POLITECNICO DI TORINO

Master's Degree in Computer Engineering

Politecnico
di Torino

Master's Degree Thesis

Cluster Analysis of Financial Transaction Data

Supervisors

Prof. Elena BARALIS

Doc. Flavio GIOBERGIA

Candidate

Jacopo DE CRISTOFARO

Academic Year 2022-2023

Abstract

Banks, being the pillars of every international financial system, have the duty to detect suspicious money movements related to criminal activities such as fraud, terrorism financing or money laundering, in order to protect their clients and countries from significant losses. The techniques used in the past, based on rule-based definitions, are no longer effective: the increase in digitization has not only made it easier for criminals to evade these systems rendering them obsolete. More reliable outlier detection systems must be built, necessarily based on the usage of big data and artificial intelligence techniques, to label suspicious transactions and provide useful insights to the final human operator, who will be responsible for conducting the necessary investigations to determine the real nature of the marked money exchange. The goal of this thesis is to design and implement a clustering-based pipeline, which is part of a larger architecture that aims to detect anomalies in a dataset of pass-through transactions. Specifically, while the other pipelines already implemented have the task of directly detecting anomalies at different levels of granularity (transaction-level or user-level) using unsupervised algorithms, the pipeline to be presented will primarily focus on enriching the information of the final user reports. For this purpose, the transactions will first be aggregated at various levels of granularity (users, banks, or countries) through a feature engineering process guided by the indications of domain experts. Subsequently, by employing clustering algorithms, actors with similar behavior in the chosen feature space are going to be detected. Multidimensional space does not allow for an easy interpretation of the clustering result and so, a continuous feature quantization step followed by a frequent itemsets extraction one will generate the descriptors, that better highlight the common structures of the entities within each cluster. Finally, an aggregate result analysis step is going to produce a small number of clusters that can be studied and labeled by a domain expert to facilitate further investigations of the reported anomalous users. In addition, to enhance the second pipeline, a new Outlier Detection Model based on the clustering output will be presented. The methodologies used for the entire pipeline implementation will be described in detail, along with the experimental results obtained.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Motivation

In the recent years financial crimes like payment fraud, money laundering, terrorist financing, etc. are on the rise. For example money laundering , which can be defined as the process of concealing the source of money generated from illegal proceeds through a complex scheme of bank transactions, in the 2023 is worth to $800 billion from $2 trillion that is almost the 2%-5% of the global gross domestic product (GDP). The illegal activities that usually are linked with this process are cyber fraud schemes, smuggling, illegal arm sales, embezzlement, insider trading, corruption, human trafficking, drug trafficking and prostitution. There are many reasons why it is crucial to combat financial crime:

- **National Security**: some financial crimes, such as terrorism financing, pose threats to national security, monitoring and preventing these crimes are essential for a country's safety.
- **Protection of Individuals and Businesses**: fraud and money laundering, can cause significant financial harm to individuals and businesses. Combating these crimes helps safeguard people's savings and investments.
- **Market Integrity**: corruption and insider trading can distort competition and harm investors.
- **Legal and Ethical Compliance**: combating financial crimes promotes adherence to laws and ethical business practices, contributing to the establishment of just and ethical societies and communities.

In 1989, on the initiative of the G7 the Financial Action Task Force (FATF) was founded in order to combat money laundering by developing effective policies.

After the events of the 11 September 2001, the mission of the task force was expanded to include also the battle against terrorism financing. In order to reflect the every evolving patterns and techniques used by the criminal minds, every year the standards set by the FATF are revised where the banks play a crucial role. They are the heart of every national financial system and also are the only entities that can have a global view of the money movements; for these reasons they have the duty to be on the front line of this fight by being equipped with effective tools that are able to detect suspicious transactions or anomalous behaviour at higher level. The first "Transaction Monitoring" models to be employer were very simple since they were based on a fixed set of rules and nowadays are almost useless; their limits are the fact that need to be constantly updated, which is impracticable in real scenarios, and output a huge number of false positive alarms that translates for the human operator, in charge of doing further researches, in wasting time unnecessarily. Also the process of the digital transformation increased the number of transactions that the banking system execute every day, forcing the financial institution to abandon the old solution that were designed to deal only with a not very large volume of data. As the technology improves helping the criminal minds to find new strategies to move at breackneck speed "dirty" money accross countries, new solutions can be designed and implemented in order to improve the detection rate of the suspicious transactions, introducing big data and machine learning techniques.

1.2 Contribution

A large italian bank is engaged in the fight against money laundering and the financing of terrorism investing in the usage of the Artificial Intelligence; this thesis work is part of a project requested by this financial institution conducted at the SmartData research center of the Politecnico di Torino. The overall project consists in the implementation of machine learning-based anomaly detection model for pass-through transactions, which is characterized by three parallel pipelines that works at different levels of granularity, from the transaction one to bank one or above. The main goal of this contribution is the development of the third pipeline, whose output is a set of labels and descriptions that can be assigned to the various entities (users, banks or countries) and which can be used to enrich the information originating from the first two pipelines, that are directly involved respectively in extracting anomalies at transaction and user level. Since the dataset provided consists of approximately 6 millions of transactions, the first block designed for the presented pipeline consists of a feature extraction module that is able to describe through a set of feature the actors in the financial network at the desired level in a specified window of time. The following block will cluster the entities in order to

group together them together accordingly to the characteristics computed before and since the most of the clustering algorithm are not natively interpretable an extraction of the descriptors, that can be translated in the mining of frequent itemsets pass, will be done in order to better highlight what are the commonalities within the points in the clusters. Since most of the feature extracted on these entities are not categorical a quantile binning step will be performed. The pipeline ends with an aggregate result analysis block will be employed to produce a small number of clusters that can be studied and labeled by a domain expert to facilitate further investigation of the anomalous user reported. The absence of annotations within the dataset poses a significant challenge for the assessment of the results of the first pipelines, for this reason the clustering block will be also employed to conduct experiments that will evaluate the degree of agreement between it and the four unsupervised outlier detection algorithm proposed so far.

Chapter 2

Theoretical elements and related works

As mentioned in the introduction, one of the core algorithm that need to be employed in the pipeline to be developed is the clustering one. For this reason an exploration of the state of art of the clustering algorithms used that deals with banking transaction has been performed. This step is not only useful to understand the best promising techniques but also to better understand where are the features that best describe the data. Before discussing some works published in the recent years, a briefly introduction of the main concepts and algorithms used in this work will be made.

2.1 Clustering overview

In the context of exploratory data analysis, which is the process that allows for initial studies on a dataset to understand its data structure, identify outliers, or test hypotheses, cluster analysis, or simply clustering, is the primary task. Informally, the definition of clustering can be as follows: it is the technique that enables the organization of objects into groups that, in some way, share certain characteristics. A cluster is thus a set of objects that are “similar" for some reason and are “dissimilar" from objects in other clusters. The formal definition of clustering cannot be uniquely provided because it is possible to “group" points using various criteria. In the literature, there are indeed different models, the main ones are:

- **Centroid-based**: According to this concept, each cluster is represented by a centroid, which is a point representing the center of the cluster and may not necessarily be one of the data points under consideration. Each data point is then assigned to the cluster whose centroid is closest to it. Algorithms falling

into this category require specifying an input, the number k of clusters to be formed. Therefore, if you don't have a clear idea of how many clusters the dataset may "contains" multiple runs of the algorithm with different values of k may need to be performed.

- **Density-based**: This approach is based on the idea that each cluster corresponds to a region in space with a high density of data points, and clusters are separated from each other by regions of space where the point density is low. The "definition" of density for these algorithms is provided by specifying two input values, $MinPts$ and ϵ. These values indicate the minimum number of points that a point must have within a certain radius for the sphere it describes to be considered as having high density. Points that do not belong to any high-density area will be classified as noise points and will not be assigned to any cluster. For some applications, it may be useful to remove outliers, while in others where every data point needs to be assigned to a cluster, the presence of noise points can be problematic.

- **Distribution-based**: This methodology relies on distribution models, where data points that appear to be "generated" by the same distribution are assigned to the same cluster. The disadvantage of using this type of clustering is that it requires prior knowledge of the data's distribution type, which may not be precisely defined from a mathematical standpoint.

Starting from the same dataset, different cluster methodologies will lead to the formation of clusters that, when visualized, have different shapes. Clustering algorithms can also be divided into two main classes:

- **Partitional**: In this class, each data point belongs to one and only one cluster.

- **Hierarchical**: In hierarchical clustering, the output of the algorithm is to identify a hierarchy of clusters, which can be graphically represented by a dendrogram. This type of model can further be categorized as agglomerative or divisive. In the agglomerative approach, starting with a specific number of clusters, clusters that are most "similar" to each other are iteratively merged until all clusters are merged into one. In the divisive approach, starting with the entire dataset, clusters are divided into smaller, more "heterogeneous" clusters at each iteration.

If there is reason to believe that the records in the dataset to be explored can be represented by a hierarchy of clusters, then the second type might be convenient. Otherwise, due to its simplicity, including from a computational perspective, the first type is preferred. The following subsections will serve to introduce briefly the two algorithms explored in this thesis.

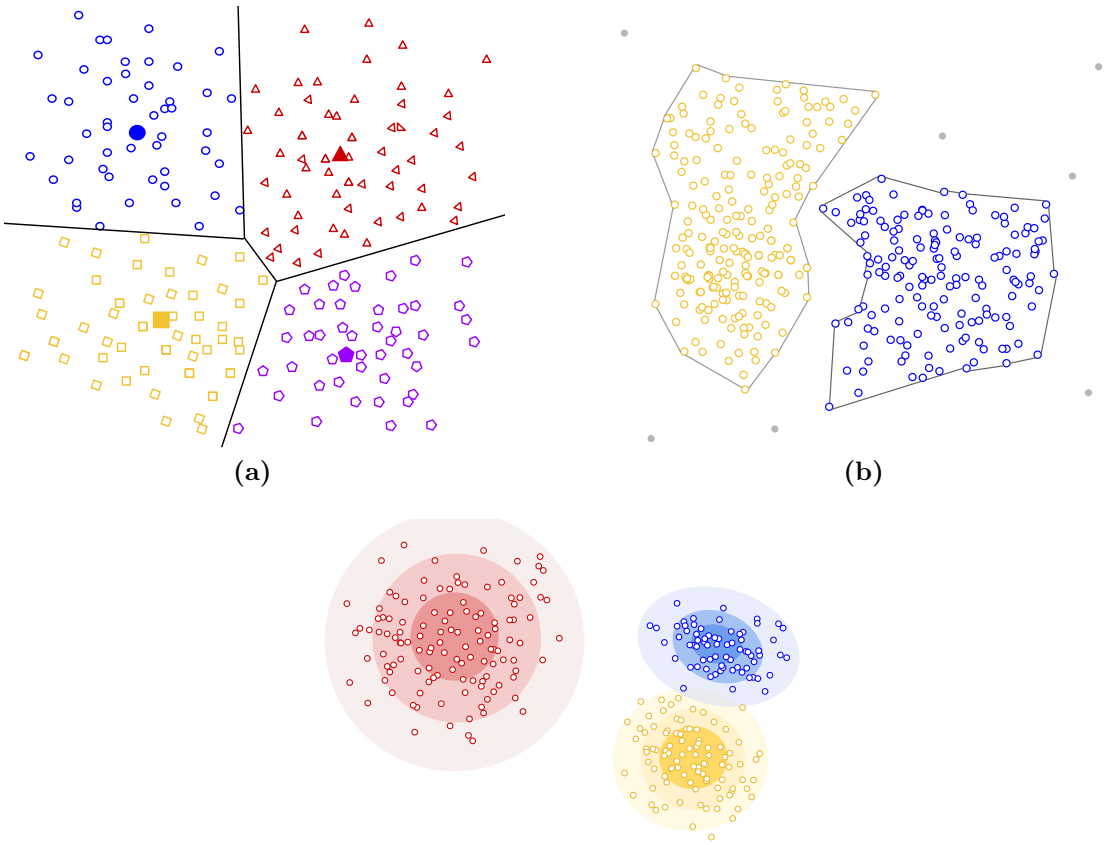


Figure 2.1: (a) example of centroid-based clustering, (b) example of density-based clustering and (c) example of distribution based clustering. Images taken from [1].

2.1.1 K-Means

K-Means [2] is the most commonly used clustering algorithm, and it falls under the partitional class and it is centroid-based. The k in K-Means refers to the number of clusters that we want to detect and so it should be specified by the user, and each cluster is identified by a point called centroid whom is computed as the mean of the objects within the cluster. Let $X = \{x_1, x_2, ..., x_n\}$ be the dataset under study, where a generic $x_i \in R^d$ and d is the number of feature used to describe the data. Then let $G_1, ..., G_k$ the clusters and $C = \{c_1, c_2, ..., c_k\}$ be the set of the associated centroids (which is initialized in someway), each point will be assigned to the cluster whose centroid is the closest one. The objective function that the

algorithm tries to minimize is the following:

$$e_k(X;C) = \sum_{i=1}^{n} \min_{j \in \{1,\dots,k\}} d(x_i, c_j) \tag{2.1}$$

where $d(x, c)$ is the distance function that we want to employ to evaluate the "similarity" between the points, the function that is commonly used is the Euclidean one (others popular are Minkowski or the Cosine Similarity). The K-Means problem belongs to the NP-hard problems and for this reason do not exist any implementation of it that is able to find the global optimum in a reasonable time. However, are available efficient heuristic algorithm that in a short time converge in a short time to a local optimum, typically them rely on Lloyd's or Elkan's algorithm. In the following an informal step-by-step explanation of how typically the implementations of K-Means work:

- **Initialization**: Start by selecting k initial cluster centroids. These centroids are typically initialized randomly from the data points, or a smarter way to do it is by using K-Means++ [3]. This last strategy is the most used one since it speed up convergence.
- **Assignment**: in this step the distances between the points and the centroid are computed. Then each point is assigned to the cluster with the closest centroid.
- **Update**: since the membership of the points to the cluster may have been changed, the centroids are recomputed. The new centroids are determined by computing the mean of all the data points assigned to each cluster in the previous step.
- **Repeat**: Steps 2 and 3 are repeated iteratively until a stopping criterion is met. The common stopping criteria used are: the membership of the points not changed between two consecutive iterations, the difference in the cluster centers of two consecutive iterations is below than a specified threshold or the specified maximum number of iterations is reached.

K-Means is very fast (is one of the fastest clustering algorithm available): since at each iteration we need to compute for each point the distance between it and the cluster centers, the average complexity is $\mathcal{O}(k \cdot n \cdot d \cdot t)$, where t is the number of iterations. For this reason, K-Means represents one of the few options to be considered when there is the need of clustering very large data or high dimensionality ones (as it will shown later, is our case). The biggest drawbacks are: k should be manually set, is highly sensitive to noise/outliers and the results depends on the initial value of the centroids.

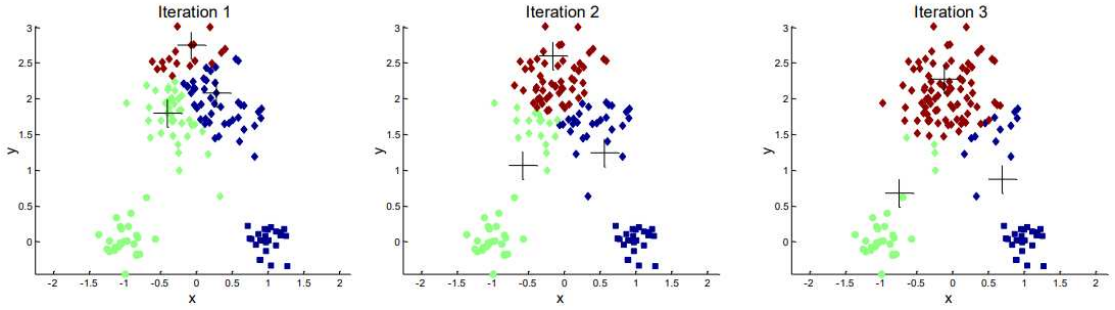


Figure 2.2: Visualization of intermediate cluster results using K-means with $k = 3$. At each iteration the centroids (represented by crosses) are recomputed and so the clusters (represented by different colors). Images taken from [4].

2.1.2 DBSCAN

Another well known partitional clustering algorithm is Density-based spatial clustering of applications with noise (DBSCAN) [5], as the name suggests falls under the density-based class models. The key basic idea behind this type of clustering is that for each point that belongs to a cluster must there, within the neighborhood described by the radius ϵ, at least a minimum number of points, represented by the hyper-parameter $MinPts$, i.e. the cardinality of it has to exceed this threshold; this means that may be points that will not belong to any cluster and so will be classified as noise. To better understand how DBSCAN works formally, some key concept will be briefly introduced. Fixing ϵ and $MinPts$:

- A point p is a ***core point*** if $|N_\epsilon(p)| \geq MinPts$ where $N_\epsilon(p)$ is the set of points whose distance from q is below ϵ (i.e. is the neighborhood of p w.r.t ϵ)
- A point $p \in X$ is ***directly-density*** from a point $q \in X$, if $p \in N_\epsilon(q)$ and q is a core point.
- A point p is ***reachable*** from an object q if there is a chain of points $p_1, ..., p_n$ with $p_1 = q$ and where each p_{i+1} is directly reachable from p_i. This means that all the points on the path must be core point, with the exception of q. This property is not symmetrical: not-core points cannot reach core points, while the opposite is possible.
- A point p is ***connected*** to a point q if there is an object o such that both q and p are reachable from o. This property is symmetrical.
- A point p is defined as ***noise*** (or ***outlier***), if is not reachable by any other point.

Having introduced the notion of reachability and connectedness, is possible to define which property the DBSCAN clusters satisfy:

- **Maximality**: if a point $p \in G_i$ and a point q is reachable from p then also $q \in G_i$

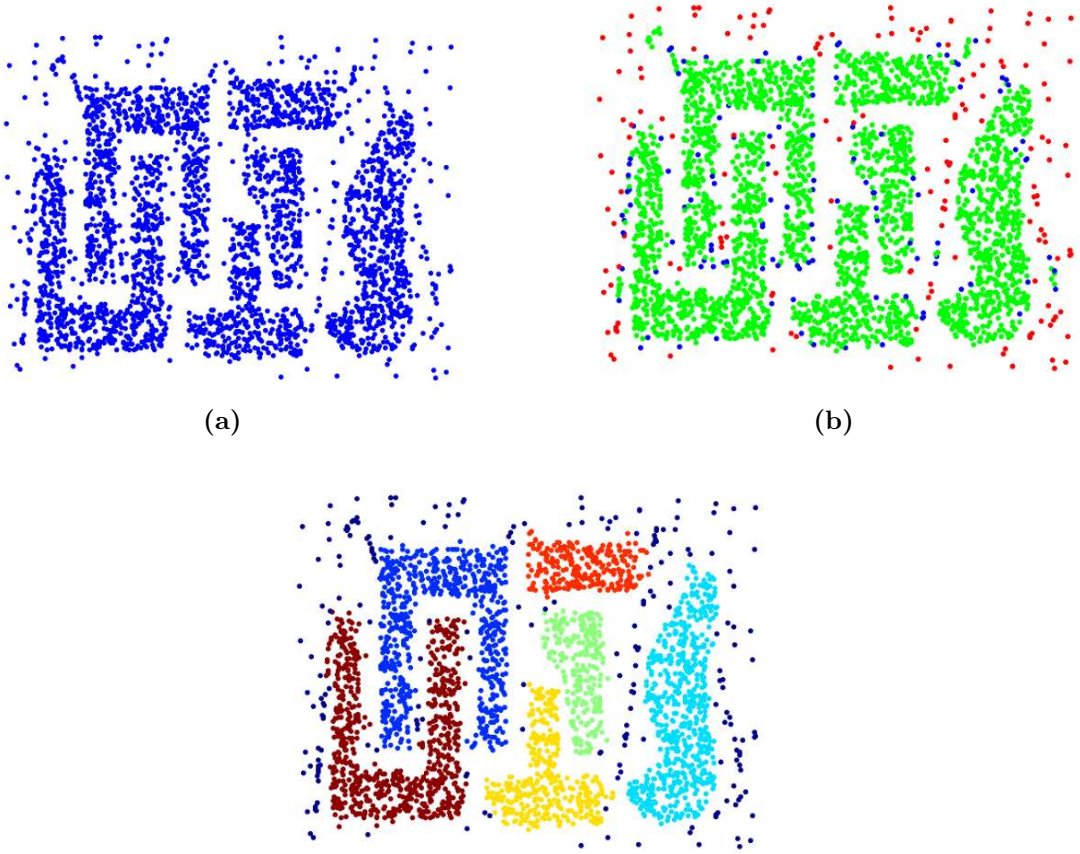
- **Connectivity**: if p and q are connected then $p, q \in G_i$

In other words if p is a core point it will forms a cluster with all the points that are reachable from it and also with the one that are only connected to it. While K-Means performs well if the assumption of having spherical shape "natural" clusters is met, DBSCAN is able to find clusters of arbitrarly shapes, since it is able to adapt to the characteristics of the data dynamically. Another huge advantage is the fact that there is no need at all of specifying the number of cluster to extract, contrary to K-Means. Since DBSCAN introduce the notion of outlier, for some applications this can be a disadvantage, for example if there is the need to assign each point to a cluster (like for the final goal of our thesis); is possible to circumvent it by assigning the outliers to a manually created cluster, however even if the data within it share a logical common characteristic (are outliers), will unlikely presents some common structural behaviors (making the clustering results even less interpretable). The other main disadvantages are:

- ϵ and $MinPts$ are hyper-parameters and so should be specified manually by the user. This values significantly influence the process, and so many runs are required in order to find the optimal values. However exists some rules of thumb that can be adopted to determine more intelligent values, for example using the K-Graph to choose a proper value of ϵ and by setting $MinPts = 2d$,

- DBSCAN may have hard time handling clusters with variable densities. Specifically, clusters with significantly different densities can be challenging to correctly identify, since the ϵ-$MinPts$ pair cannot be chosen in a way to adapt to the characteristic of each cluster.

Let's now analyze the computational complexity of the algorithm: the time part is mostly covered by the regionQuery operations (the one used to discover the neighborhood of a point). This operation is performed only one for each point and if an indexing structure is used it will execute in $\mathcal{O}(\log n)$. Choosing a good value of ϵ the overall complexity of the algorithm will be $\mathcal{O}(n \log n)$, but in the worst case would be $\mathcal{O}(n^2)$. This means that DBSCAN is not suited for large datasets and will be executed in a reasonable time only if good values for the hyper-parameters are set.



(a) **(b)** **(c)**

Figure 2.3: (a) Original points to be clustered using DBSCAN. (b) Point types: core (green), border (blue) and noise (red). (c) Clustering output (clusters are identified by colors). Images taken from [4].

2.1.3 Evaluation

So far we have discussed how the clustering algorithms build a cluster, but how we can assess the quality of the results in order to understand which is better? Quoting James and Dubes "The validation of clustering structures is the most difficult and frustrating part of cluster analysis" [6]. While for supervised classification we have a variety of measures to evaluate how good our model is (accuracy, precision, recall), for cluster analysis, since it lies under the unsupervised domain, is difficult to assess the how well our data have been grouped. Numerical measures can be employed to judge various aspects of cluster validity, and be classified into three classes:

- **Internal**: when the quality of the result is assessed based on the features

considered by the clustering algorithm itself without respect to external information. For this reason, since a clustering algorithm may already optimize metrics used by these indices, a high score does not always indicate a useful result. Some example are Sum of Squared Error (SSE), cluster cohesion, cluster separation, RandIndex, adjusted rand-index, Silhouette index.

- **External**: these indices are based on data that was not considered for cluster formation, such as class labels or external benchmarks, which are nothing more than a set of preclassified items created by domain experts. This allows for an assessment of how close the clustering result is to the desired one. Some example are purity and entropy
- **Relative index**: Used to compare two different clusterings or clusters.

The external metrics cannot be always used since they require the strict requirement of having information that can be only be supplied by the experts of the dataset domain. However this kind of indexes can bias the final choice of the clustering since the reproduction of known knowledge may will not allow to capture hidden intrinsic structures of the data. In the majority of cases the internal ones are used, since them do not require to met any particular constraint.

Silhoutte

The silhoutte score [7] is a internal metrics that can be used to evaluate how well each point lies within its cluster and generally can be also computed for an individual cluster or for the cluster results. The silhoutte score s computed for a point i can be defined as follows:

$$s(i) = \frac{b(i) - a(i)}{max(a(i), b(i)} \tag{2.2}$$

where $a(i)$ is the the average distance of i with all other objects within the same cluster, and $b(i)$ is the minimum average distance between any other cluster (the one of which i is member is excluded). To be complete, formally we have:

$$a(i) = \frac{1}{|G_I| - 1} \sum_{j \in G_I, i \neq j} d(i, j) \tag{2.3}$$

$$b(i) = \min_{J=I} \frac{1}{G_j} \sum_{j \in G_j} d(i, j) \tag{2.4}$$

Observing the formulation, the metric can be computed for each point only if each cluster contains at least 2 points, and typically this requirement is always satisfied. The silhoutte range spans from -1 to 1, and higher is the value better is the point

is matched to its own cluster and badly matched to other ones. Heuristically if many points have a value close to 1, then the resulting configuration is good. As mentioned before is possible to compute the silhoutte even for a specific cluster and for the overall clustering. Respectively and formally, with K equals to the number of clusters, he have:

$$s(G_I) = \frac{1}{|G_I|} \sum_{i \in G_I} s(i) \tag{2.5}$$

$$s(C) = \frac{1}{K} \sum_{I=1} K s(G_I) \tag{2.6}$$

The observations made for the values of the metric are still valid here. Since for K-Means is necessarily to manually select the value k, a silhoutte analysis can be performed: by launching the algorithm with different values of it and computing for each execution the overall silhoutte score, it is possible to show through a plot how the metric change. Then the value of k that shows the best promising performance, will be chosen with the corresponding results.

2.2 Dimensionality reduction

One of the most used and famous distance definition is the Euclidean one, it was defined by Euclid by observing our real world which is essentially three dimensional. However in a high dimensionality world it loses all the meaning, due to a popular issue called “Curse of Dimensionality" [8]; the term refers to various problems that may arise when we are dealing with high dimensional data that do not occur in a low dimensional one. Intuitively all this problem are due to the fact that as the number of dimension the volume of this space grows exponentially and so the points that are lying within it become sparse. Let d be the number of dimension in the space and $MaxDist$ and $MinDist$ respectively the maximum and the minimum distances that we can observe between any points within it, formally we have:

$$\lim_{d \to \infty} \frac{MaxDist - MinDist}{MinDist} = 0 \tag{2.7}$$

So in this particular settings, the Euclidean distance between any two points essentially become a constant and so meaningless. Not only the Euclidean distance is affected by this “curse": in fact we can observed it also for other distance metrics like the Manhattan or the Cosine Similarity ones. In order to deal with this kind of problem, in such way to still use the Euclidean distance, is to circumvent it by reducing the dimensionality of the dataset before injecting it into the final algorithm. This can be achieved in several ways, the common ones involve to “drop"

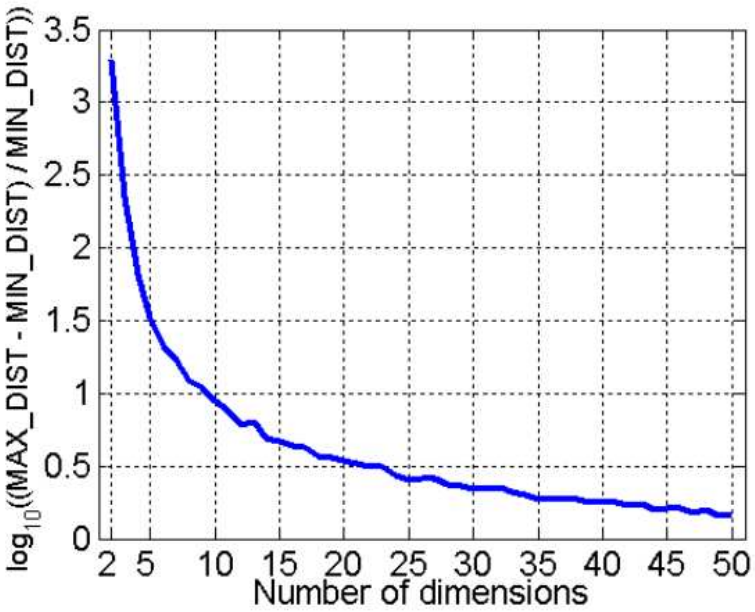


Figure 2.4: Plots that shows how the difference between the maximum distance between two points and the minimum one shrinks to zero as the number of dimensions increases, within a dataset of 500 randomized records. Images taken from [4].

those features that irrelevant (i.e. the ones that present always the same value) or redundant (i.e. the ones strongly correlated to another one) or by projecting it into a lower space, in which the mapping captures the largest amount of variation in data. Reducing the dimensionality of the data has additional benefits:

- **Memory**: less features means less amount of space required to store each point in memory or in disk

- **Time**: the computation of any distance depends on the number of features that characterize the data. Computing the distance in a lower space will be faster

- **Visualization**: as humans we can only understands 3 or lower dimension spaces. Lowering the dimensionality with the help of techniques such as PCA, allows to visualize the data projecting them on the so-called “principal" direction.

2.2.1 Pearson correlation coefficient

In some cases the data under analysis are described by features that brings the same kind of information. For example let's assume that we are dealing with receipts, possible features can be the purchase price of the product and the amount of taxes to be applied on it. The last one, for each product, will always be a fixed percentage of the first one, and for this reason do not "enrich" the description of the receipt. It's possible to compute the linear correlation between the distribution of two features by using the Pearson correlation coefficient [9]. Let X and Y are two random variables that describe the distribution of two distinct features, formally this coefficient is defined as:

$$\rho_{X,Y} = \frac{cov(X,Y)}{\sigma_X \sigma_Y} \tag{2.8}$$

where *cov* is the covariance function, σ_X is the standard deviation of X and σ_Y is the standard deviation of Y. This metric can be seen as a normalized version of the covariance between two R.V. since the value will be always between -1 and 1, and higher is the absolute value of it and more correlated the variables will be. For instance, if this value computed between two features is close to 1, we can consider to drop one of the two in order to decrease the number of dimensions.

2.2.2 Principal component analysis

Principal component analysis (PCA) [10] is one the most used technique for reducing the dimensionality of a dataset. Roughly this mapping is achieved by searching for a new "coordinate" system, with less dimensions, where a good amount of the variation of the data is preserved. Formally we want to search for a subspace, defined by $\mathbf{P} \in R^{n \times m}$ (where n is the number of dimension of the original space and m is the one of the subspace), whose columns are orthonormal. A reasonable criterion, that this lower dimension space needs to satistfy, may be the minimization of the average reconstruction error, and so (where K is the number of samples, formally we have:

$$\boldsymbol{P}^* = \arg\min_P \frac{1}{K} \sum_{i=1}^{K} ||x_i - PP^T x_i|| \tag{2.9}$$

which is a eigenvalue/eigenvector problem. It can be shown that the optimal solution is then given by the matrix P whose columns are the m eigenvectors of $\frac{1}{K}\sum_{i=1}^{K} x_i x_i^T$ corresponding to the m largest eigenvalues, where the first column is called principal component. The lower the dimensionality respect to the original space, the higher the information loss is, so how to select an optimal value of m? Remembering that each eigenvalue corresponds to the explained variance along the corresponding axis, we can select m as the lowest number such that the cumulative

explained variance of the first m eigenvalues is higher than a fixed threshold (i.e. to retain at least the 80% of the variance).

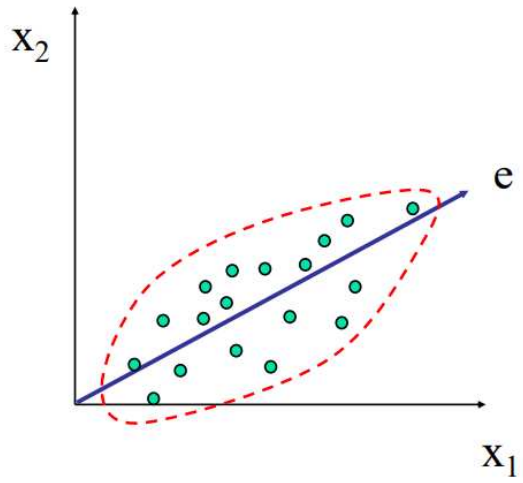


Figure 2.5: Principal component (the e axis) computed on a 2D dataset using PCA. The principal component captures the largest amount of variation in data. Images taken from [4].

Data are often described by features with very different scales, and the one that presents the wider ranges will have a greater contribute in computation of the principal axis w.r.t the ones that presents narrower ones. Also, if the dataset mean is far from the origin, the principal component of PCA will connect them; this direction in most cases does not bring any value. For this reason is almost mandatory to scale our data before fitting PCA, and a standard approach to achieve this is by using Z-Score (each feature is recalculated by subtracting the mean of the distribution and by dividing the outcome by distribution standard deviation). Also, it is important to drop redundant (highly correlated) features before this analysis since them can amplify the variance of the principal components. To better understand this phenomenon, the following experiment will be conducted by picking up three features that are highly correlated among them and a forth one which is independent, them are respectively *feature_1*, *feature_2*, *feature_3* and *feature_4*. By computing the PCA only on the first and the last features which are totally uncorrelated, the contribution, that can be measure in terms of the explained variance of each component, is almost identical. After adding a variable correlated with the first one, the variance of the first component is now twice the size of the second. After adding another correlated variables, the size of the first component is now three times bigger than the second; the results are shown in 2.6. This means that, by retaining correlated features, the contribution of each variable

will be indirectly unbalanced in an eventual clustering process, since the formation of clusters will most likely based only on those that are correlated.

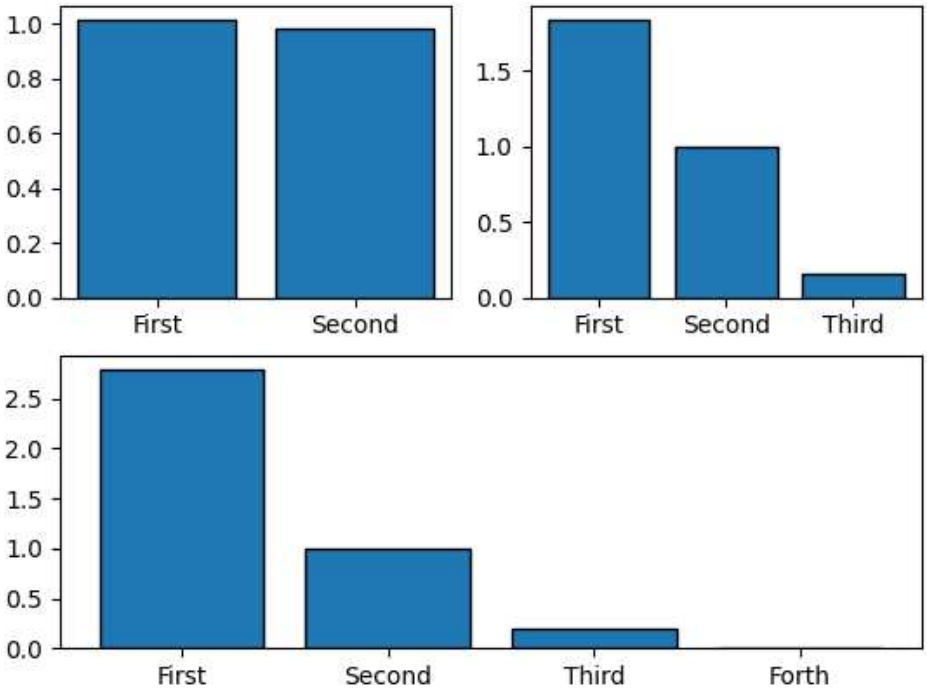


Figure 2.6: This bar plots show how the variance of the principal components is affected if the variables under analysis are correlated. The first plot on the top left has performed on *feature_1* and *feature_4*, which are totally independent. The one on the top right, has been performed by adding the *feature_2* variable which is correlated to the first one. The last plot involves the contribution of an additional correlated feature *feature_3*.

Since the first two (also three) components of this analysis are the ones to "embed" most of the variance of the original space, them can be used to visualize the data in a human understandable plot; in this way it is possible to investigate how the data are clustered in the space and doing further investigations is possible to find to which of the original features these principal axis are referring to.

2.3 Frequent Pattern Mining

The problem of frequent itemset extraction is often mistakenly confused with association rule mining. However, the latter is significantly more complex and depends on a prior step of itemsets extraction. To intuitively describe this technique of data characterization, we will refer to the “market-basket" model [11]. In this model, there are essentially two types of elements: “items" and “baskets", where the latter are nothing more than collections of the former. The goal of frequent itemsets extraction is to find sets of items that appear with a frequency greater than a specified parameter within the available baskets. This model was originally used for analyzing real market baskets. By extracting frequent itemsets, a retailer could determine which products were frequently purchased together. Nowadays, this technique is used in various fields such as recommendation systems, user behavior analysis, medical analysis, text analysis, and more. Formally we have, given a a transactional dataset, where a transaction is a (not ordered) set of transactions (i.e. market basket data, textual data, structured data), the following definitions:

- ***Itemset***: is a set including one or more items.
- ***Support***: is the fraction of transactions that contain an specific itemset
- ***Frequent itemset***: is an itemset whose support is greater than or equal to a specified support threshold
- ***minsup***: the threshold mentioned before

Unlike machine learning algorithms, the results of frequent itemsets extraction is deterministic. The property that the result should satisfy are:

- **Correctness**: all the itemsets found must be frequent
- **Completeness**: all the frequent itemsets within the transactional dataset must be found

A trivial way to extract frequent itemsets is the brute force one by enumerating all the possible permutations and the count the support; obviously this is completely ineffecient, the complexity is $\prime(T2dw)$, where T is the size of the dataset, d is the number of items and w the average length of a transaction. Exist several implementation of the frequent itemset generation, that cut down by a lot, the complexity of the brute force approach. The most famous are:

- **Apriori** [12]: which is based on the homonym principle “If an itemset is frequent, then all of its subsets must also be frequent". One of the main perfomance issue of the it, is the fact that requires multiple database scan (if n is the length of the longest frequent itemsets, it requires $n+1$ scans in total)

- **Fp-growth** [13]: Exploits a main memory compressed representation of the database, the FP-tree. The frequent itemsets are extracted by a visiting recursevily it and only two database scans are needed, one for creating the FP-tree and one for count the item support.

Since both the algorithm will lead to the same result (given a specified transactional dataset), the choice between which implementation to use is based only on the computational efficiency; in most cases Fp-growth is the fastest. In somecases the number of frequent itemsets that are “lying" in the dataset is very high, so may be preferred to extract only a compact representation. For this reason we need to introduce the definition of *maximal frequent itemsets*: an itemset is **maximal** if none of its immediate supersets is frequent. Through the apriori principle we are able to deduce all the frequent itemsubset from the maximal ones. The discovery of maximal itemsets is faster and requires less memory, however if we are interested also in the support of each frequent this become useless since we can only know the lower-bound of this value (the support of the related maximals). **Fp-max** [14] is a variant of the Fp-growth algorithm, which focuses only on the extraction of this compressed representation.

2.4 Clustering of financial data in literature

In the financial world, clustering techniques are primarily used for the anomaly detection task. The goal of this task is to identify those points, also known as outliers, that exhibit substantial differences from the majority of other data. Yang et al., [15] use DBSCAN to extract bank accounts that may are performing money laundering activities. Starting from a database of transactions, the data are aggregated to compute the features that describes the accounts under analysis: this set extracted includes monthly deposit frequency, monthly deposit amount, monthly withdrawal frequency, and monthly withdrawal amount. Then after this preparation step, different runs of DBSCAN are executed: each one will cluster the points according to only one feature at time; this step will produce for each run a set of outliers whom differs a lot from the current feature under analysis. In the final step, called by the author the “Link Analysis" one, the level of agreement between the various clustering results is computed and according to it a suspicious level is assigned to the found outliers. Arévalo et al., [16] experiment both K-Means and DBSCAN to form clusters that will be analyzed to detect anomalous transactions. The feature engineering process is interesting: each transaction is described by the amount of money involved and a set of one-hot encoded features describing the transaction’s origin and beneficiary, as well as a set of network features that describe interactions between the involved participants in the system within a specific time window (total number of transactions sent/received and the total amount of money

sent/received); this is done in order to let the algorithm to also considers the behaviour of them during the period under consideration. Since the transactions that will be clustered are also characterized by these institution-level features, large one-hot encoded vectors are introduced. In order to deal with the dimensionality increasing of the dataset studied and with the noise introduced by the one hot encoded features, PCA was employed in order keeping the 85% of the explained variance. Finally, to make interpretable the results of the clustering, Random Forests are used in order to understand which are the features that influenced the algorithm the most. Larik and Haider [17] developed a Anti Money Laundering system in which for the suspicious user is computed an anomaly index score, named AICAF (Anomaly Index Computation based on Amount and Frequency), to measure how much the behaviour of a customer deviate from the one of the cluster in which belongs. The feature on which the process relies are the transaction amount and its type (credit or debit card). Then, the K-Means algorithm is used to determine the optimal number of clusters using the Sum of Squared Error metric, and then a modified version of the Euclidean Adaptive Resonance Theory (EART) algorithm is used to form clusters of balanced sizes. After this step, the AICAF index is computed for each user, and the ones that exhibit the higher values are considered anomalous. Shahriar et al., [18] the goal of the work is different from the previous discussed ones: a bank database is analyzed from the viewpoint of customer behavioral. Since the final goal of the architecture presented is to help the organization to acquire new customers, the main features on which the clustering process focuses, beside the amount carried out by the transaction, are categorical like: education level, occupation, sex and age of the entities. K-Means is used to cluster the data and an association rule step take place to understand the behavior of the users within the groups created.

Chapter 3

Architecture overview

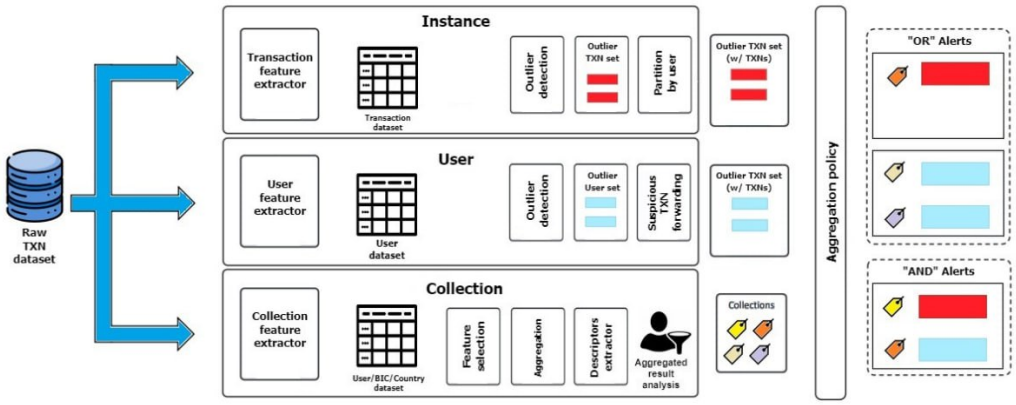


Figure 3.1: The overall architecture, comprised of a shared data loading phase, three parallel pipelines that work at different levels of granularity, and a final aggregation phase that produces user-level reports.

In this chapter will be introduced the overall architecture for a machine learning-based anomaly detection in pass-through transactions. It is characterized by the three pipelines that can be executed in parallel, that share a common data-loading phase. Each pipeline is designed in order to identify anomalies at different level of granularity, from the single transaction one to the country one. Then, the output of these pipelines is injected in an aggregation phase that will produces a user-level reports, that will contain the suspicious ones along their most relevant transactions enriched with useful information extracted for the human operation, that will later

investigate the case.

3.1 Data Loading

This phase is common to all the three pipelines. In this step, will be loaded in memory transaction data from CSV files. At the moment, only one file is available and contains about 6.3 million transactions that spans for four consecutive months. Due to the sensitive nature of the financial data, the identifiers of the users and the banks were been anonymized by the partner bank with not meaningful strings. Each transaction is described by many feature, we will focus on the main most relevant ones that are useful for our task:

- *transaction_id*: represents a censored identifier of the transaction which in most cases is a unique universal identifier.
- *data_ref*: indicates the date and the time in which the transaction has been carried out. This feature is useful in order to group that transactions that took place in a specified time window.
- *BIC_o_anon*: it is the censored Bank Identifier Code of the sender's bank.
- *BIC_b_anon*: it is the censored Bank Identifier Code of the receiver's bank.
- *user_o*: it is the anonymized version of the sender's account identifier.
- *user_b*: it is the anonymized version of the receiver's account identifier.
- *amount*: represents the transaction amount carried by the current transaction converted to Euro.
- *CTRYbnkCD_o*: describes to the sender's bank country code (i.e. IT for an italian bank).
- *CTRYbnkCD_b*: describes to the receiver's bank country code.
- *CTRYresCD_o*: describes the sender country code.
- *CTRYresCD_b*: describes the receiver country code.
- *party_name_anon_o*: a string that represent the party name of the sender of the transaction.
- *party_name_anon_b*: a string that represents the party name of the receiver of the transaction.
- *account_o_anon*: censored version of the sender's account number.

- *account_b_anon*: censored version of the receiver's account number.

It is important to mention that if the same real entity has two accounts at the same bank it will be treated by the architecture as two separate users, for this reason the identifiers for this chosen granularity are the *user_o* and *user_b* features which are created by concatenating the party name, the BIC and the account number extracted from the relative fields.

3.2 Istance-Level anomalies

The first pipeline of the architecture have been already implemented. At the time being, the solution implemented is based on an autoencoder, that reconstruct transactions after projecting them in a latent space. So, receiving a transaction as input will output a score that represents the quality of the reconstruction, where a low value implies one that is very close to the original representation. Assuming that the suspicious transaction are only a small fraction within the whole dataset, the autoencoder will mostly learn the common patterns that describes the regular ones. So, since anomalous transactions do not follow the normal behaviour of the "expected" ones will likely output an high reconstruction error scores. Without going into too much details, a briefly overview of the tree main blocks of this pipeline will be done.

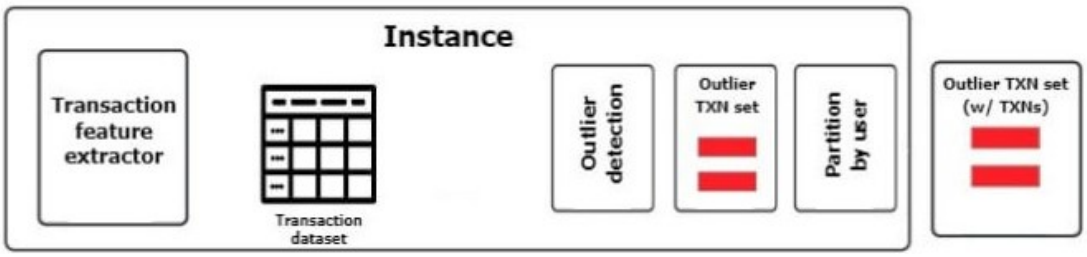


Figure 3.2: The first pipeline detects suspicious users by directly detecting their anomalous transactions.

3.2.1 Autoencoder

The core of this pipeline is the previously mentioned autoencoder. This model can be described by introducing two opposite functions: an encoding function $h = e(x)$ and a decoding one $\tilde{x} = d(h)$. So, denoting a transaction as $x \in \mathbb{R}^n$, the reconstructed transaction $\tilde{x}$ is obtained by chaining the output of e with d, i.e. $\tilde{x} = d(e(x))$. The encoding phase will project x to a latent space that has a lower

dimensionality than the original one; in this way, the reconstruction will, in most cases, be lossy.

The autoencoder proposed in this moment follows a standard architecture since it is comprised by an encoder followed by decoder, that implement respectively the e and d functions. Both the module are composed by various linear layers interleaved with ReLU activation functions. Since the lack of a supervised dataset, the number of layer is at the moment no strictly defined. The autoencoder proposed is trained in a traditional way through the usage of a loss function that represents the quality of the reconstruction of the current batch appropriately defined in order to take into account the different nature of the feature, which can be both binary or continuous. Formally, by denoting as X_c and $\tilde{X}_c$ the continuous feature matrix of the batch in the original space and the reconstructed one

Since the continuous and the binary loss portions can have different importance and/or also to address the fact that these two value can have different ranges of values, the λ hyperparameter can be tuned. The reconstruction error of each transaction is computed according to this loss function, this means that a bad reconstruction will lead to higher value as already mentioned before. The last block of this pipeline is the aggregation one, in which starting from each transaction and the relative reconstruction error, an aggregation policy is employed in order to convert the transaction-level scores to user-level ones. At the time being, a transaction is associated to a user either if he is the sender or the receiver and since each user have a different number of transactions associated, three aggregation policies can be used. Informally:

- Average rating: the score assigned to the user is equal to the average of his transaction scores. The drawback of this policy is the fact that users that have 1 or few transactions with a high reconstruction error will be easily marked as anomalous, while the ones that have many “medium" ones will be flagged as “expected".

- Sum of ratings: in this case the score assigned to each user is the sum of his transaction scores. In this case, those users that have a lot of low score transactions will be still considered suspicious.

- Log-n average ratings: this represents a trade-off between the first two mentioned ones. To each user will be assigned the average score multiplied by $log(n + 1)$, where n is the number of transactions, in this way the we are taking into account also the number of the transaction (i.e. users with a few number of transaction will have a lower multiplying factor).

3.3 User-level anomalies

The first pipeline is able to mark users as anomalous by observing the common pattern of the extracted feature at transaction level. Since some behaviours cannot be captured at that level (i.e. the total amount sent within a time window), the second pipeline has been defined in order to deal with the trends that can be observed only at the user-level. The output of this pipeline will not only produce an outlier user set, but will enrich this with the most relevant transactions that lead the relative users to be labeled as anomalous.

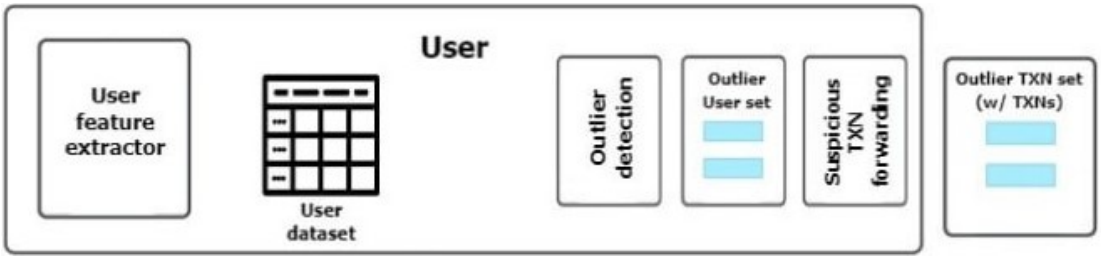


Figure 3.3: The second pipeline. The user dataset is generated through a feature extraction and then it is injected in the outlier detection module. The final output is the outlier users set with their most relevant transactions.

3.3.1 Features Extraction Module

As described before, all the pipelines share a common data loading phase, in which roughly the transaction are loaded in memory. Since the outlier detection methods proposed in this pipeline works on an higher level, a new feature extraction module has been designed and implemented for scratch. This module is able to extract features at different levels of granularity (user, BIC or country) and for this reason is also shared with the last pipeline. It will be discussed in details in Chapter 4.

3.3.2 Outlier detection

The outlier detection module is the core of this pipeline and it basically constituted of an outlier detection algorithm. Due to the absence of labels, supervised or semi-supervised approaches cannot be employed and so far, 4 different models have been proposed:

- Autoencoder: the only deep learning option. It works in the same way of the one employed in the first pipeline.
- Isolation Forest

- One Class Support Vector Machine
- Local Outlier Factor

Each model outputs for each user under test an anomaly score, the higher the score, more suspicious the user is. Also is worth mentioning, that since models like the One Class Support Vector Machine and the Local Outlier Factor, cannot be trained on the whole dataset of users (due to the lack of adequate computational resources) an ensemble approach has been adopted. By specifying an hyperparameter N, such a number of different instances of the same model will be trained, by distributing uniformly the data.

3.4 Collection-level enrichment

The output of this pipeline is to produce a set of descriptors and labels that can be assigned to the users but also to their banks and countries. Since the design and the implementation of this pipeline is the focus of this thesis, it will explained in details in Chapter 5.

Chapter 4

Feature Engineering

The feature extractor was described at very high level in 3.3.1 and since it is a core module for both the second and the third pipeline, is worth describing in details its implementation. By specifying the aggregation level (user, BIC or country), the module will produce a dataset of the desired entities; the set of features extracted are all requested by the domain experts of the partner bank. Some of these higher level features cannot be implemented without extending the instance level feature extraction of the first pipeline; for this reason before describing the feature engineering process of the high level entities, the integration of the new ones at transaction-level will be described.

4.1 Instance-level features extractor extension

The extension of this module includes the implementation of two brand new features: *unexpected corridor* and *closeness to round amount*.

4.1.1 Unexpected Corridor

In the financial context a corridor can be roughly defined as the combination of two countries, one is the country where the transaction is sent and the other one is where it is received. Let O be the country of the originator (of the transaction) and Ob the one of its bank, B the country of the beneficiary and Bb the one of its bank, we can define (by extending the given the definition for our purposes) the following types of corridors: $O - Ob$, $B - Bb$, $O - B$ and $Ob - Bb$. For each of this corridor a relative feature will be computed that will evaluate the degree of unexpectedness of it w.r.t. the distribution of the combination observed in the dataset. Let x be the normalized count for a specific corridor (i.e. Italy-Ireland) in an arbitrary corridor type (i.e. $O - B$), the value of the corresponding feature

will be $1/\log_{10}(x+\epsilon)$. Since we need to compute an unexpectedness value, the feature need to be inversely proportional to x. Also since some combination can be extremely rare, and so having a value of $x \approx 0$, to avoid numerical issues a positive number ϵ is added to it. This feature can be discriminative in the outlier detection task, since anomalous transactions will likely belong to not usual corridors.

4.1.2 Closeness To Round Amount

As pointed out by the domain experts, anomalous transactions often carry out a well “rounded" amount. This features describe how much the original import is close to the next rounded value w.r.t. a specified parameter named *over*. The function used to describe this feature is $\log_{10}(\epsilon + x)$, where:

$$x = \min(|over - (a \mod over)|, a \mod over) \tag{4.1}$$

and where a is the amount moved by the transaction in the original currency. Even in this case ϵ is a positive integer used to avoid numerical issues.

4.2 Users-level Features Extractor

The main entities that are involved in the network described by the transaction database are the users, the banks and the countries; this high level features extractor have the goal of computing the most relevant information about these entities in a specific time window. The features that will be described can be subdivided into two categories:

- **Bidirectional**: these features aim to describe the user/bank/country both as the originator and as beneficiary of the transactions related to it. Each feature that belongs to this category will have a incoming and an outgoing version, in order to describe the two different roles that the user/bank/country can play.
- **Profile-level**: these features aims to describe some information about the profile of the user/bank/country or that is not strictly related to the directional flow of its transactions.

Before describing how these features are going to be computed, is worth showing how many users, banks and countries are involved in each month, which for our study will be the time windows that are going to be considered. Analyzing 4.1, can be observed that the transactions are distributed uniformly over the various months and about the 55% of the users send or receive money in each of these months (and so their banks and countries). Of course, Month 4 will be not considered since only one transaction has been registered within it.

	Month 1	Month 2	Month 3	Month 4	Total
#Txns	2,094,669	2,013,688	2,236,507	1	6,344,865
#Users	1,458,174	1,412,687	1,541,610	2	2,548,772
#Banks	3392	3,418	3392	2	4130
#Countries	235	239	234	2	241

Table 4.1: This table shows the number of transactions, users, banks and countries involved for each month.

In order to work, this module takes as mandatory input the aggregation level (user/bank/country) and the time window (i.e. Month 2). Without loss of generality and to make the following descriptions more intuitive and clear, we will consider:

- User as aggregation level. To make the explanation less verbose we will refer only to the case where the features are extracted at this level.
- Month 3 as time window in order to show the distributions of the feature extracted, on account of the fact that the distribution of each feature is roughly the same across each month. Also since some features are based on the transactions that were executed before the time windows specified, choosing this month allow us to have more data to work one and so the outcome will be more “reliable".
- For the bidirectional feature set, only the active version will be analyzed.

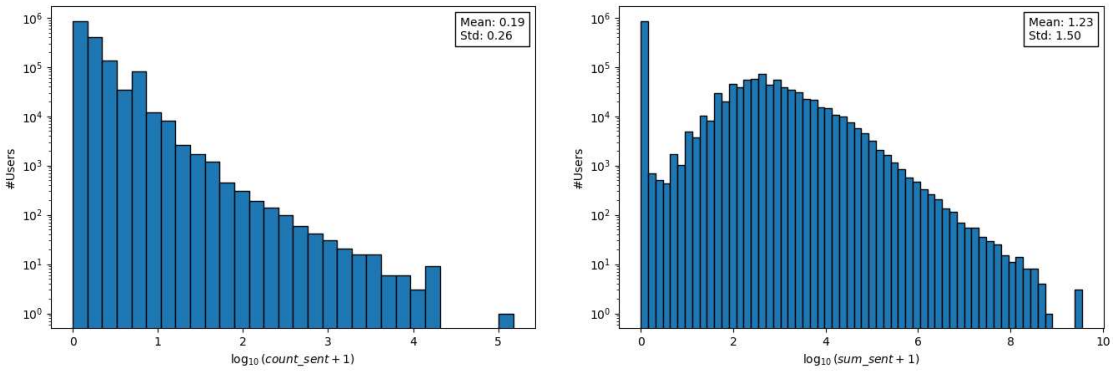
The first features to be described are the bidirectional ones.

4.2.1 Total Amount

The *Total Amount* is actually a group that comprises four bidirectional features used to describe the money flow of the user within the time window. These features are:

- **Amount Sent**: represents the $\log_{10}$total amount of money sent by the user, i.e. is obtained by summing up the *amount* field of the raw transactions in Euro. Since there might be users that never sent a transaction, in order to avoid numerical issues the value of 1 is added before applying the logarithmic function.
- **Count Sent**: represents the number of transaction executed by the user.
- **Mean Sent**: represents the mean of the amount sent by the user at each transaction, i.e. is obtained by dividing the previous two.

- **Std Sent**: represents the variability in the amounts sent by the user, i.e. is the standard deviation of the *amount* field. The users that never sent a transaction or sent just 1, will have a default value of 0.



(a) **(b)**

Figure 4.1: Histograms that show the distribution of the $\log_{10}$ version of Count Sent and the Amount Sent features (left and right respectively) at user level in the month of Month 3. It is important to observe that almost 1 million of users never sent a transaction. To obtain a better visualization of the distributions, the transformations specified on the x-axis has been applied.

By observing the power low distribution of the feature **Count Sent** in figure 4.1, can be deduced that many users in the network only have a passive role since are only receiving transactions; this statement is also truth for the other direction: many users only play an active role without receiving any transaction at all. Also the majority of users within a single time window are related only to one transactions: this means that almost all user send/receive only one transaction while receiving/sending none of them. As a consequence of this common pattern, the distributions of Mean Sent and Std Sent, that will be not shown can be easily inferred: the first one will have the same shape of Amount Sent while the second one will have almost all values in the 0 bin. Additionally the common pattern mentioned before, will lead the final feature vector to somehow sparse: only one part of the bidirectional features set will not have default values.

4.2.2 Growth

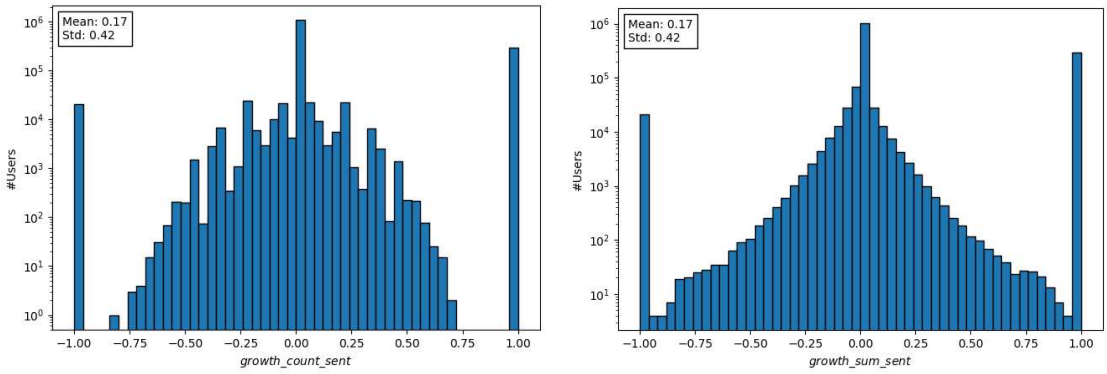
As the previous one, *Growth* is a group of four different features, that are used to describe how the financial flow of the user within the time window considered has

changed w.r.t the past. To compute them we need to firstly compute the *Total amount* group as if we were analyzing the previous time window (i.e. if Month 3 is the current time window, the previous one is Month 2), and by having also the standard one, we can compute each variation using this formula:

$$\frac{x_c - x_p}{x_c + x_p} \tag{4.2}$$

where x_c represents the feature computed in the current window, and x_p stands for one in the previous window. There are various ways to express user growth; this particular normalization was chosen because even in cases where the user has not submitted transactions in either of the two windows (past or current), it is possible to determine a real value. For instance, the simple ratio between the current and past windows could be indeterminate if the latter is empty.

The name of these features are **Growth Amount Sent**, **Growth Count Sent**, **Growth Mean Sent** and **Growth Std Sent**. The values of these features are, by definition, always between -1 and 1; a positive value of the growth shows a positive trend of the user w.r.t. to the past while a negative value the contrary. An extreme value of -1 can be found within the user that were active in the past but not now, while 1 can be found in the mirror case. Another particular value is 0, it appears when the user did not send any transaction at all, in the union of the two time windows. As can be observed in 4.2, the most frequent values are 1, -1

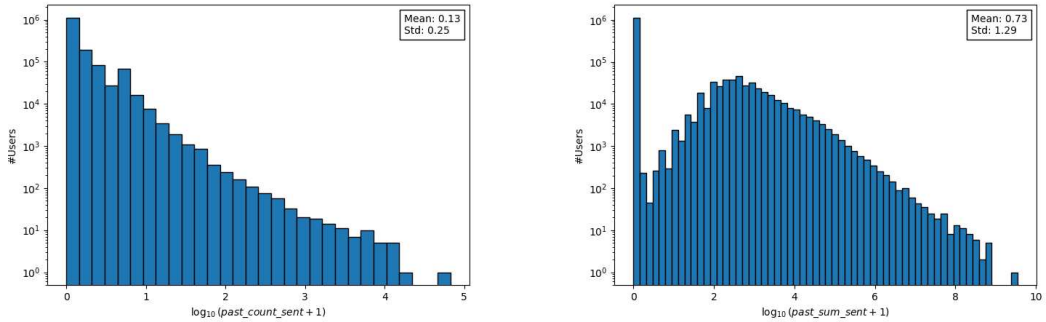


(a) **(b)**

Figure 4.2: Histograms that show the distribution of the Growth Count Sent and Growth Sum Sent at user level in the month of Month 3.

and 0: this means that the majority of users have sent transaction only in Month 3 or Month 2, or have almost sent the same number of transactions and amounts in these month (this is also valid for the “received" version). The formula used

to compute the growth features is one of the many options that can be employed to estimate the variation between two observations, another simple alternative can be the ration between them. However, keeping in mind the considerations about the user database depicted previously, in order to avoid indeterminate results (i.e. divisions by 0), the proposed solution is a very good one, since can always determine a value. Another way, to effectively,describe the growth is by introducing directly the *Total amount* features computed in the past; in this way we let to the employed models (especially to the deep learning ones that can be used in the outlier detection block of the second pipeline) to put indirectly in relation the current user behavior with the past one. These features are named **Past Amount Sent**, **Past Count Sent**, **Past Mean Sent** and **Past Std Sent**.



(a) **(b)**

Figure 4.3: Histograms that show the distribution of the $\log_{10}$ version of Past Count Sent and the Past Amount Sent features (left and right respectively) at user level in the month of Month 3. This plots can be seen also as the Count Sent and Amount Sent that would have been obtained in the month of Month 2. To obtain a better visualization of the distributions, the transformations specified on the x-axis has been applied.

4.2.3 Rapid

The bidirectional **Rapid** features measures the frequency at which the transactions are sent by the user within the time window specified. The feature is simply calculated as the Count Sent feature over the days of the time window, i.e. it represents the mean number of transactions executed by the user each day. This feature is simply a scaled version of Count Sent, and since it is a redundant information is not interesting at all, for this reason the relative histogram will be not shown.

4.2.4 Count of BICs

The bidirectional feature **Count of BICs Sent** depicts the number of distinct banks (identified by their BIC) to which each user has sent its transactions. Since

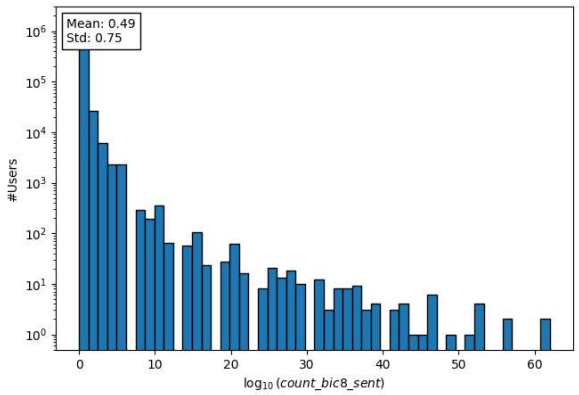


Figure 4.4: Histogram that shows the distribution of the Count of BICs Sent feature at user level at user level in Month 3. To obtain a better visualization of the distribution, the transformation specified on the x-axis, has been applied.

there number of transactions sent by a user cannot be lower than this feature and also taking into account the nature of the dataset as highlighted before, the distribution of this feature is a power low which case shrinks faster w.r.t the one that describes Count Sent.

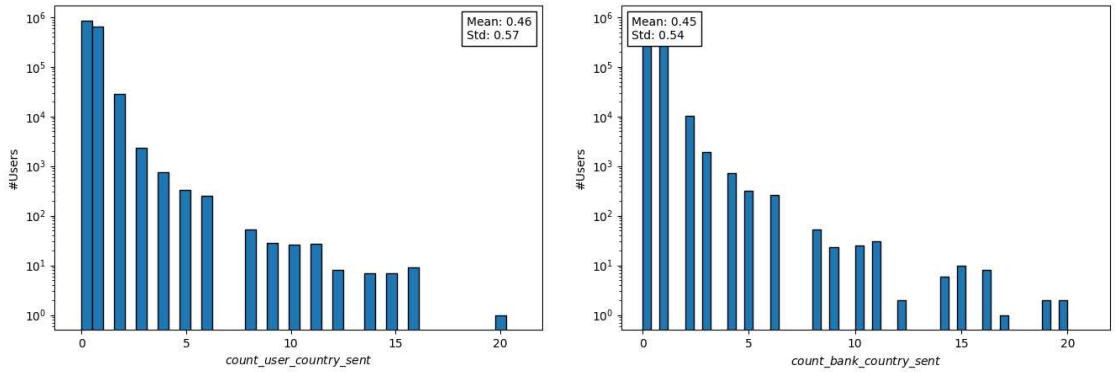
4.2.5 Count of Countries

Count of Countries is a group of two features, **Count of Users' Countries** and **Count of BICs' countries**, which respectively represents the number of the distinct countries of the users and banks to which each user sent its transactions.

Even in this case the distribution of the feature (4.5) is a power low one that expires faster than the Count Sent one. It is important to point out that only a very few percentage of users have a value higher than 1 for these features, roughly 30000 and 10000 respectively.

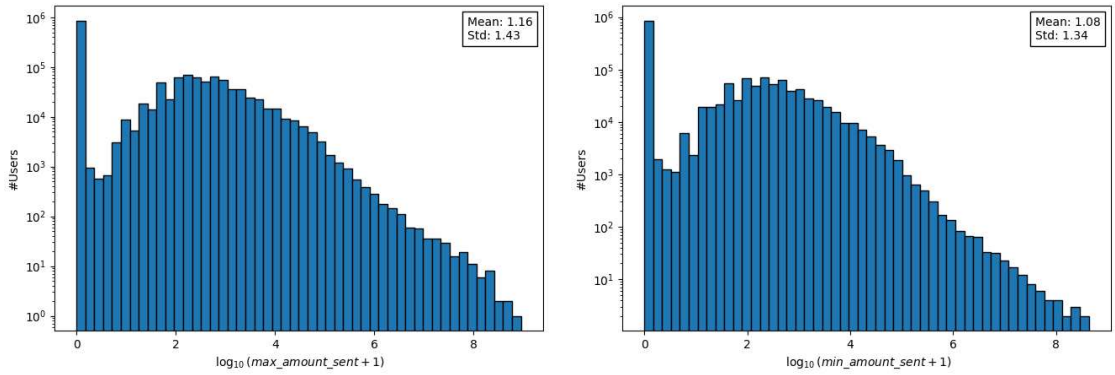
4.2.6 Min-Max Amount

The **Max Amount Sent** and **Min Amount Sent** features as the names suggest, describe the minimum and maximum amounts carried out by the transactions sent by each user.



(a) **(b)**

Figure 4.5: Histograms that show the distribution of the $\log_{10}$ version of Count of Users' Countries and the Count of BICs' countries (left and right respectively) in the month of Month 3. Only more than 30000 and 10000 users, respectively, have a value higher than 1.



(a) **(b)**

Figure 4.6: Histograms that show the distribution of Max Amount Sent and the Min Amount Sent (left and right respectively) at user level in the month of Month 3. The distribution is very similar to the one that can be observed in Amount Sent. To obtain a better visualization of the distributions, the transformations specified on the x-axis has been applied.

Both the distributions (Figure 4.6) shows a behavior very close to the Amount Sent one (Figure 4.1); this is due to the fact that many users have sent only one

transaction (or any at all) and so the minimum and the maximum amounts are the same and also is trivially equal to the sum.

4.2.7 Past Active

The number of days elapsed between the last transaction sent by the user before the start of the current time window and the first transaction sent within it, is described by the **Past Active** bidirectional feature. This feature is useful in order to describe how long a bank account remained inactive, and can be used to highlight the user that wait a long period of time to sent two consecutive transactions (accounts that are involved in international terrorism financing usually operate in this way). Users that never sent a transaction in the current time window or in the past one will receive a default value equal to 62, which is two times the length of the longest month that can be analyzed. It can be observed in Figure 4.11, that Month 1 many

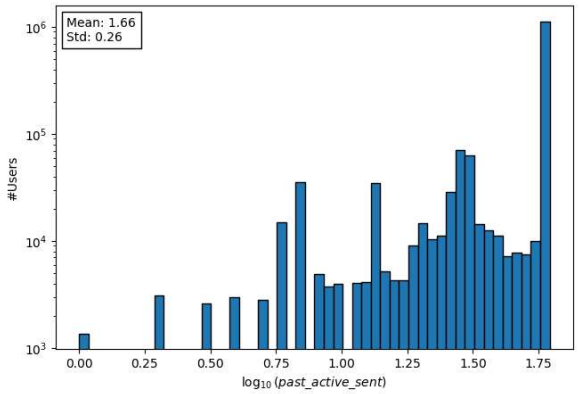


Figure 4.7: Histogram that show the distribution of Past Active Sent at user level in Month 3. The highest spike represents to the default value assigned to the many users that never sent a transaction in Month 3 and/or in the past. To obtain a better visualization of the distributions, the transformations specified on the x-axis has been specified.

users never sent a transaction even in the past.

4.2.8 Mean Time Between Transactions

The **Mean Time Between Transactions** bidirectional feature, measure the mean number of days that elapse between two consecutive (in time) transactions sent by the user. It's possible to proof that the value of this feature is equal to number of days elapsed between the first transaction sent by the user and the last

one within the current window divided by the overall number of transactions sent. Users that sent only one transactions or any at all will receive a default value of 0.

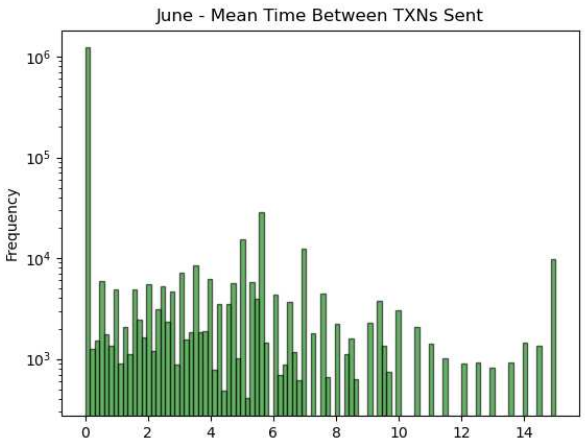


Figure 4.8: Histogram that show the distribution of Mean Time Between Transactions Sent at user level in Month 3. Excluding those users that received the default value of 0, the value of this feature is roughly uniformly distributed.

4.2.9 High-Level Unexpected Corridor

High-Level Unexpected Corridor is a group of two features denominated **Unexpected Corridor O-B Sent** and **Unexpected Corridor OB-BB Sent** which are computed relying on the relative values obtained at transaction level (4.1.1). The values of these features are obtained by summing up the values of unexpectedness of the O-B and Ob-Bb corridors. For both channels, as it can be seen in Figure 4.9, the majority of the users have a relative low degree of unexpectedness, while only a few percentage (about 2500 ones) have a very high one.

4.2.10 Funneling

In the context of financial crimes, “funneling" refers to an illegal practice where funds or assets are transferred through a series of complex and intricate transactions in order to conceal the source of the funds or for money laundering purposes. The *Funneling* is a profile-level group of feature that aims to estimate the “funneling" degree of each user. The two features that belongs to this group are **Funneling User** and **Funneling Bank** which measure in $\log_{10}$ scale respectively the ratio between the number of distinct users/banks to which the transactions are sent and the one from whom transactions are received; in order to avoid numerical

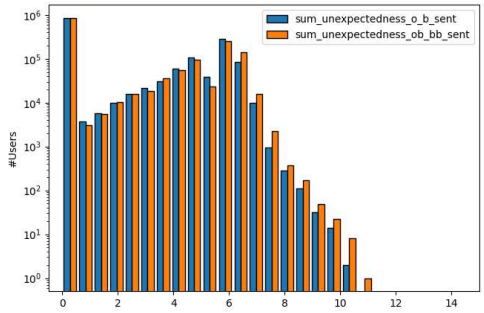


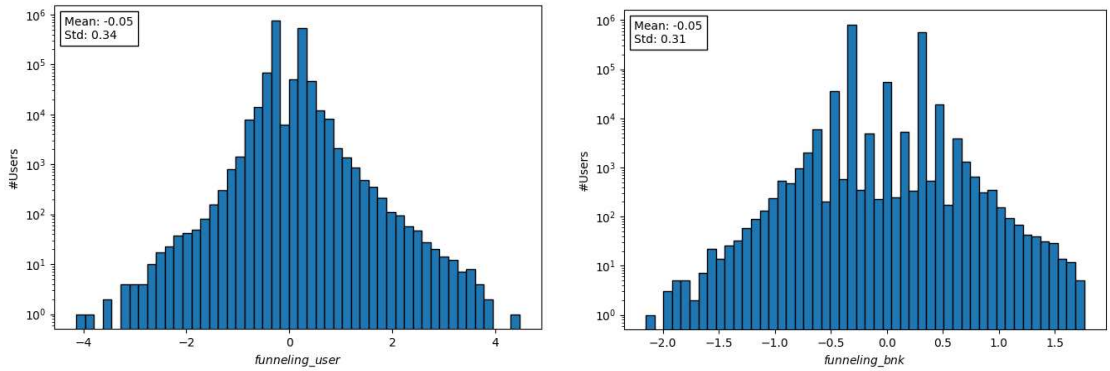
Figure 4.9: The plots shows both the histograms of Unexpected Corridor O-B Sent and Unexpected Corridor Ob-Bb Sent (respectively in blue and orange).

issues the value 1 is added to both the numerator and denominator. By using the logarithmic scaling the following properties are achieved: a positive value shows that the user sent transactions to many other users respects to the ones from which it receive them, while a negative value shows the opposite trends; also in this way both sides have a fair “ranges" of values(without using the logarithm, the actual negative values would have been shrinked between 0 and 1, while the positive value would have been higher than 1 without an upper limit). An high absolute value of this features highlight the users that acts as a “funnel" for the money flow, which is a typical behavior of the money launders. The Figure 4.10, shows the distributions of these two features. If we take only the positive (or negative) slice of the histograms, the funneling index follows a power low distribution: meaning that there are only very few users that acts like funnels. Also the highest spikes for both sides represents the majority of users that sent or received only one transactions in the month of Month 3.

4.2.11 Motifs

Smurfing is money laundering technique exploited by criminals that involves splitting up high amount of money into a set of multiple transactions that carry small amounts. The detection of smurfs can be tackle by translating the interactions of the various entities into a graph, in which the nodes are the financial actors and the links represent the money exchange between them. There are differents smurf-like motifs, the three that we want to represents are:

- $N-1-M$, this motif consists in a set of N source nodes that inject money to a middle node, that in turn will sends the amount to a set of M target nodes.
- $1-N-1$, this motif consists in a single source node that sends money to a



(a) **(b)**

Figure 4.10: Histograms that show the distribution of Funneling User and the Funneling bank features (left and right respectively) in the month of Month 3. The high spikes around the value 0, represents that users that sent or received only one transactions.

set of N middle nodes, that in turn will resend the amount to a single target node.

- *U-turn*, in this motif the source node send money to another node that later will send them back to it.

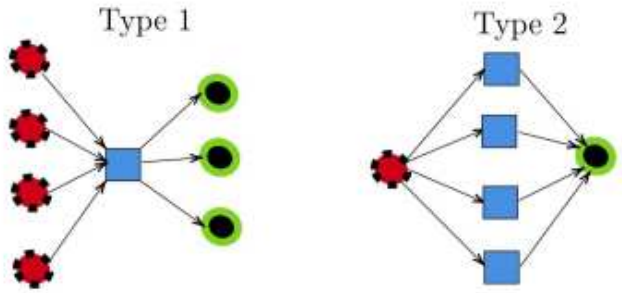
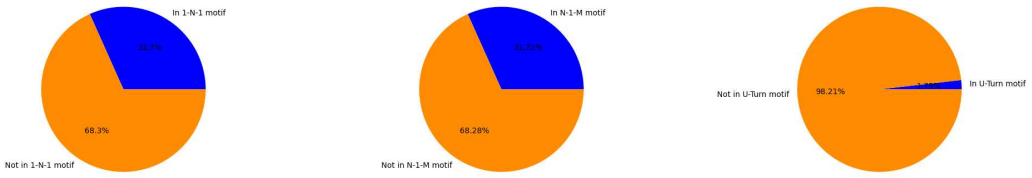


Figure 4.11: Illustration of smurf-like motifs $N-1-M$ (on the left labelled as Type-1) and $1-N-1$ (on the right labelled as Type-2), where the source nodes are represented as red circles, the middles ones as square and the targets green circles. Figure taken from [19].

The feature *motif_1n1*, *motif_n1n* and *motif_uturn* measure for each user the number of his transactions that belongs to the these motifs, that can be detected

using a graph representation as mentioned before. Since there are few users that acts as *smurfs* and these set of features seems to be not very useful for the formation of clusters, we will not discuss in deep the implementation of the detection process. The extraction of these patterns are showed by Starnini et al. [19].



(a) **(b)** **(c)**

Figure 4.12: Pie charts that show the share of users that are occuring respectively in the $1-N-1$, $N-1-M$ and *U-turn* motifs in Month 3.

4.2.12 Country-Region

The countries and the continents of the users and their relative banks have been encode as 1-hot encoded vectors. Given the wide variety of countries that appear in the dataset, only the top-N countries (i.e. the N countries that are most frequent in the current window) have been explicitly represented, while the other countries are encoded in a shared *<other>* bucket. Since, for some users is not available the belonging country, the *missing* category will be treated as it were one. Choosing $N = 9$, the 4.2 shows the distribution of the encoded countries for users and banks. The 4.3 shows the distribution of the encoded regions for users and banks.

4.2.13 Country counters

The country counters feature is a set of feature that measures the share of transactions sent by each user towards a specific country of users and their banks. As in the previous case only the N most frequent countries will be taken in considerations, while the other ones will be represented as a one, the $< other >$ category. Since these feature represents a percentage, the values will lies in the range from 0 to 1. Choosing $N = 4$, the 4.6 shows the numbers of users that sent at least one transactions towards the encoded countries.

	Users	Banks
IT	0.40	0.40
IE	0.26	0.27
RO	0.16	0.16
DE	0.03	0.03
FR	0.01	0.01
BE	0.01	0.01
NL	0.01	0.01
ES	0.01	0.01
LT	-	0.01
missing	0.03	-
$< other >$	0.07	0.07

Table 4.2: Recap of the most commonly found countries, in percentage, of users and banks in Month 3. For each column, the top-9 (most frequent) countries have been included. Transactions not belonging to those 9 countries have been included in the $< other >$ category. The missing category represents those users for which the country information is not avaiable.

Regions	Users	Banks
Europe	0.95	0.97
Asia	0.02	0.02
Americas	> 0.01	> 0.01
Africa	> 0.01	> 0.01
Oceania	> 0.01	> 0.01
missing	0.03	> 0.01

Table 4.3: Distribution of the regions among users and countries. The missing category represents those users or banks where the country region was missing or the mapping operation between country and region is not available.

4.2.14 High-Risk Geography counters

Each country is associated to a High-Risk Geography level, that can be *low*, *medium* or *high*. The High-Risk Geography counters is a set of three features that measures the share of transactions sent by each user towards these three categories. Since these feature represents a percentage, the values will lies in the range from 0 to 1. The 4.6 shows the numbers of users that sent at least one transactions towards a specific HRG level.

Countries	Users	Banks
IT	0.32	0.33
IE	0.31	0.32
RO	0.24	0.25
DE	0.01	0.01
$< other >$	0.12	0.13

Table 4.4: The table shows the percentage of users that have sent at least one transactions to the encoded countries for users and banks. The not frequent countries are represented in the $< other >$ category.

HRG Levels	Users	Banks
Low	0.92	0.96
Medium	0.06	0.02
High	0.02	0.02

Table 4.5: The table shows the percentage of users that have sent at least one transactions towards the three HRG levels.

4.3 Bank-Country Features Extractor

Bank and Country level features can be extracted in the same of the user ones, i.e. the Count Sent for a bank can be the total number of transactions that sent in the window. However there are several reasons why this strategy can be detrimental for the clustering process and for the final purpose of this pipeline. Indeed:

- Either the banks and the countries do not have a balanced number of transactions associated to them; most banks and countries receive or sent very few transactions, in many cases only one. Computing the aforementioned features in the same way, by changing only the aggregation level, will lead to a result where the most frequent banks and countries will be treated likely as outliers, showing very noisy features' values.
- Even if we consider a uniform distribution of transactions among banks and countries, it can happen that for some entities, a significant portion of these movements is associated with the same user who is transferring a large sum of money. This could lead to characterizing the bank or country based on the behavior of that individual user. In this scenario, other users cannot be effectively represented.
- For the same reason discussed before, there is the need to represents the banks

and the countries by summarizing the behavior of the belonging users with other strategy like computing the mean or the median features wise. In this way the descriptions of this higher level entities will be very close to the most frequent patterns that can we observe at user level.

- The final goal of the architecture is finding anomalous user and to provide useful insights that can help the final human-operator. By describing the banks and the countries by the behavior of the average users, can be more useful to define the profile of the suspicious users.

#Users	Banks	Countries
Equal to 1	0.24	0.06
Between 2 and 10	0.36	0.22
Between 11 and 100	0.26	0.34
Higher than 101	0.15	0.37

Table 4.6: The table shows the how many banks and countries have a number of users, in percentage, associated that lies in a specified range (i.e. the number of users equal to 1).

At this point, there are two possibility that we can take into consideration to compute the features at these aggregation levels, having for each bank/country the set of users that belongs to them:

- Each feature is expressed as the *mean* of the same feature on the set of users. This is suggested when the distribution of the feature among the users follows a normal shape.
- Each feature is expressed as the *median* of the same feature on the set of users. This is strongly recommended when the distribution of the feature among the users is a power-law one.
- A mix of them, considering the distribution of each feature.

Given how the features are distributed the best choice for the considered time windows is the second one. However, the one hot encoded features cannot be represented in this way: if no of the bucket within the same feature shows a population higher than the half of the whole one, the final vector will be filled by zeros (i.e. if a bank has 10 users and each user have a different country residency, the median of each bucket will be 0 and the aggregated final feature will not bring any value). For this reason, the country/region residency/bank features, will be computed by observing the most frequent country/region among the set of users

related to the current entity (in case of tie, the most frequent country/region among the whole dataset will be picked). We can observe trivially, that:

- The country and the region at bank level, for each bank will certainly correspond to its country and region.
- The country at country level, will correspond itself. Since the relative 1-hot encoded vector is able to represent only the top N countries plus the $<$ *other* $>$ category, these feature will be dropped; otherwise the clustering process will favor the “infrequent" countries to be grouped together.

Chapter 5

Collection Level-Enrichment Pipeline

In this chapter will be explained the design and the implementation of the third pipeline. To do so, a briefly introduction to its skeleton will be made.

5.1 Overview

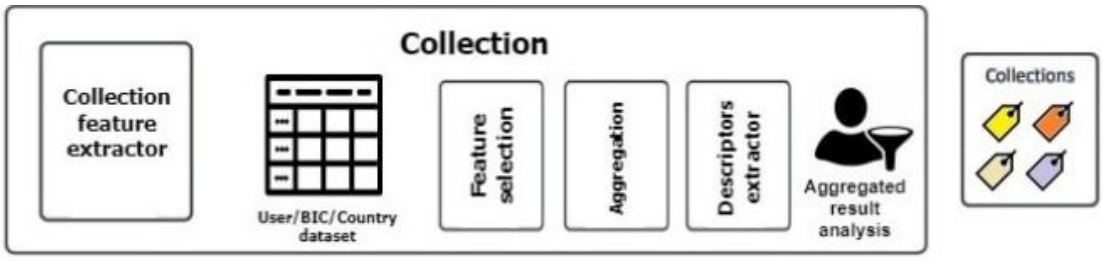


Figure 5.1: The third pipeline with its main components. Its input is the batch of transactions previously loaded (not represented in the figure).

The collection enrichment pipeline (shown in 5.1), which takes as input the batch of transactions under study, comprises the following components:

- Features extractor: by specifying the *aggregation level* parameters, this step allows to compute the features at user, BIC or country level, it is also mandatory to specify the time window in which them will be computed. Optionally, other parameters can be specified to influence the outcome of some features like the one hot encoded ones (i.e. the number of the most frequent countries to consider).

- Aggregation/Clustering module: in this step, a clustering algorithm will run taking as input the entities previously extracted. It is possible to specify the algorithm to run (i.e. K-Means, DBSCAN or OPTICS) and depending on the strategy chosen other hyperparameters can be passed (i.e. the K for K-Means).

- Dimensionality reducer: since the entities are described by a multitude of features, there is the need to reduce the dimensionality of the data by dropping highly correlated features or trough the usage of PCA.

- Descriptors extractor: to make the clustering results interpretable, i.e. to better highlight the commonalities among the entities that have been clustered together, this step need to be employed. This extractor in turn, includes two sub-steps: a quantization and a frequent itemsets extraction ones. The former by mapping the continuous features into categorical ones prepares the cluster to be analyzed by the latter. Its possible to specify the quantization strategy, which will be discussed in depth later, and the algorithm that will extracts the itemsets along with the minimum desired support.

- Aggregated result analysis module: so far, all the steps involved in this pipeline are automated, however there is no a “domain" meaning behind the clusters detected. This step, requires human intervention and aims to produce a small number of clusters that will be studied and labelled. After this step, we have a set of labels that can be assigned to users and their banks and residency, which are representative of some behaviors that can help the further investigation of the suspicious users detected.

Following, will be presented in depth the implementations of these modules, by discussing also all the possibilities that have been explored during the thesis work highlighting the pros and cons of each one.

5.2 Features Extraction

The feature extraction module behave very similarly to the one presented in 3.3.1, however this pipeline allows to describe other higher level entities like the banks (identified by their BICs) and the countries of residence of the users. The main parameters that this module takes as input are:

- **Aggregation level**: a string that can be either *user*, *bank* or *country*, which serves to specify which entities (users, banks or countries) will be extracted, through the features presented in 4.

- **Raw transactions**: the transactions on which the user features will be computed. Additionally, its possible to associate each user to its bank and residency and each bank to its country.
- **Country-Region mapper**: the object that allows to map each country to its region (Europe, Americas, Africa, Oceania, Asia).
- **User-level features**: if the aggregation level is set either to bank or country, pre-computed user-level features need to be passed as input. In this way we can recycle the dataset of users extracted in a previous run of this pipeline at this level, saving a lot of resources.
- **Start date** and **end date**: two datetime objects which describes the time window in which the features will be computed. This means that, if the aggregation level is set at user, the batch of transactions passed as input will be filtered by mantaining only those transactions in which the *data_ref* field is between these dates (i.e. by setting the start date to $2022-04-0100:00:00$ and end date to $2022-04-3023:59:59$, only the transactions executed in April 2022 will be kept). Of course, the start date should precedes the end one. If the aggregation level is set to bank or country, these parameters will be ignored (this means that the precomputed user-level features to pass should have been computed on the desired time period).

There are other secondary parameters that may be specified (otherwise the default values will be retained) are:

- **Top-N countries**: it specify how many of the most frequent countries should be explicitly encoded in the relative 1-hot encoded vector. The default value is set to 9.
- **Top-N country counters**: it specify for how many of the most frequent countries the feature described in 4.2.13 should be computed. The default value is set to 4.

By keeping the default value of the secondary parameters each entities (user, banks and countries) are described by 121 features, where 87 are continuous and 34 are 1-hot encoded bits.

5.3 Feature selection/Dimensionality reduction

Clustering algorithms are distance-based, in order to measure how similar two points are a distance metric is employed. As discussed in 2.2, the curse of dimensionality phenomenon, can make most of these metrics not very significant. Since our data

are described by a reasonably high number of features, its highly recommended to compress them in order to achieve a better clustering results. This goal can be achieved, in two ways: dropping highly correlated features and/or using PCA.

5.3.1 Correlated features

The insights drawn by the distribution of the user-level features presented in 4, show that most users send only one transactions while receiving none, and vice versa. More specifically, referring only to the former without loss of generality, they have all the passive features (the ones that describe the users as the beneficiary of the transactions) set to default values (this is also true in the mirror case). Formally, the number of transactions sent (or received) is correlated to many features, i.e. the number of distinct banks towards which the user sent his transactions cannot be higher than it, and in many cases represents an upper bound. The correlations between these features is not-linear by definition (excluding the *count_sent* and *txns_frequency_sent* pair), however the correlation heatmap proves otherwise. By observing 5.2, we can observe that there two main groups of correlated features, in depth:

- The first main group that can be observed on the top left of the heat map, comprises features like *count_sent*, *sum_sent*, *count_bank_country_sent*, *count_user_country_sent*, *max_amount_sent*, *min_amount_sent*, etc..
- The second smaller group can be observed is on the bottom right of the heat map. These group is described by the features that describes the behavior of the users in the past, like *past_count_sent*, *past_sum_sent*, *past_mean_sent*, etc..
- The two groups are also correlated together, probably due to the fact that behavior of a user is staple across different time windows (i.e. if a user did not send any transaction in the past, likely he will not do it in the future).

The fact that a great number of users, only sends or receive transactions suggest that the this behavior are anti-correlated, indeed the Pearson correlation coefficient between *count_sent* and *count_receive* pair its around -0.55. The *funneling_bank* (or *funneling_user*) feature is very effective in representing the nature of user: a value higher than 0 indicate that the user only sent transactions, while a value below 0 indicate the opposite case. More specifically, the Pearson correlation coefficient between this feature and the *count_sent* one is around 0.81; very close values can be observed by computing the metric also on the features that are highly correlated with this last one. The correlation metric computed with the mirror features (the “received" ones) are around -0.80, demonstrating that a very

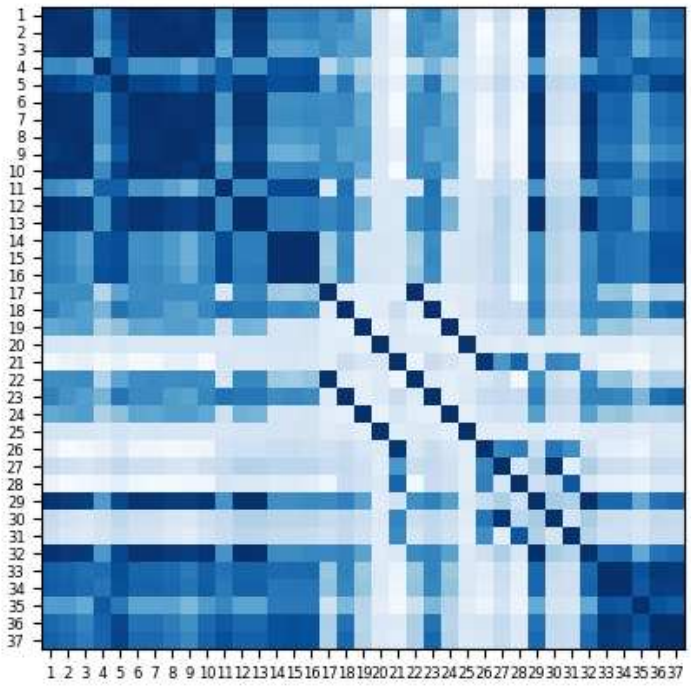


Figure 5.2: Heat map of the bidirectional “sent" features on users.

high number of feature can be summarized using only one, despite them describe opposed aspects of the user profile. So far have been discussed and highlighted the presence of groups of correlated features, but how we can pick among each group the feature that will represent it? Two possibilities are proposed:

- **Manual choice**: given the heatmap, the operator selects manually the features to retain. The main advantage of this strategy is the fact that it is simple and there is a full control over the process, while a deep domain knowledge poses a huge drawback. Also, since the distribution of the features can evolve trough different time windows this choice it does not fit into an automated scenario.
- **Feature selection algorithm**: the heatmap is passed as input in order to estimate through an heuristic the most representative features, which roughly are the ones that show the higher average correlation. Its main advantage is certainly the fact that the features are selected in a fully automated way, while the biggest drawback as it will be discussed later its the need of a threshold that can affects the final results.

Feature selection algorithm

The algorithm employed to perform feature selection its inspired from CORR-FS (Correlation-Based Feature Selection) proposed by Giobergia et al.[20]. This algorithm is able to identify a set of independent features, within which all the features have among them a Pearson correlation coefficient lower than a specified threshold r_{min}. Also this algorithm ensures that all the features that not belongs to the returned set, are correlated with a feature within it with a coefficient higher or equal than r_{min}. The steps that characterize CORR-FS are:

1. Given the dataset of entities, the absolute value of the Pearson correlation coefficient is computed between each pair of features i and j as $r_{i,j}$.
2. The list L that represents the "remaining features" to choose is initialized to all the features that describes the data.
3. For each feature i in L, the sum of squared correlation coefficient s_i is computed as $s_i = \sum j \in L r_{i,j}^2$
4. Than the variable b computed as:

$$b = \arg\max_{i \in L} s_i \tag{5.1}$$

   represent the most representative feature left.
5. All variables $j \in L$ that $r_{j,b} > r_{min}$ are removed since they are also well represented by b. This step guarantees that at least one feature is removed since $r_{b,b} = 1$.
6. If L is empty the algorithms ends otherwise it restart from Step 3.

5.3.2 Principal Component Analysis

Another way to reduce the dimension of the dataset is by performing the well-known PCA. By employing this analysis is possible to reduce the number of dimension to a desired one or by retaining the number of dimension that guarantee at least a specified cumulative explained variance ratio. In both cases, its extremely important to perform on each features (excluded the 1-hot encoded ones) a standardization step and the aforementioned feature selection one. The former is mandatory since PCA is sensible to to the scale of the data (and also to their "center" as explained in 2.2.2), otherwise it will give more importance to the variable with larger standard deviation; the latter its also important since correlated features can amplify the variance of the principal components. To better understand this phenomenon, the following experiment will be conducted by picking up three features that are

highly correlated among them and a forth one which is independent, them are respectively *count_sent*, *count_bank_country_sent*, *count_user_country_sent* and *past_active_sent*. By computing the PCA only on the first and the last features which are totally uncorrelated, the contribution, that can be measure in terms of the explained variance of each component, is almost identical. After adding a variable correlated with the first one, the variance of the first component is now twice the size of the second. After adding another correlated variables, the size of the first component is now three times bigger than the second; the results are shown in 2.6. This means that, by retaining correlated features, the contribution of each variable will be indirectly unbalanced in the clustering process, since the formation of clusters will most likely based only on those that are correlated.

5.4 Aggregation/Clustering module

The data preprocessed by the previous step are now taken as input by this module, who will build clusters, whose number depends on the clustering algorithm and/or on the set of hyperparameters chosen. Even though clustering methods such as DBSCAN and OPTICS have been explored, the only algorithm capable of generating “meaningful" clusters is K-Means. It is also the only usable algorithm when the entities to be grouped are users. Since, on average, there are more than 1 million users for the considered time windows, density-based algorithms with memory complexity that is, in the worst case, $O(n^2)$ cannot be executed without using a specific hyperparameter configuration. Furthermore, using these algorithms for other higher-level entities leads a significant portion of banks and countries to be considered as outliers. Even if the latter could be assigned to a specially created cluster, the points within it are unlikely to share a similar structure, decreasing the overall result quality.

As is well known, the major disadvantage of K-Means is the need to specify the input hyperparameter k, i.e., the number of clusters to form. Since this value is difficult to predict in advance, the following algorithm has been employed to identify the best cluster configuration:

- Two positive integers, k_{min} and k_{max}, are specified as input, such that $1 < k_{min} < k_{max}$.

- K-Means is run for each value i ranging from k_{min} to k_{max}, resulting in the cluster configuration G_i. For each configuration, the corresponding Silhouette score s_i is calculated. If the entities involved are users, only an estimate of this metric will be performed, calculated on only 10 percent of the data randomly selected.

- The configuration G_k is chosen such that $k = \arg\max_{i \in [k_{min}, k_{max}]} s_i$. G_k is then the final result of this module.

In this case, the choice of the range in which the final value of k will be selected is crucial. That is, if k_{max} is set too high, it will likely result in the formation of very small and less meaningful clusters with a consequent high Silhouette score. Therefore, it is essential to have prior knowledge of the characteristics of the entities that will be aggregated (such as population) to achieve satisfactory results.

5.4.1 Outlier model

Although this pipeline is used to characterize the entities under examination, the result of the clustering phase can be utilized as an outlier detection model to be employed in the second pipeline. In the literature, there are various definitions for the term anomaly/outlier: one of these describes such points as those that exhibit behavior significantly different from the behavior of the points in the considered dataset. While in density-based algorithms like DBSCAN, the concept of an outlier is naturally defined, for K-Means, it is not (all points are assigned to a cluster). In this regard, a simple heuristic has been employed, which marks points as anomalies if they are farthest from their respective centroids. More specifically:

- Consider the most promising result in terms of Silhouette score among multiple runs of K-Means, as described in the algorithm outlined earlier (5.4).
- Each entity i is assigned an anomaly score s_i, equal to the distance of that entity from its centroid c_i.
- By specifying a contamination parameter c, which is a number between 0 and 1, the top K entities with the highest anomaly scores are flagged, where $K = \frac{N}{c}$ and N is the number of entities considered.

Another possibility is to consider the anomaly score s_i of each entity i equal to its Silhouette score. However, this option is computationally expensive, specifically $\mathcal{O}(n^2)$, as it requires calculating distances between all points. This new outlier model, which could be integrated into the second pipeline, despite its simplicity, has the significant advantage of not reducing the overall execution times of the entire architecture since the aggregation phase of this pipeline and the outlier detection phase of the second one are executed almost simultaneously.

5.5 Descriptors extraction

Interpreting the result of a clustering algorithm, especially when points are described by a relatively high number of features, can be challenging. To identify common

characteristics within each cluster, the objective of this step is to find a list of traits (e.g., *user_country=IE*) that are frequent within them. To achieve this goal, algorithms for extracting frequent itemsets like FP-Max have been employed. Since these techniques can only operate on transactional data, i.e., data described by categorical features, it is necessary to map the continuous features of entities into categorical features. In the next subsection, various possible strategies for performing this operation will be illustrated. Subsequently, various techniques explored for the extraction of frequent itemsets will be discussed.

5.5.1 Binning strategy

Binning is a method for mapping the values of a continuous feature into a one-hot encoded vector, where the bits indicate whether the original value belongs to a specific interval called “bin". These intervals should not overlap, and if combined, they must align with the domain of the continuous feature. The main strategies, for which the implementation is already available, include:

- **Uniform**: all bins have equal length. This strategy, while usable, would lead to the formation of extremely imbalanced bins. The features describing entities are distributed according to a power-law, so it is highly likely that all values will be mapped to the same bin.

- **Quantile**: all bins have approximately the same number of points. For example, assuming you want to identify 10 intervals, mapping the continuous variable to each bin would associate around 10 percent of the points with each bin. In this case, the power-law distribution of features makes this goal impossible. For instance, considering that about half of the users do not send transactions, the only way to quantize such a variable is to choose 2 as the number of output intervals (which is quite limiting).

- **Manual**: the intervals are predefined in advance. However, for this strategy to be effective, a deep understanding of the domain is required.

Given the peculiar nature of the involved features, a custom discretization strategy has been designed and implemented based on the idea of quantiles. The goal of this custom technique is indeed to obtain bins that are populated as optimally as possible, reserving specific bins for those duplicated values that make fair division impossible. Formally, the algorithm used for each feature F can be described as follows:

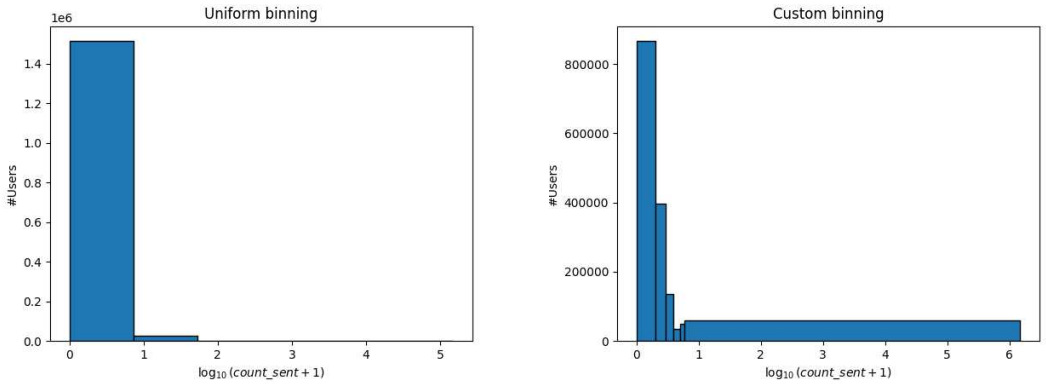
- Let B be the number of bins to be identified, and N be the number of data points. Let n_v be the number of occurrences of the value v for the variable F.

- Let N_l be the number of elements not yet assigned to a bin, and B_l be the remaining number of bins to identify at each execution step. Initially, we have $N_l = N$ and $B_l = B$.

- In this first step, special bins will be assigned to values v such that $n_v > \frac{N_l}{B_l}$. Since this step could update the values of N_l and B_l, the step will be repeated until an iteration does not result in the creation of any new bins.

- Having removed the values that obstructed the quantile strategy, it is now possible to apply it directly. The special bins identified earlier may have led to the fragmentation of the remaining intervals (it could happen that the domain identified by the values not yet assigned also includes these special ranges, but we must ensure that the final result does not have overlapping bins). In this regard, denoting I as the number of ranges of this nature, the quantile strategy will be applied to it by inputting the integer part of $\frac{B_l}{I}$ as the number of bins to calculate (if the operation returns a remainder, it will only be added to the first interval).

An example of the result of this algorithm is shown in Figure 5.3, compared to the one obtained using a uniform division, which is clearly ineffective for the purposes of this module.



(a) **(b)**

Figure 5.3: Histograms that show how the values of *count_sent* are distributed among the 6 bins detected using the uniform (a) and the custom (b) strategies, showing how the latter is the only one employable.

.

5.5.2 Frequent Itemset Extraction

Having mapped each continuous feature to categorical features, it is now possible to apply frequent itemset extraction algorithms such as FP-Max to each cluster identified in the previous step. Traditional extraction algorithms like Apriori and FP-growth are not applicable due to the unique characteristic of the data. Assuming the need to extract common features in a cluster where almost all entities have never sent transactions, the number of frequent itemsets that will be identified would be at least 32!, given that the 32 features describing the active behavior of the user (in the current window, excluding those related to the past) will all be set to default values. Such a high number of features would saturate memory and, from a practical standpoint, is unnecessary. Essentially, it could be described by the frequent itemset containing all 32 default categories. Since it's not possible to convey this kind of functional dependency to the algorithm, the only usable algorithm is FP-Max. FP-Max ensures returning only maximal itemsets that satisfy the specified minimum support, avoiding the degenerate case illustrated earlier while simultaneously ensuring a non-trivial solution.

After the descriptors are extracted, the following information is written to a text file:

- For each feature, the calculated intervals in the binning phase will be specified.
- For each cluster, starting from the set of extracted itemsets, the set of unique individual itemsets present is written. The list of all frequent itemsets, along with their respective support, will be written for all clusters in a separate file, accessible as needed. This choice is made for practical reasons, as there are cases where the list contains a high number of elements.
- For each cluster, the population is reported, and for each feature, the mean, standard deviation, and the number of values different from default values are provided.
- For each cluster, finally, the identifiers of the 5 most representative entities are reported: those that are least distant from the centroid.

5.6 Aggregated results analysis

At this point, a fairly comprehensive description of the characteristics of each cluster is available. However, it is not possible to qualitatively validate the final result. In this regard, at this stage, the intervention of a domain expert is necessary. The expert, consulting the text file previously produced, will select a limited number of clusters and label them by observing their description. These labels will then

be potentially assigned to suspicious users (as well as their banks and countries of residence), enriching the available information that will be essential for the investigation to be conducted, which will determine the innocence or guilt of the marked subjects.

Chapter 6

Experimental Results

6.1 Implementation Details

The experiments described in this chapter were conducted using the BigData @Polito Cluster. Specifically, this infrastructure is equipped with more than 1700 CPU cores, 19 TB of RAM memory, and 8 PB of storage available for users. A single machine that a user can reserve is equipped with 35 to 60 CPU threads and 120 to 260 GB of memory. The experiments that will follow focus on describing the results obtained by analyzing users, banks, and countries, also proposing a possible outcome of the Aggregate Results Analysis step, which will be carried out in production by a domain expert. An experiment involving the second pipeline will also be conducted, showing the agreement between the originally proposed models for outlier detection and the new one proposed in this thesis. For each pair of models, the agreement will be calculated as the percentage of users marked as suspicious by both models compared to the total number of users marked by the models. For the purposes of the experiments, Month 3 2022 will be considered as the time window.

6.2 Evaluating the Collection Enrichment Pipeline

By considering the same time window, specified previously, an analysis of the third pipeline on all the three possible aggregation level will be conducted.

6.2.1 Users

The number of users extracted are about 1.5 million and the extraction features process at this level took about 70 minutes. Since the majority of users reside in Europe, as well as their banks, the one-hot encoded vectors expressing these pieces

of information have been dropped, as including them would worsen the results of clustering.

Feature selection

In order to determine a suitable value for r_{min}, the number of retained features was evaluated as a function of its variation, as shown in Figure 6.1. It can be observed that for values close to 0.80, the number of retained features is almost half. For this reason, the result obtained for $r_{min} = 0.80$ will be selected for further analysis.

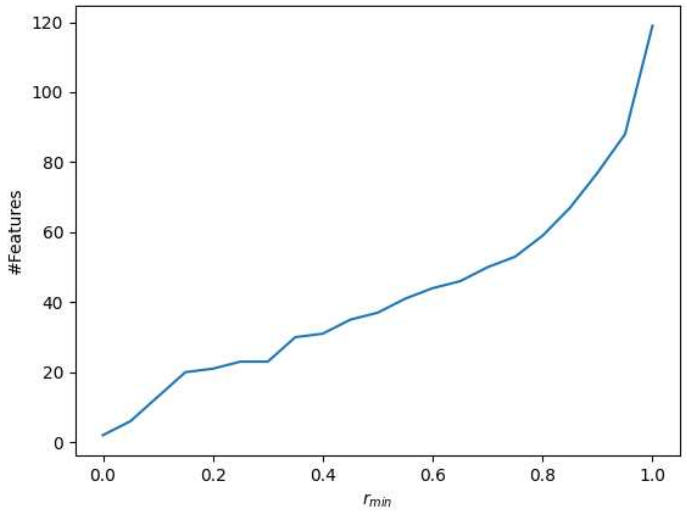


Figure 6.1: Plot that shows how many features are retained for different values of the r_{min} parameter, on users.

As shown in Figure 6.2, through the single feature *funneling_bnk*, it is possible to represent a very high number of features. The feature selection algorithm narrowed the number of user features from 99 down to 60, where 22 are one-hot encoded bits.

Using PCA (applied exclusively to continuous features), it can be observed through the plot in Figure 6.3 that only the first 16 components are able to express more than 80 percent of the total variance, while the last 17 components appear to be less significant. Therefore, 16 components will be selected, and when added to the 22 bits set aside, they will constitute the components on which the clustering algorithm will be executed.

```
Retained: funneling_bnk
Dropped: txns_frequency_sent, txns_frequency_received, sum_sent, mean_sent,
count_sent, sum_received, mean_received, count_received, count_bank_country_sent,
count_user_country_sent, count_bank_country_received, count_user_country_received,
max_amount_sent, min_amount_sent, max_amount_received, min_amount_received,
count_bic8_sent, count_bic8_received, sum_unexpectedness_o_b_sent,
sum_unexpectedness_ob_bb_sent, sum_unexpectedness_o_b_received,
sum_unexpectedness_ob_bb_received, hrg_users_low_sent, hrg_banks_low_sent,
hrg_users_low_received, hrg_banks_low_received, funneling_user
```

Figure 6.2: The list of features that have been dropped by the feature selection algorithm, represented by *funneling_bnk*, on users.

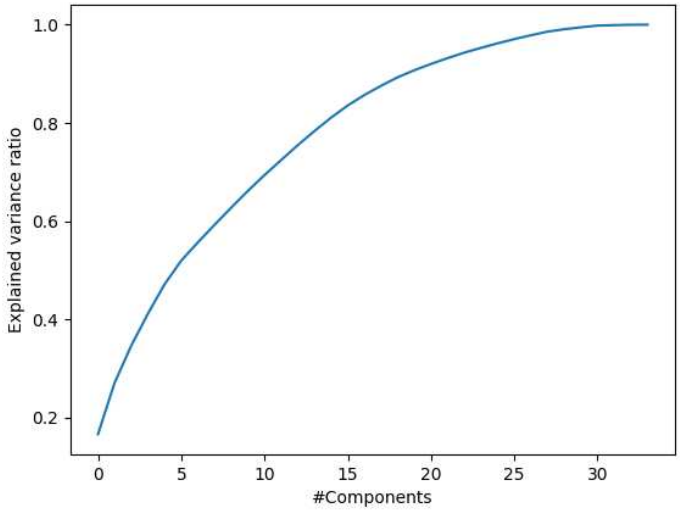


Figure 6.3: Plot that shows the explained variance ratio against the number of components on the PCA computed, on users. The first 15 components retain about the 80% of the variance.

Aggregation/Clustering module

For this step, the K-Means algorithm is employed, specifying $k_{min} = 2$ and $k_{max} = 40$ as range boundaries for the search of the optimal k in terms of Silhouette score. As shown in Figure 6.4, high values are obtained for $k = 2$ and for $15 \leq k \leq 30$, while the result obtained for $k = 5$ is the worst. The result obtained for $k = 2$ is easily justified by the fact that users can naturally be divided into those who do not receive or never execute transactions, a hypothesis confirmed by observing the extracted descriptors. It can be hypothesized instead that the poor quality of

clustering for $k = 5$ is due to points belonging to a small cluster that are very close to points assigned to another larger cluster, lowering the metric employed. The highest Silhouette score is obtained for $k = 21$, in which the smallest cluster has approximately 5000 points, while the largest has 200,000.

This step takes roughly, 8 hours.

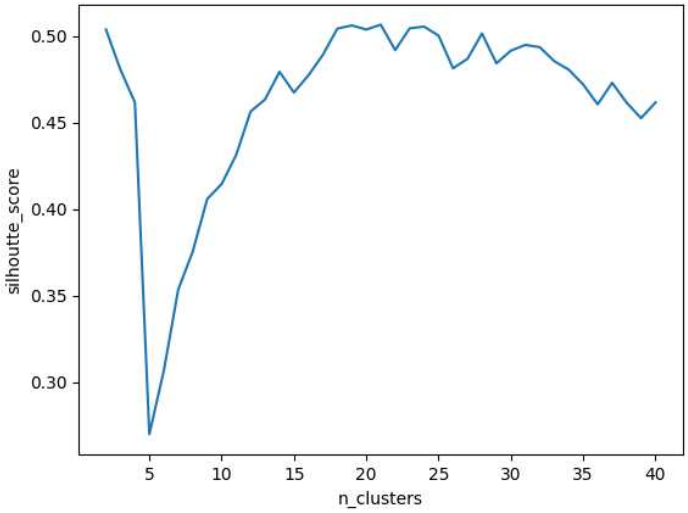


Figure 6.4: Plot that shows the estimation of the Silhoutte scores for values of k that goes from 2 to 40, on users. The best value is obtained for $k = 21$.

Descriptors Extractor

For this step, continuous features have been binned using the custom strategy specifically implemented, calculating 6 intervals for each of them and choosing a *minimum support* equal to 0.9. Observing the extracted descriptors for each cluster, it is possible to notice that the formation of each cluster is concentrated on the following characteristics:

- Users have never received or sent transactions.
- The country of the users or the banks to which they have sent or from which they have received money.
- The HRG level of the countries of the users and the banks to which they have sent or received money.
- The country of the owning bank.

- Whether they have sent or received transactions in the past.

Examples of extracted descriptors are shown in the Tables 6.1, 6.2, 6.3 and 6.4. The extraction took roughly 10 minutes. The generic descriptors *feature*_$[f, t)$, indicates the bin where the continuous variable *feature* has value in the range that goes from f to t, where $f, t \in \mathbf{R}$ and $f \leq t$.

Cluster 0 - Descriptors
bank_RO_sent_[1.0,1.0)
count_received_[0.0,0.0)
funneling_bnk_[0.301,0.301.0)
hrg_banks_low_received_[1.0,1.0)
hrg_users_low_received_[1.0, 1.0)
motif_n1n_[0.0, 0.0)
past_mean_received_[61.0, 61.0)
user_RO_sent_[1.0, 1.0)

Table 6.1: Descriptors extracted for Cluster 0, at user level using a *minimum support* of 0.9. The number of points is 155270. A possible level could be "*Users that did not sent any transactions and receiving them from medium HRG countries*".

Cluster 6 - Descriptors
bank_IE_sent_[1.0,1.0)
bank_country=IE
count_received_[0.0,0.0)
count_sent_[1.0,1.0)
funneling_bnk_[0.301,0.301.0)
hrg_banks_low_received_[1.0,1.0)
hrg_users_low_received_[1.0, 1.0)
motif_n1n_[0.0, 0.0)
past_mean_received_[61.0, 61.0)
user_IE_sent_[1.0, 1.0)

Table 6.2: Descriptors extracted for Cluster 6, at user level using a *minimum support* of 0.9. The number of points is 102311. A possible level could be "*Users, with irish banks, that did not received any transactions and sent them to irish users and banks*".

Cluster 8 - Descriptors
bank_<other>_received_[1.0,1.0)
count_sent_[0.0,0.0)
funneling_bnk_[-0.301,-0.301.0)
hrg_banks_medium_received_[1.0,1.0)
hrg_users_medium_received_[1.0, 1.0)
motif_n1n_[0.0, 0.0) past_active_sent_[61.0, 61.0)
user_<other>_received_[1.0, 1.0)

Table 6.3: Descriptors extracted for Cluster 8, at user level using a *minimum support* of 0.9. The number of points is 20131. A possible label could be "*Users that did not sent any transactions and receiving them from medium HRG countries*".

Cluster 10 - Descriptors
bank_<other>_received_[1.0,1.0)
count_received_[1.0,1.0)
count_sent_[0.0,0.0)
funneling_bnk_[-0.301,-0.301.0)
hrg_banks_high_received_[1.0,1.0)
hrg_users_high_received_[1.0, 1.0)
motif_n1n_[0.0, 0.0) past_active_sent_[61.0, 61.0)
user_<other>_received_[1.0, 1.0)

Table 6.4: Descriptors extracted for Cluster 10, at user level using a *minimum support* of 0.9. The number of points is 24871. A possible label could be "*Users who have never sent transactions and received from infrequent countries located in high level HRG*".

Aggregated Result Analysis

A possible labeling of the identified clusters could be based on the key characteristics that guide the clustering algorithm in their formation, as illustrated earlier. Taking Cluster 10, as shown in Figure 6.4, and examining descriptors such as *count_sent_[0.0,0.0)*, *count_received_[1.0,1.0)*, *bank_<other>_received_[1.0, 1.0)*, and *hrg_banks_high_received_[1.0, 1.0)*, a potential label could be "Users who have never sent transactions and received from infrequent countries of banks located in high HRG".

6.2.2 Banks

The banks extracted are 3,392, and the feature extraction process at this level took about 1 minute and 20 seconds, having the users' features precomputed. In this case, the one-hot encoded vectors specifying the country of the users were dropped, and some features with zero variance were ignored, such as *motif_uturn*.

Feature selection

Again, to determine a good value for r_{min}, the number of retained components for different values was evaluated. Observing 6.5 (the figure reference seems to be missing), $r_{min} = 0.80$ seems like a reasonable value to choose. The number of features in this case reduces from 121 to 64, of which 22 are one-hot encoded bits.

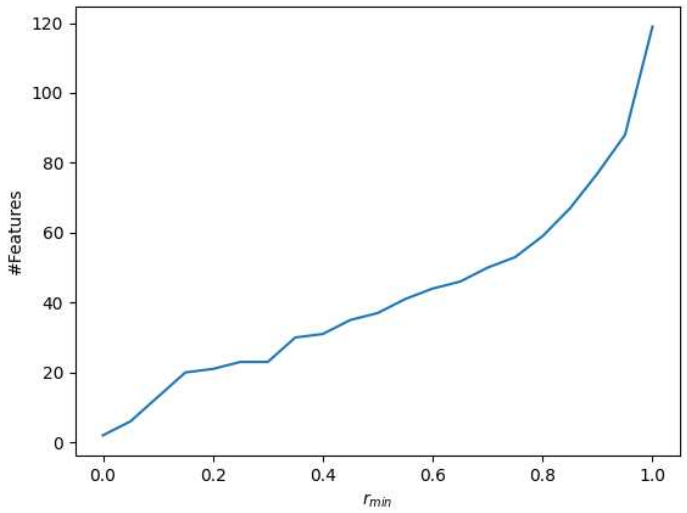


Figure 6.5: Plot that shows how many features are retained for different values of the r_{min} parameter, on banks. For values around 0.80 the number of features are practically halved.

.

Employing PCA, it can be observed through the plot in Figure 6.6 that only the first 15 components are able to express more than 80 percent of the total variance. Therefore, this 15 components with the other retained one hot encoded bits, are the preprocessed features on which the clustering algorithm will work.

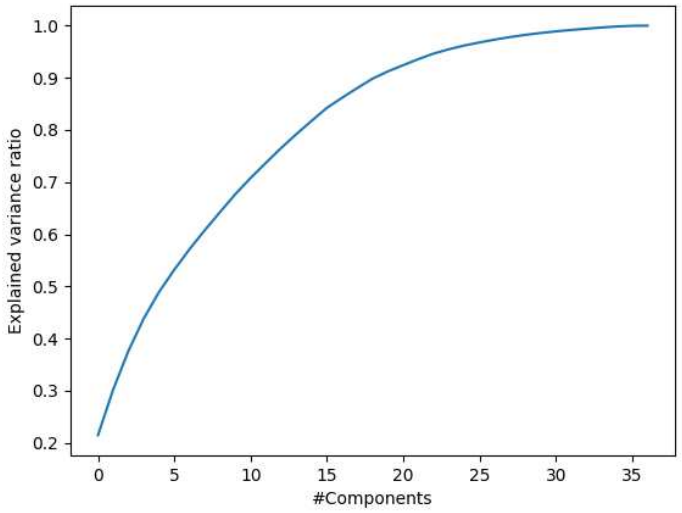


Figure 6.6: Plot that shows the explained variance ratio against the number of components on the PCA computed, on banks. The first 15 components retain about the 80% of the variance.

Aggregation/Clustering module

For this step, $k_{min} = 2$ and $k_{max} = 20$, are the range boundaries for the search of the optimal k in terms of Silhouette score. A restricted range has been selected since the number of banks is much lower than the users one. As showed in figure 6.7, the highest values are for $k > 10$. The best promising Silhouette score is obtained for $k = 15$, where the clusters have a minimum size of around 50 points up to the most populous one consisting of 900 points. This step took 10 seconds.

Descriptors Extractor

In this step, the same number of intervals chosen for the case of users was selected and a *minimum support* equal to 0.8 was used. Observing the descriptors output for each cluster, the banks were also grouped according to the same pattern followed for users. The extraction took approximately 3 minutes. Examples of extracted descriptors are shown in Tables 6.5 and 6.6.

Aggregated Result Analysis

Taking Cluster 13 as example (Table 6.6) and observing the descriptors *count_sent_[0.0,0.0)*, *count_received_[1.236,10.0)*, *bank_IE_received_[1.0, 1.0)*, and *past_count_[1.236,*

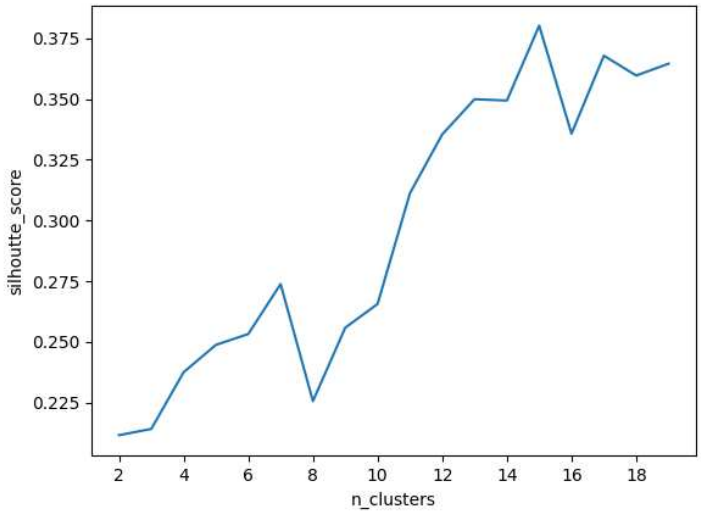


Figure 6.7: Plot that shows the estimation of the Silhoutte scores for values of k that goes from 2 to 20, on banks. The best value is obtained for $k = 15$.

Cluster 12 - Descriptors
count_received_[0.0,0.0)
count_sent_[2.0, 649)
funneling_bnk_[0.301,0.301.0)
motif_uturn_[0.0, 0.0)

Table 6.5: Descriptors extracted for Cluster 12, at bank level using a *minimum support* of 0.8. The number of points is 127. A possible label could be "*Banks, whose users, sent a very high number of transactions and never received some*".

12.0), a potential label could be "Banks, whose users, have never sent transactions and have received multiple transactions from Irish banks, also receiving multiple transactions in the past".

6.2.3 Countries

The extracted countries are 235, and the feature extraction process at this level took about 30 seconds, reusing precomputed users features. In this case, unlike the experiments performed on users, the one-hot encoded vectors specifying the region of origin of users and the bank are not ignored.

Cluster 13 - Descriptors
bank_IE_received_[1.0,1.0)
count_received_[1.236,10.0)
count_sent_[0.0,0.0)
funneling_bnk_[-0.301,-0.301.0)
hrg_banks_low_received_[1.0,1.0)
motif_n1n_[0.0, 0.0)
motif_uturn_[0.0, 0.0) past_count_received_[2.0, 12.0)
past_mean_sent_[0.0, 0.0)
user_region=Europe

Table 6.6: Descriptors extracted for Cluster 13, at bank level using a *minimum support* of 0.8. The number of points is 44. A possible label could be *"Banks, whose users, have never sent transactions and have received multiple transactions from Irish banks, also receiving multiple transactions in the past"*.

Feature selection

Even in this case, the behavior of the number of features dropped against r_min has been studied. Observing 6.8, $r_{min} = 0.80$, confirms to be a reasonable value to choose. The number of features in this case reduces from 121 to 64, of which 22 are one-hot encoded bits.

The Principal Component Analysis (Figure 6.9) shows that the 13 components keep more than 80 percent of the total variance, and for this reason only these axis will be retained for the next step.

Aggregation/Clustering module

Since the number of different countries is very small, the search of the optimal clustering is between $k_{min} = 2$ and $k_{max} = 8$. The highest Silhouette score is achieved for $k = 6$ (Figure 6.10), where the clusters have a minimum size of about 50 points up to the most populous one consisting of 900 points. The search, took less than a second.

Descriptors Extractor

Again, the number of intervals chosen is 6 and even for this aggregation level, the entities are grouped in the same way did for banks and users . The extraction, took few milliseconds using *minimum support* equal to 0.8. Examples of extracted descriptors are shown in Tables 6.7 and 6.8.

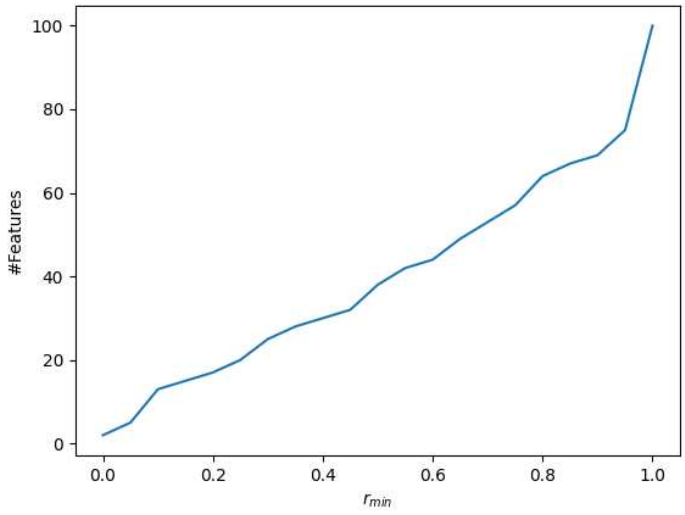


Figure 6.8: Plot that shows how many features are retained for different values of the r_{min} parameter, on countries.

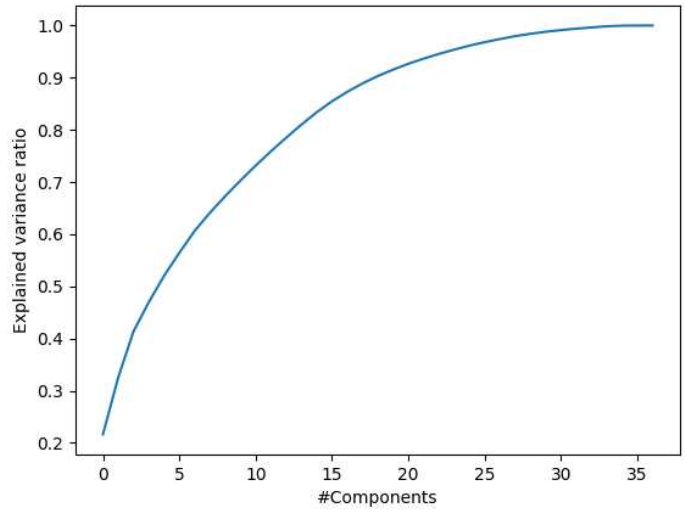


Figure 6.9: Plot that shows the explained variance ratio against the number of components on the PCA computed, on countries. The first 13 components retain about the 80% of the variance.

.

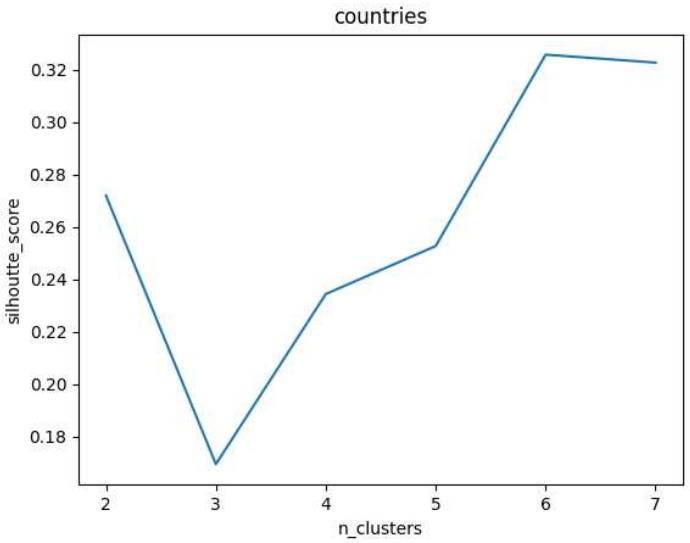


Figure 6.10: Plot that shows the estimation of the Silhoutte scores for values of k that goes from 2 to 8, on countries. The best value is obtained for k = 7. 61

Cluster 0 - Descriptors
bank_IT_sent_[1.0, 1.0)
bank_country=<other>
count_received_[0.0,0.0)
count_sent_[1.0,1.0)
funneling_bnk_[0.301, 0.301.0)
hrg_banks_low_received_[1.0,1.0)
motif_n1n_[0.0, 0.0)
past_mean_sent_[0.0, 0.0)

Table 6.7: Descriptors extracted for Cluster 0, at country level using a *minimum support* of 0.8. The number of points is 30. A possible label could be "*Countries, whose users, did not received any transactions and sent them to italian banks. The banks of this users are in not frequent countries*".

Aggregated Result Analysis

Taking Cluster 0 as an example (as shown in Table 6.7), by observing the descriptors *count_sent_[1.0, 1.0)*, *count_received_[0.0, 10.0)*, *bank_IT_sent=[1.0, 1,0)* and *bank_country=<other>*, a potential label could be "*Countries, whose users, did not received any transactions and sent them to italian banks. The banks of this*

Cluster 3 - Descriptors
bank_region=Europe
count_received_[0.0,0.0)
count_sent_[1.0,1.0)
funneling_bnk_[0.301, 0.301.0)
hrg_banks_low_received_[1.0,1.0)
hrg_users_low_received_[1.0,1.0)
motif_n1n_[0.0, 0.0)
user_RO_sent_[0.0, 0.0)

Table 6.8: Descriptors extracted for Cluster 3, at country level using a *minimum support* of 0.8. The number of points is 18. A possible label could be "*Countries, whose users, with european banks, did not received any transactions and sent them to romanian users*".

users are in not frequent countries".

6.3 Outlier Detection Models Agreement

The currently available transaction dataset is unlabeled; hence, all proposed outlier detection models operate in an unsupervised manner. The absence of labels not only diminishes the overall reliability of the architecture but also prevents the evaluation of the two outlier detection modules in terms of crucial metrics such as precision and recall. The objective of this experiment is to assess the agreement among the various models implemented so far and those proposed in this thesis, aiming to determine which pairs of models identify the same users as suspicious and which ones rely on different patterns during their operation. The metric we will employ is the Intersection Over Union, formally calculated as follows:

- Let $D = u_1, u_2, ..., u_N$ be the set of users under examination, where N is the cardinality of this set.
- Let M and R be two instances of models for which we want to calculate the metric, i.e., Autoencoder and One-Class Support Vector Machine.
- Let c (*contamination*) be the hyperparameter indicating the percentage of users within D assumed to be anomalous.
- For both models M and R, calculate for each user u_i their anomaly score, denoted as $s_{i,M}$ and $s_{i,R}$, respectively.

- Determine, for both models, the sets T_M and T_R containing the $\frac{c}{N}$ users with the highest anomaly scores for models M and R respectively.
- Next, calculate the sets $U = T_M \bigcup T_R$ and $I = T_M \bigcap T_R$. The Intersection Over Union metric between models M and R is computed as IOU $= \frac{\#I}{\#U}$.

A value of IOU equal to 1 indicates that the models have precisely labeled the same set of users as anomalous, while a value of 0 indicates that the sets are completely disjoint. The configuration and the hyperparameters chosen for this experiment are:

- The Autoencoder model employed is comprised of two linear layers for the encoder part and two linear layers for the encoding one. The hyperparameter λ, which controls the tradeoff between the Mean Squared Error and the Binary Cross Entropy is set to 1.
- The number of trees of the Isolation Forest is set to 100.
- The One-Class SVM, use a radial basis kernel with a value of $\nu = 0.1$, employing an ensemble approach with 10 instances.
- The number of neighbors to use by default for k-neighbors queries, in the Local Outlier Factor model is set to 200, employing an ensemble approach with 10 instances.
- For the K-Means based model, the one proposed in this thesis, the range in which the best clustering results should be found is specified by setting $k_{min} = 2$ and $k_{max} = 40$.

The set of features on which the models will work, are the one selected with the feature selection algorithm employed for clustering, setting $r_{min} = 0.80$. The PCA will be employed for all the run, except for the Autoencoder one (since it is a deep learning model). The results of the experiment is show in Table 6.9, choosing $c = 0.0001$.

The results show how the new proposed model exhibits, except for the Local Outlier Factor, a high level of agreement with all the models. The behavior of the One Class Support Vector Machine also seems to be aligned with the same models, while the predictions between the Autoencoder and the Isolation Forest do not overlap.

To try to understand what leads a model to classify a user as anomalous, it is possible to use the descriptors extraction module on the set of suspicious users, treating them as if they were clusters. The extracted descriptors could then help explain what the most significant features are in the outlier detection task, as they represent the description of the structure that unites the marked bank accounts. This analysis has shown that:

IOU	AE	IF	LOF	OCSVM	KM
AE	1.0	0.08	0.0	0.16	0.18
IF	0.08	1.0	0.0	0.42	0.39
LOF	0.0	0.0	1.0	0.02	0.0
OCSVM	0.16	0.42	0.02	1.0	0.57
KM	0.18	0.39	0.0	0.57	1.0

Table 6.9: The table show the Intersection Over Unit values, for *contamination* set to 0.0001. The K-Means based model shows an high degree of agreement with all the models, except the Local Outlier Factor one.

- For the K-Means model, anomalous users have sent and received discrete sums of money in the past. In the current window, users seem to be involved in a high number of transactions involving the exchange of money that replicate the motif $N - 1 - M$.
- Consistent with the results obtained, the One Class Support Vector Machine presents the same set of descriptors.
- The model based on an Autoencoder presents descriptors very close to those of the previous analyzed models. Moreover, the set of users extracted for this model has in common the fact that they have sent transactions to Italian banks.
- The Isolation Forest marks as anomalous mainly users who have sent large amounts in the past and are involved in the motif $N - 1 - M$.
- On the other hand, the Local Outlier Factor identifies as anomalous users who have received from banks and Irish users, not presenting any transactions as originators.

Chapter 7

Conclusions

In this thesis work, a pipeline for extending a machine learning-based architecture for anomaly detection in transaction pass-throughs has been described. The goal was to find an effective way to cluster entities in the financial network at different levels and make these results interpretable, enriching the information available about suspicious users found in other pipelines.

To achieve this objective, an initial feature engineering process was designed, where raw transactions were used to describe users, banks, and countries through a set of features recommended by domain experts. Since the quality of the features directly influenced the outcomes of the last two pipelines, a careful analysis of their distributions was conducted, showing that most users sent or received only very few transactions. The particular nature of these bank accounts made many features redundant and highly correlated with each other, leading to the implementation of a feature selection algorithm that extracts only the most representative ones. Subsequently, data mining algorithms like FP-Max were employed to make the results of the clustering step interpretable, a procedure facilitated by a previous binning process implemented from scratch, necessary due to the power-law shape of the features.

The final results at all aggregation levels have been presented, showing that entities are grouped mostly based on whether they have only sent or received transactions and/or on the country to or from which they sent or received them. Furthermore, the second pipeline has been extended by reusing the K-Means-based algorithm employed in the third one as an outlier detection model, which shows a higher level of agreement with the already implemented Autoencoder.

A continuation of this thesis work could certainly focus on improving the feature engineering process through active cooperation with a domain expert. Indeed, some features may prove useless or even harmful to the aggregation and outlier detection process, and some that play a fundamental role may have been dropped during feature selection because they were believed to be redundant for the employed

feature selection algorithm. Moreover, an extension of entity descriptions could be considered, for example, by introducing, for instance, features that include the amount of money sent or received to or from a set of countries/regions.

Additionally, assuming that labeled dataset will be available with metadata about the entities involved, it would be possible not only to employ supervised models whose performance can be evaluated by well-defined metrics but also to refine the clustering process by introducing external evaluation metrics, making the final result closer to the desired outcome.

Finally, in a hypothetical future scenario where this architecture will be used for online detection of anomalous transactions, incremental clustering techniques can be employed to provide as much information as possible in a very short time to assess the degree of anomaly of a transaction. Although this approach significantly increases the complexity of the entire architecture, it would greatly improve response times in detecting financial crimes.

Bibliography

[1] In: (). URL: https://developers.google.com/machine-learning/clustering/clustering-algorithms?hl=en (cit. on p. 6).

[2] Stuart P. Lloyd. «Least squares quantization in PCM». In: *IEEE Trans. Inf. Theory* 28 (1982), pp. 129–136. URL: https://api.semanticscholar.org/CorpusID:10833328 (cit. on p. 6).

[3] David Arthur and Sergei Vassilvitskii. «K-Means++: The Advantages of Careful Seeding». In: *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms*. SODA '07. New Orleans, Louisiana: Society for Industrial and Applied Mathematics, 2007, pp. 1027–1035. ISBN: 9780898716245 (cit. on p. 7).

[4] Pang-Ning Tan, Michael Steinbach, and Vipin Kumar. *Introduction to Data Mining*. US ed. Addison Wesley, May 2005. ISBN: 0321321367. URL: http://www.amazon.com/exec/obidos/redirect?tag=citeulike07-20%5C&path=ASIN/0321321367 (cit. on pp. 8, 10, 13, 15).

[5] Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. «A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise». In: *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*. KDD'96. Portland, Oregon: AAAI Press, 1996, pp. 226–231 (cit. on p. 8).

[6] Anil K. Jain and Richard C. Dubes. *Algorithms for Clustering Data*. USA: Prentice-Hall, Inc., 1988. ISBN: 013022278X (cit. on p. 10).

[7] Peter J. Rousseeuw. «Silhouettes: A graphical aid to the interpretation and validation of cluster analysis». In: *Journal of Computational and Applied Mathematics* 20 (1987), pp. 53–65. ISSN: 0377-0427. DOI: https://doi.org/10.1016/0377-0427(87)90125-7. URL: https://www.sciencedirect.com/science/article/pii/0377042787901257 (cit. on p. 11).

[8] Richard Bellman. «Dynamic programming». In: *Science* 153.3731 (1966), pp. 34–37 (cit. on p. 12).

[9] David Freedman, Robert Pisani, and Roger Purves. «Statistics (international student edition)». In: *Pisani, R. Purves, 4th edn. WW Norton & Company, New York* (2007) (cit. on p. 14).

[10] Karl Pearson F.R.S. «LIII. On lines and planes of closest fit to systems of points in space». In: *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science* 2.11 (1901), pp. 559–572. DOI: 10.1080/14786440109462720 (cit. on p. 14).

[11] Rakesh Agrawal, Tomasz Imielinski, and Arun Swami. «Mining Associations Between Sets of Items in Massive Databases». In: *Proceedings of the ACM-SIGMOD 1993 International Conference on Management of Data, 1993* (Jan. 1993) (cit. on p. 17).

[12] Rakesh Agrawal and Ramakrishnan Srikant. «Fast Algorithms for Mining Association Rules». In: *Proc. 20th Int. Conf. Very Large Data Bases VLDB* 1215 (Aug. 2000) (cit. on p. 17).

[13] Jiawei Han, Jian Pei, and Yiwen Yin. «Mining Frequent Patterns without Candidate Generation». In: *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data*. SIGMOD '00. Dallas, Texas, USA: Association for Computing Machinery, 2000, pp. 1–12. ISBN: 1581132174. DOI: 10.1145/342009.335372. URL: https://doi.org/10.1145/342009.335372 (cit. on p. 18).

[14] Gösta Grahne and Jianfei Zhu. «High Performance Mining of Maximal Frequent Itemsets Gösta». In: 2003. URL: https://api.semanticscholar.org/CorpusID:13264349 (cit. on p. 18).

[15] Yan Yang, Bin Lian, Lian Li, Chen Chen, and Pu Li. «DBSCAN Clustering Algorithm Applied to Identify Suspicious Financial Transactions». In: *2014 International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery*. 2014, pp. 60–65. DOI: 10.1109/CyberC.2014.89 (cit. on p. 18).

[16] Franklim Arévalo, Paolo Barucca, Isela-Elizabeth Téllez-León, William Rodríguez, Gerardo Gage, and Raúl Morales. «Identifying clusters of anomalous payments in the salvadorian payment system». In: *Latin American Journal of Central Banking* 3.1 (2022), p. 100050. ISSN: 2666-1438. DOI: https://doi.org/10.1016/j.latcb.2022.100050. URL: https://www.sciencedirect.com/science/article/pii/S2666143822000059 (cit. on p. 18).

[17] Asma S. Larik and Sajjad Haider. «Clustering based anomalous transaction reporting». In: *Procedia Computer Science* 3 (2011). World Conference on Information Technology, pp. 606–610. ISSN: 1877-0509. DOI: https://doi.org/10.1016/j.procs.2010.12.101. URL: https://www.sciencedirect.com/science/article/pii/S187705091000476X (cit. on p. 19).

[18] Mohammad Ali Farajian and Shahriar Mohammadi. «Mining the Banking Customer Behavior Using Clustering and Association Rules Methods». In: *Int. J. Indust. Eng. Prod. Res* 21 (Dec. 2010), pp. 239–245 (cit. on p. 19).

[19] Michele Starnini et al. «Smurf-Based Anti-money Laundering in Time-Evolving Transaction Networks». In: *Machine Learning and Knowledge Discovery in Databases. Applied Data Science Track*. Ed. by Yuxiao Dong, Nicolas Kourtellis, Barbara Hammer, and Jose A. Lozano. Cham: Springer International Publishing, 2021, pp. 171–186. ISBN: 978-3-030-86514-6 (cit. on pp. 37, 38).

[20] Flavio Giobergia, Elena Baralis, Maria Camuglia, Tania Cerquitelli, Marco Mellia, Alessandra Neri, Davide Tricarico, and Alessia Tuninetti. «Mining Sensor Data for Predictive Maintenance in the Automotive Industry». In: Oct. 2018, pp. 351–360. DOI: 10.1109/DSAA.2018.00046 (cit. on p. 48).