Call it \( sample_{unsup}^t \)
7.     Add the selected points to \( L^{t-1} \) i.e. \( L^t = L^{t-1} \cup (x_i, y_i) \in sample_{unsup}^t \)
8. else if \( t > 0 \) then
9.     Train \( mod_{unsup} \) on \( U^{t-1} \)
10.    Train \( mod_{sup} \) on \( L^{t-1} \)
11.    Compute the scores \( S(x_i) \) where \( x_i \in U^t \)
12.    Query \( \frac{K}{2} \) samples from \( U^t \) using the sampling strategy \( strat_{unsup} \)
13.    Collect analyst feedback. Call it \( sample_{unsup}^t \)
14.    \( U^t = U_t \setminus sample_{unsup}^t \)
15.    Select \( \frac{K}{2} \) samples from \( U \) using the sampling strategy \( strat_{sup} \)
16.    Collect analyst feedback. Call it \( sample_{sup}^t \)
17.    Add the selected points to \( L^{t-1} \) i.e. \( L^t = L^{t-1} \cup sample_{unsup}^t \cup sample_{sup}^t \)

5.2. 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 [15, 24]: 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 in that day. Also, aggregating transactions over a period of time is useful in the AML use case since it can be used to capture correlations over time across multiple transactions. Within each time period, 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. For each window in which the user performed at least one transaction, a dataframe is created, collecting all the activities of the customer. 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 comprises a combination between Currency, Product Class, Product Type, and Input/Output 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 a fraudulent 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. Afterwards, 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 together 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.

### Table 2: High-level Vector Example

<table>
<thead>
<tr>
<th>Feature name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Originator</td>
<td>Client_304</td>
</tr>
<tr>
<td>EntryDate</td>
<td>2019-01-01 00:00:00</td>
</tr>
<tr>
<td>Weekday</td>
<td>1</td>
</tr>
<tr>
<td>Hour</td>
<td>7</td>
</tr>
<tr>
<td>Night</td>
<td>1</td>
</tr>
<tr>
<td>Morning</td>
<td>0</td>
</tr>
<tr>
<td>Evening</td>
<td>0</td>
</tr>
<tr>
<td>Anomaly</td>
<td>1</td>
</tr>
<tr>
<td>Transactions_count</td>
<td>73</td>
</tr>
<tr>
<td>Normalized Amount _sum</td>
<td>20886918.32900001</td>
</tr>
<tr>
<td>Normalized Amount _Small_Amount</td>
<td>15.0</td>
</tr>
<tr>
<td>Normalized Amount _Round_Amount</td>
<td>0.0</td>
</tr>
<tr>
<td>InputOutput_delta</td>
<td>17</td>
</tr>
<tr>
<td>Collateral_delta</td>
<td>-39</td>
</tr>
<tr>
<td>Normalized Amount _sum_Buy_Collateral_Cash_Currency</td>
<td>34888.65</td>
</tr>
<tr>
<td>Normalized Amount _sum_Sell_Collateral_Cash_Currency</td>
<td>16782.43</td>
</tr>
<tr>
<td>Normalized Amount _sum_Sell_Collateral_Security_Currency</td>
<td>11629.5568800000001</td>
</tr>
<tr>
<td>Normalized Amount _mean_Buy_Collateral_Cash_Currency</td>
<td>16782.43</td>
</tr>
<tr>
<td>Normalized Amount _mean_Sell_Collateral_Cash_Currency</td>
<td>27584.75</td>
</tr>
<tr>
<td>Normalized Amount _mean_Sell_Collateral_Security_Currency</td>
<td>13792.375</td>
</tr>
</tbody>
</table>

5.3. Unsupervised Module

As pointed out in Section 3, an unsupervised method is essential to detect new anomalous patterns never seen before. We decided to use Isolation Forest [31] due to its high performance in detecting outliers even if they are present in small amounts. 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. The splitting is performed along a random feature axis, using a random split value which 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. And $c(n)$ 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 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 between 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.

5.4. 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 [31], because it exhibited the best performance when compared to other supervised algorithms (as highlighted in Section 3).

Random Forest. The basic component of a Random Forest is a Decision Tree [31]. 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 datapoint 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 [32], 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 from 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 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).

6. Selection Strategies Module

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

6.0.1. 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. For this stage, two possible active learning selection strategies are available: 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 by [25], 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.

Algorithm 3: First stage: SELECT-DIVERSE

Input: \( U^t, \text{ratio}_{\text{unsup}}, \text{p}_{\text{unsup}} \)
Output: sample\(_{\text{unsup}}^t\)
1 Sort \( U^t \) by unsupervised score
2 Find \( \text{p}_{\text{unsup}} \) percentile. Call it \( \text{x}_{\text{p}} \)
3 Select \( x_i \subset U^t \) with score \( s_i > \text{x}_{\text{p}} \). Call it \( \text{anoms}_{\text{unsup}} \)
4 Run HDBSCAN algorithm on \( \text{anoms}_{\text{unsup}} \)
5 Define \( \text{cluster\_ratio} = \max \{ 1, \frac{\text{ratio}_{\text{unsup}}}{\text{nclusters}} \} \)
6 Sort cluster by cluster\_density
7 for \( i = 0, \ldots, \text{nclusters} \) do
8 Select \( \text{cluster\_ratio} \) samples from \( \text{cluster}_i \). Call it \( \text{C}_i \)
9 Append \( \text{C}_i \) to \( \text{sample}_{\text{unsup}}^t \). i.e.
10 \( \text{sample}_{\text{unsup}}^t = \text{sample}_{\text{unsup}}^t \cup \text{C}_i \)
11 if \( |\text{sample}_{\text{unsup}}^t| \geq \text{ratio}_{\text{unsup}} \) then
12 Break
13 Return \( \text{sample}_{\text{unsup}}^t \)

The clustering algorithm used for this strategy is HDBSCAN \[33, 34\]. This algorithm is based on the work by \[33\] and \[34\]. The first step of the algorithm is to build a weighted graph, where each data point 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: \( d(a, b) = \max \{ \text{core}_k(a), \text{core}_k(b), d(a, b) \} \). \( \text{core}_k(x) \) 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 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 \( \lambda_{\text{birth}} \) and \( \lambda_{\text{death}} \) 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 \( \lambda_p \), as the lambda value at which that point “fell out of the cluster”, which is a value somewhere between \( \lambda_{\text{birth}} \) and \( \lambda_{\text{death}} \). 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 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 unsselect all its descendants. Once we reach the root node we call the current set of selected clusters our flat clustering and return that.

6.0.2. 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.

6.0.3. 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.

Algorithm 4: Third stage: SELECT-ENTROPY

Input: \( U^i, \text{ratio}_{\text{sup}}, p_{\text{center}} \)
Output: \( c_{\text{center}} \)
1. Compute the distance of the score for each aggregation \( i \) to the center = 0.5.
   \( \text{dist}_i = |s_i - 0.5| \)
2. Sort \( \text{dist}_i \) in ascending order
3. Select \( \text{ratio}_{\text{sup}} \times p_{\text{center}} \) least distant aggregations. Call it \( c_{\text{center}} \)
4. Return \( c_{\text{center}} \)

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, samples, where the score discrepancy is close to 1.0, are selected (i.e., the models are in complete disagreement on whether the vectors are anomalous or not). In Algorithm 5 we present the pseudocode of the SELECT-CONFLICT strategy.

**Standardization and Ensembling.** Amaretto exploits the power of Random Forest, which outputs class probabilities \( \in [0,1] \), and Isolation Forest, which
Algorithm 5: Third stage: SELECT-CONFLICT

Input: $U^i, \text{ratio}_{sup}, \text{p}_{center}$
Output: $c_{center}$

1. Compute the difference between the supervised score and the unsupervised score for each aggregation $i$. Call it $unc_i$
2. Sort $unc_i$ in descending order
3. Select $\text{ratio}_{sup} \times \text{p}_{center}$ most uncertain aggregations. Call it $c_{center}$
4. Return $c_{center}$

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 that fits the anomaly score distribution and allows 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).

7. 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 7.2). Then, we test the same unsupervised techniques to assess their prediction ability with different daily budgets (Section 7.3). Afterward, we evaluate Random Forest against other supervised solutions to prove that they perform the best in our domain (Section 7.4). Furthermore, we prove the importance of an unsupervised model in combination with a supervised one in detecting new anomalous patterns (Section 7.5). Then, we compare the different selection strategies of Amaretto (Section 7.6). Finally, we compare Amaretto with AI², a state of the art active learning framework, in an AML scenario (Section 7.7).

7.1. Evaluation Approach and Metrics

The data contained in our dataset can be considered as time-series data. Therefore, 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 transactions. The second one is used to evaluate the model performance by running tests, and it includes 5 weeks of transactions. Given the temporal link of the data, we used a walk-forward testing approach [35] for evaluating the models. For the hyper-parameter optimization, we used Bayesian Optimization [36] because of 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.

7.1.1. 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:

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

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:

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

FScore. 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 random prediction and −1 indicates total disagreement between prediction and observation:

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

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.
Figure 4: Exp. 1 - ROC unsupervised algorithms comparison

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 [37] as: \( \text{Cost} = FP + C_R \times FN \). A normalization process can be applied to obtain a value that is independent from the number of transactions: \( \text{Norm. Cost} = \frac{FP + C_R \times FN}{TN + FP + C_R \times (TP + FN)} \). As suggested in [37], 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 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.

7.2. Experiment 1: Unsupervised Comparison

With this test, we compared models used in state-of-the-art solutions with Isolation Forest. In [18], Autoencoders are used for outlier detection. We also tested Variational Autoencoders as proposed in [38]. In [24], the unsupervised models comprise a Matrix-decomposition model, a Density-based model, and an Autoencoder, using PCA as a Matrix decomposition model [39] and using a Copula distribution as a Density-based model. We also tested a threshold-based model 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 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. As shown in Figure 4, 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.

7.3. Experiment 2: Daily Budget K Estimation

<table>
<thead>
<tr>
<th>K</th>
<th>Performance Metric</th>
<th>Autoencoder</th>
<th>Copula</th>
<th>Isolation Forest</th>
<th>Matrix Decomposition</th>
<th>Threshold based</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>TPR</td>
<td>0.003</td>
<td>0.032</td>
<td>0.142</td>
<td>0.021</td>
<td>0.028</td>
</tr>
<tr>
<td></td>
<td>FPR</td>
<td>0.001</td>
<td>0.001</td>
<td>0.0</td>
<td>0.001</td>
<td>0.001</td>
</tr>
<tr>
<td></td>
<td>FNR</td>
<td>0.997</td>
<td>0.968</td>
<td>0.858</td>
<td>0.979</td>
<td>0.972</td>
</tr>
<tr>
<td></td>
<td>AUC</td>
<td>0.612</td>
<td>0.558</td>
<td>0.897</td>
<td>0.676</td>
<td>0.898</td>
</tr>
<tr>
<td></td>
<td>Accuracy</td>
<td>0.991</td>
<td>0.991</td>
<td>0.993</td>
<td>0.991</td>
<td>0.991</td>
</tr>
<tr>
<td></td>
<td>Precision</td>
<td>0.015</td>
<td>0.208</td>
<td>0.904</td>
<td>0.131</td>
<td>0.181</td>
</tr>
<tr>
<td></td>
<td>FScore</td>
<td>0.005</td>
<td>0.055</td>
<td>0.246</td>
<td>0.037</td>
<td>0.048</td>
</tr>
<tr>
<td></td>
<td>MCC</td>
<td>0.003</td>
<td>0.078</td>
<td>0.357</td>
<td>0.05</td>
<td>0.068</td>
</tr>
<tr>
<td></td>
<td>Norm. Cost</td>
<td>0.448</td>
<td>0.435</td>
<td>0.385</td>
<td>0.44</td>
<td>0.437</td>
</tr>
<tr>
<td>50</td>
<td>TPR</td>
<td>0.015</td>
<td>0.063</td>
<td>0.279</td>
<td>0.051</td>
<td>0.205</td>
</tr>
<tr>
<td></td>
<td>FPR</td>
<td>0.006</td>
<td>0.006</td>
<td>0.004</td>
<td>0.006</td>
<td>0.005</td>
</tr>
<tr>
<td></td>
<td>FNR</td>
<td>0.985</td>
<td>0.937</td>
<td>0.721</td>
<td>0.949</td>
<td>0.795</td>
</tr>
<tr>
<td></td>
<td>AUC</td>
<td>0.612</td>
<td>0.558</td>
<td>0.897</td>
<td>0.676</td>
<td>0.808</td>
</tr>
<tr>
<td></td>
<td>Accuracy</td>
<td>0.986</td>
<td>0.987</td>
<td>0.99</td>
<td>0.986</td>
<td>0.989</td>
</tr>
<tr>
<td></td>
<td>Precision</td>
<td>0.018</td>
<td>0.081</td>
<td>0.355</td>
<td>0.065</td>
<td>0.261</td>
</tr>
<tr>
<td></td>
<td>FScore</td>
<td>0.017</td>
<td>0.07</td>
<td>0.312</td>
<td>0.057</td>
<td>0.229</td>
</tr>
<tr>
<td></td>
<td>MCC</td>
<td>0.01</td>
<td>0.064</td>
<td>0.309</td>
<td>0.05</td>
<td>0.225</td>
</tr>
<tr>
<td></td>
<td>Norm. Cost</td>
<td>0.446</td>
<td>0.424</td>
<td>0.326</td>
<td>0.429</td>
<td>0.36</td>
</tr>
<tr>
<td>100</td>
<td>TPR</td>
<td>0.021</td>
<td>0.086</td>
<td>0.366</td>
<td>0.066</td>
<td>0.364</td>
</tr>
<tr>
<td></td>
<td>FPR</td>
<td>0.013</td>
<td>0.012</td>
<td>0.01</td>
<td>0.012</td>
<td>0.01</td>
</tr>
<tr>
<td></td>
<td>FNR</td>
<td>0.979</td>
<td>0.914</td>
<td>0.634</td>
<td>0.935</td>
<td>0.696</td>
</tr>
<tr>
<td></td>
<td>Accuracy</td>
<td>0.98</td>
<td>0.981</td>
<td>0.985</td>
<td>0.98</td>
<td>0.984</td>
</tr>
<tr>
<td></td>
<td>Precision</td>
<td>0.013</td>
<td>0.055</td>
<td>0.233</td>
<td>0.041</td>
<td>0.195</td>
</tr>
<tr>
<td></td>
<td>AUC</td>
<td>0.612</td>
<td>0.558</td>
<td>0.897</td>
<td>0.676</td>
<td>0.898</td>
</tr>
<tr>
<td></td>
<td>FScore</td>
<td>0.016</td>
<td>0.067</td>
<td>0.284</td>
<td>0.05</td>
<td>0.237</td>
</tr>
<tr>
<td></td>
<td>MCC</td>
<td>0.007</td>
<td>0.059</td>
<td>0.284</td>
<td>0.042</td>
<td>0.236</td>
</tr>
<tr>
<td></td>
<td>Norm. Cost</td>
<td>0.446</td>
<td>0.417</td>
<td>0.29</td>
<td>0.427</td>
<td>0.318</td>
</tr>
<tr>
<td>200</td>
<td>TPR</td>
<td>0.032</td>
<td>0.112</td>
<td>0.463</td>
<td>0.092</td>
<td>0.407</td>
</tr>
<tr>
<td></td>
<td>FPR</td>
<td>0.025</td>
<td>0.025</td>
<td>0.022</td>
<td>0.025</td>
<td>0.022</td>
</tr>
<tr>
<td></td>
<td>FNR</td>
<td>0.968</td>
<td>0.888</td>
<td>0.537</td>
<td>0.908</td>
<td>0.593</td>
</tr>
<tr>
<td></td>
<td>Accuracy</td>
<td>0.967</td>
<td>0.968</td>
<td>0.974</td>
<td>0.968</td>
<td>0.973</td>
</tr>
<tr>
<td></td>
<td>Precision</td>
<td>0.01</td>
<td>0.036</td>
<td>0.147</td>
<td>0.029</td>
<td>0.13</td>
</tr>
<tr>
<td></td>
<td>AUC</td>
<td>0.612</td>
<td>0.558</td>
<td>0.897</td>
<td>0.676</td>
<td>0.898</td>
</tr>
<tr>
<td></td>
<td>FScore</td>
<td>0.015</td>
<td>0.055</td>
<td>0.223</td>
<td>0.044</td>
<td>0.197</td>
</tr>
<tr>
<td></td>
<td>MCC</td>
<td>0.004</td>
<td>0.05</td>
<td>0.251</td>
<td>0.038</td>
<td>0.219</td>
</tr>
<tr>
<td></td>
<td>Norm. Cost</td>
<td>0.449</td>
<td>0.412</td>
<td>0.253</td>
<td>0.421</td>
<td>0.278</td>
</tr>
</tbody>
</table>

For this experiment, we benchmark the performances of all 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 mean 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 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.

### 7.4. Experiment 3: Supervised Comparison

With this test, we compare the Random Forest model in Amaretto, with state-of-the-art supervised models, cited in Section 3 including Gradient Boosting and Category Boosting. Gradient Boosting [40] is considered one of the best algorithms for classification tasks. Category Boosting [41] is an alternative boosting algorithm based on decision trees. It offers computational and efficiency improvements compared to Gradient Boosting based models. The experiment is done by running predictions on a daily basis. Table 4 presents the average metrics for each technique. As shown in Table 4, the metrics are quite similar between Random Forest, Category Boosting (CatBoost), and Gradient Boosting (GBM) 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.

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

Table 4: Exp. 3 - Supervised algorithm comparison
7.5. Experiment 4: Detecting New Anomalies

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 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 5a and Figure 5b. As expected, the Isolation Forest model performance is consistent, while Random Forest exhibits a decay in performance when new anomalies are introduced. The True Positive Rate (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 Isolation Forest is not negatively affected by the new anomalies, showing its capability of detecting the new anomalous pattern introduced.

7.6. Experiment 5: Amaretto Configurations

The purpose of this experiment is to test which of the strategies are the most suitable for our dataset. The experiment works as follows: the first day, the labeled dataset is empty. 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 6a and 6b 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 6b) 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 strategy selected, even with a daily budget $K=10$, the average precision is higher than the one obtained with $K=20$ and comparable with the one obtained with $K=50$.

7.7. Experiment 6: Comparison with the State of the Art

In this final experiment, we compare Amaretto with $AI^2$ [24], since it represents, at 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 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
Figure 7: Experiment 6 - Comparison with the state of the art for scenario (I), (II), (III), and K=10.
per day over a period of 10 days from each framework and then use this labeled data to train the Random Forest and to predict all remaining high-level vectors. In the second part, we compare the frameworks in a real-world scenario, comparing 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 in a real-world scenario, comparing the performance taking into account different risk profiles for a financial institution.

I - Static Scenario. The purpose of the first part of this experiment is to verify the performance of the framework 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 7a and Figure 7b show the comparison of our system against AI² using the probability score. As exhibited by the ROC curve plotted in Figure 7a, Amaretto achieves an AUROC of 0.93, whereas AI² obtained 0.89. Furthermore, as shown in the Precision/Recall curve in Figure 7b, Amaretto reaches an average precision of 0.61, which is 31% better than AI².

II - 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 can be employed since no feedback was collected. After the first day, the Random Forest starts to work 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 7c shows the average precision computed using the probability score. Amaretto doubled its precision in approximately 10 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 AI² 0.57. As shown in Figure 7d, 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 AI² by circa 14%.

III - 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 7e, Figure 7f, Figure 7g, and Figure 7h, Amaretto outperforms AI² 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$^2$.

8. Limitations and Future Work

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 any useful insight and cannot be used to help the human analyst to better understand the result. A future work may focus on integrating into Amaretto a library implementing the explainability processes. For example, SHAP\cite{12} is one of the most interesting approaches for explaining model output.

Another limitation is represented by the dataset we worked on. Even though it was synthetically forged from a real-world dataset, it is very limited in terms of timespan, containing 60 days of transactions. A wider dataset would have allowed us to perform a deep analysis of the system’s evolution over time. For example, 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 to further analyze the anomalies. For example, investigating the relationships among these anomalies or which anomalies occur more frequently. As part of this work, 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 can accommodate for such errors. Another avenue for future research includes automatically tuning Amaretto parameters to further 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 models or deep learning models. For example, LSTM neural networks could be employed due to the temporal correlations in money laundering patterns. Finally, it would be interesting to evaluate Amaretto on the detection of other kinds of financial crime, like credit card fraud detection, to investigate its flexibility.

9. 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 of an unsupervised model for detecting known and unknown anomalous patterns, including four distinct 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 performance of the system. 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 was able to achieve state-of-the-art performance within a short time frame and with minimal manual input from a subject matter expert.

Acknowledgment

The research leading to these results has received funding from Napier Technologies Limited, which we would like to thank for the support, resources, and dataset provided.

References


[34] L. McInnes, J. Healy, Accelerated hierarchical density based clustering, 2017 IEEE International Conference on Data Mining Workshops (ICDMW) (Nov 2017). doi:10.1109/icdmw.2017.12 URL http://dx.doi.org/10.1109/ICDMW.2017.12


