

**CHARLES UNIVERSITY**  
**FACULTY OF SOCIAL SCIENCES**

Institute of Economic Studies



**Comparative Analysis of Outlier Detection  
Models for Transaction Monitoring**

Master's thesis

Author: Ing. Petra Kohoutová

Study program: Economics and Finance

Supervisor: prof. PhDr. Ladislav Křištofuk, Ph.D.

Year of defense: 2024

## **Declaration of Authorship**

The author hereby declares that he or she compiled this thesis independently, using only the listed resources and literature, and the thesis has not been used to obtain any other academic title.

The author grants to Charles University permission to reproduce and to distribute copies of this thesis in whole or in part and agrees with the thesis being used for study and scientific purposes.

Prague, July 25, 2024

Ing. Petra Kohoutová

## Abstract

Outlier detection is a critical task in various domains, such as finance and cybersecurity, as it helps identify anomalies that can provide valuable insights for data cleansing and decision-making. The increasing availability of large and complex datasets has led to a growing demand for effective outlier detection models. While numerous approaches exist, there is a need for comprehensive research that compares and evaluates these models to understand their performance and suitability for different datasets and outlier scenarios. This thesis aims to conduct a comparative analysis of outlier detection models and apply them to data used in transaction monitoring, to gain insights into their strengths, weaknesses, and real-world applicability in this field. The models examined include Isolation Forest, cluster-based analysis, and copulas, each suitable for different sets of use cases. Given the challenges of evaluating transaction monitoring data due to missing or unreliable data labels, this comparative analysis seeks to provide a clear understanding of how these models perform under such conditions and how can they be evaluated based on the expert-based knowledge.

**JEL Classification** C39, C52, G21, L59, O16, O33

**Keywords** outliers, anomaly, model, data

**Title** Comparative Analysis of Outlier Detection Models for Transaction Monitoring

## Abstrakt

Detekce odlehlých pozorování je klíčovou součástí různých oblastí, jako jsou finance a kybernetická bezpečnost, protože pomáhá identifikovat anomálie, které mohou poskytnout cenné poznatky pro čištění dat a rozhodování. Zvyšující se dostupnost velkých a komplexních datových souborů vedla ke zvyšující se poptávce po efektivních modelech detekce odlehlých pozorování. I když existuje mnoho přístupů, je potřeba komplexního výzkumu, který porovná a vyhodnotí tyto modely, aby porozuměl jejich výkonu a vhodnosti pro různé soubory dat a specifické scénáře. Tato práce si klade za cíl provést komparativní analýzu modelů detekce odlehlých hodnot, aby bylo možné získat náhled na jejich silné a slabé stránky a jejich použitelnost v reálném světě. Izolační les, klastrová analýza a kopule jsou modely, které jsou všechny vhodné pro různé případy

použití. Další metody jsou zmiňovány především proto, že transakční data používaná pro trénování těchto modelů jsou často bez označení, případně jejich značení není spolehlivé a hodnocení výkonnosti těchto modelů je často dělané na expertní znalosti jedince.

**Klasifikace JEL** C39, C52, G21, L59, O16, O33

**Klíčová slova** odhledlá pozorování, anomálie, model, data

**Název práce** Komparativní analýza modelů detekce odlehlých pozorování pro účely monitoringu transakcí

## Acknowledgments

The author is especially grateful to prof. PhDr. Ladislav Krištoftek, Ph.D. for his guidance, comments, and ideas that helped improve this thesis's contents. Additionally, the author expresses deep appreciation to their colleagues for their essential knowledge and support, which were instrumental in the completion of this work. Special thanks are also due to Dr. Mehyaar Najla and Dr. Andrea Gagna for their exceptional guidance and numerous crucial ideas that greatly enriched this thesis.

Typeset in L<sup>A</sup>T<sub>E</sub>X using the IES Thesis Template.

### **Bibliographic Record**

Kohoutová, Petra: *Comparative Analysis of Outlier Detection Models for Transaction Monitoring*. Master's thesis. Charles University, Faculty of Social Sciences, Institute of Economic Studies, Prague. 2024, pages 89. Advisor: prof. PhDr. Ladislav Krištoftek, Ph.D.

# Contents

List of Tables	viii
List of Figures	ix
Acronyms	xi
<b>1 Introduction</b>	<b>1</b>
<b>2 Literature Review</b>	<b>4</b>
2.1 Supervised Learning . . . . .	5
2.2 Unsupervised Learning . . . . .	6
2.3 Outlier Detection . . . . .	7
<b>3 Methodology</b>	<b>9</b>
3.1 Transaction Monitoring . . . . .	9
3.2 Customer Segmentation . . . . .	12
3.3 Scenarios . . . . .	13
3.4 Isolation Forest . . . . .	16
3.4.1 Masking and Swamping Phenomena . . . . .	17
3.4.2 Anomaly Score . . . . .	20
3.5 Clustering . . . . .	21
3.5.1 K-Means . . . . .	22
3.5.2 Euclidian Distance . . . . .	23
3.6 Copulas . . . . .	24
3.6.1 Theoretical Framework of COPOD . . . . .	26
<b>4 Data</b>	<b>29</b>
4.1 Artificial Data . . . . .	29
4.2 Real-world Data . . . . .	32
4.3 Data Preparation . . . . .	38

---

<b>5</b>	<b>Model Implementation</b>	<b>43</b>
5.1	Isolation Forest . . . . .	45
5.2	K-Means . . . . .	52
5.3	Copulas . . . . .	60
<b>6</b>	<b>Limitations</b>	<b>66</b>
<b>7</b>	<b>Model Comparison</b>	<b>69</b>
<b>8</b>	<b>Conclusion</b>	<b>73</b>
	<b>Bibliography</b>	<b>78</b>

# List of Tables

5.1	Performance metrics of Isolation Forest on real-world data . . .	50
5.2	Outliers detected by Isolation Forest after applying the thresholds to scenarios . . . . .	51
5.3	Performance metrics of K-Means on real-world data . . . . .	59
5.4	Outliers detected by K-Means after applying the thresholds to scenarios . . . . .	60
5.5	Performance metrics of COPOD on real-world data . . . . .	65
5.6	Outliers detected by COPOD after applying the thresholds to scenarios . . . . .	65
7.1	Quantitative performance metrics of models across all scenarios	71
7.2	Qualitative performance metrics of models across all scenarios .	72



# List of Figures

3.1	Transaction monitoring process . . . . .	10
3.2	Example of scenario . . . . .	11
3.3	Example of TM system coverage with proper segmentation model	13
3.4	Isolation Forest separation example . . . . .	17
3.5	Isolation Forest . . . . .	18
3.6	Original dataset . . . . .	19
3.7	Example of subsampling . . . . .	19
3.8	K-Means steps . . . . .	23
4.1	Conceptual schema of the artificial data . . . . .	31
4.2	Default distribution of artificial transactions . . . . .	31
4.3	Conceptual scheme of the real-world data . . . . .	34
4.4	Methodology of the parametrization process . . . . .	39
4.5	Transaction count by segment and risk category . . . . .	39
5.1	Confusion Matrix of Isolation Forest - Scenario 1 . . . . .	47
5.2	ROC Curve of Isolation Forest - Scenario 1 . . . . .	47
5.3	Confusion Matrix of Isolation Forest - Scenario 2 . . . . .	48
5.4	ROC Curve of Isolation Forest - Scenario 2 . . . . .	48
5.5	Confusion Matrix of Isolation Forest - Scenario 3 . . . . .	49
5.6	ROC Curve of Isolation Forest - Scenario 3 . . . . .	50
5.7	Feature Distribution - Scenario 1 [zoomed in] . . . . .	51
5.8	Inertia - Scenario 1 . . . . .	53
5.9	Confusion Matrix of K-Means - Scenario 1 . . . . .	54
5.10	ROC Curve of K-Means - Scenario 1 . . . . .	55
5.11	Inertia - Scenario 2 . . . . .	55
5.12	Confusion Matrix of K-Means - Scenario 2 . . . . .	56
5.13	ROC Curve of K-Means - Scenario 2 . . . . .	56
5.14	Inertia - Scenario 3 . . . . .	57

---

5.15	Confusion Matrix of K-Means - Scenario 3 . . . . .	57
5.16	ROC Curve of K-Means - Scenario 3 . . . . .	58
5.17	Inertia - Scenario 1 . . . . .	58
5.18	Inertia - Scenario 3 . . . . .	59
5.19	Inertia - Scenario 2 . . . . .	59
5.20	Confusion Matrix of COPOD - Scenario 1 . . . . .	61
5.21	ROC Curve of COPOD - Scenario 1 . . . . .	62
5.22	Confusion Matrix of COPOD - Scenario 2 . . . . .	63
5.23	ROC Curve of COPOD - Scenario 2 . . . . .	63
5.24	Confusion Matrix of COPOD - Scenario 3 . . . . .	64
5.25	ROC Curve of COPOD - Scenario 3 . . . . .	64

# Acronyms

<b>AI</b>	Artificial Intelligence
<b>AUC</b>	Area Under the Curve
<b>AML</b>	Anti-Money Laundering
<b>BST</b>	Binary Search Tree
<b>CTF</b>	Counter Financing of Terrorism
<b>COPOD</b>	Copula-based Outlier Detection
<b>FATF</b>	Financial Action Task Force
<b>FP</b>	False Positive
<b>IF</b>	Isolation Forest
<b>ML</b>	Machine Learning
<b>SAR</b>	Suspicious Activity Report
<b>TM</b>	Transaction Monitoring
<b>WA</b>	Worthy Alert

# Chapter 1

## Introduction

In the contemporary information era, Machine Learning (ML) assumes a pivotal role in various industries, driving revolutionary trends. Positioned as a subset of artificial intelligence (AI), machine learning operates within the realm of computational science, dedicated to analyzing and interpreting patterns and structures in data. Its primary objective is to autonomously facilitate learning, reasoning, and decision-making, transcending human interaction. This process endows systems with the capability to learn and improve from experience without explicit programming, underscoring the transformative potential of machine learning (Jhaveri *et al.* 2022). Essentially, users can input vast datasets into computer algorithms, enabling the system to autonomously analyze data and provide data-driven recommendations and decisions based solely on input data. Experiential learning is manifested through data collected within specific application domains, typically comprising values on various features relevant to the scenarios.

Fundamental to machine learning is the assumption that collected data encapsulates multiple potential patterns through which the characteristics of interesting behaviors can be modeled. Machine learning aims to devise scientific, compelling, and robust approaches to unveil these underlying models. Machine learning algorithms broadly fall into two categories: generative and discriminative models. Despite their categorization, both types of models operate on the premise that the collected data represents a set of samples from unknown distributions. Generative models seek to learn the distribution of generating the data by estimating the parameters of the assumed model (Goodfellow *et al.* 2016). On the other hand, discriminative models aim to optimize the observed data with fewer assumptions on the underlying distributions. As the volume of

training examples increases, discriminative models tend to outperform generative models with superior performance. Consequently, discriminative models are favored, mainly when dealing with large datasets, to ensure performance and robustness (Alpaydin 2020).

Discriminative models, further classified into regression and classification, play a crucial role in machine learning (Bishop 2006). Regression models explore how the target value changes concerning variations in independent feature variables while keeping other independent variables fixed. These models find widespread application in prediction and forecasting. In classification, samples in the dataset are assumed to belong to different classes distributed across various regions of the feature space, with a hyperplane existing between each class's areas.

Outlier detection, or one-class classification, becomes particularly relevant when data primarily comes from one category, with minimal or no representation from other classes (Chandola *et al.* 2009). This scenario is common in many real-world applications, such as fraud detection, network security, and medical diagnosis, where the normal cases vastly outnumber the anomalies. Identifying outliers is crucial because they often represent significant, albeit rare, events that can impact the system's performance or security. For instance, in fraud detection, outliers may indicate fraudulent transactions that deviate from normal spending patterns. In network security, unusual patterns of data can signal potential intrusions or attacks. Similarly, in medical diagnosis, detecting anomalies in patient data can lead to the early identification of rare diseases or conditions. By focusing on the majority class and identifying deviations from it, one-class classification models can effectively isolate these outliers, providing critical insights and enhancing the robustness of predictive models (Murphy 2012a).

This thesis undertakes the construction and comparison of three machine learning models, assessing their capability to identify outliers. These models range from simplistic to more complex approaches, and their performance is assessed using multiple metrics tailored to specific use cases and labeled data.

The models will undergo testing on two types of transactional data used in transaction monitoring scenarios. Transaction monitoring presents a unique challenge for outlier detection, which is integral to optimizing system performance and therefore crucial for identifying potential money laundering activity.

The thesis is structured as follows. Chapter 2 provides a theoretical foundation for machine learning models, encompassing both supervised and unsu-

---

pervised learning methodologies. Chapter 3 focuses on the methodology used in this study, detailing the specific use cases where these models are applied. It also examines outlier detection techniques employed in the analysis. Chapter 4 elaborates on the datasets utilized, including the creation of synthetic data and the procurement of real-world transactional data. It discusses the processes involved in data preparation and addresses the challenges encountered during this phase.

Chapter 5 centers on the training of machine learning models and evaluates their predictive capabilities using various performance metrics. Chapter 6 discusses the limitations inherent in transaction monitoring and examines the complexities involved in accurately assessing model performance on transactional data. Chapter 7 compares the performance of each machine learning model, offering insights into their respective strengths and weaknesses across different use cases. Finally, Chapter 8 synthesizes the findings from the preceding chapters, presenting a comprehensive summary of the study's outcomes and implications.

# Chapter 2

## Literature Review

The following literature review section focuses on academic papers predominantly centered on Outlier Detection techniques, recognizing the inefficiency of simple methods in handling vast datasets. Consequently, the review emphasizes Machine Learning methods for Outlier Detection due to their aptitude for managing large and complex datasets.

There exist various definitions of Machine Learning. Mitchell (1997) provides a technical definition: "A computer program is said to learn from experience  $E$  with respect to some class of tasks  $T$  and performance measure  $P$  if its performance at tasks  $T$ , as measured by  $P$ , improves with experience  $E$ ." To elucidate the use case of Outlier Detection, task  $T$  involves detecting anomalies in a given dataset. Task  $E$  comprises the training data supplied to the algorithm, controlled by  $P$ , to ensure the algorithm operates with high performance.

Numerous algorithms are tailored to address function approximation problems, wherein the objective is encapsulated within a function. For instance, in the context of fraud detection, the task may involve assigning a "fraud" or "not fraud" label to a given input transaction. The learning challenge revolves around enhancing the accuracy of this function, with the learning experience derived from a sample of known input-output pairs of the function. In certain instances, the function is explicitly represented in a parametrized functional form, while in other cases, it is implicit and acquired through processes such as search procedures, factorization, optimization techniques, or simulation-based procedures. Even in cases of implicit representation, the function typically relies on parameters or other adjustable degrees of freedom, and training involves determining values for these parameters that optimize the performance metric (Jordan & Mitchell 2015).

The two primary approaches in machine learning are called supervised and unsupervised learning. Each offers diverse applications across various contexts and datasets. The subsequent chapter will present a detailed exploration of the advantages and disadvantages of supervised and unsupervised learning methods.

## 2.1 Supervised Learning

Supervised learning is a machine learning technique wherein the algorithm is trained on a labeled dataset. In this paradigm, the algorithm learns to establish a mapping between input features and targets based on the labeled training data. The essence of supervised learning lies in the provision of input features along with corresponding output labels to the algorithm, allowing it to generalize from this information and make predictions on new, unseen data (Jhaveri *et al.* 2022).

Models developed through Supervised Learning demonstrate exceptional efficacy in addressing classification and regression tasks, particularly in datasets that are appropriately labeled. Common applications include spam identification, facial recognition in images, and weather forecast price predictions. These applications exemplify the function approximation problem discussed earlier, where training data takes the form of  $(x, y)$  pairs, and the objective is to generate predictions  $y^*$  in response to a query  $x^*$ . As previously mentioned, supervised learning is prominently employed in two main tasks: classification and regression (Murphy 2012b).

Classification represents a supervised learning method wherein the algorithm acquires the ability to assign input data to specific categories or classes based on input features. The output labels in classification are discrete values, and these algorithms can either be binary, with the output belonging to one of two classes, or multiclass, allowing for multiple class possibilities. In the context of our test case, we are exclusively dealing with two classes, making it a binary classification problem.

On the other hand, regression, another form of supervised learning, involves algorithms learning to predict continuous values based on input features. The output labels in regression are continuous values, such as stock prices or housing prices. Despite the multitude of regression models available, this thesis concentrates solely on the classification problem, as the data at hand pertains to a binary classification scenario (Jordan & Mitchell 2015).



The advantages of supervised learning are in the ability to manually delineate class boundaries, providing specificity in defining desired classes. Moreover, supervised learning tends to be computationally less demanding, translating into reduced processing time. However, it may exhibit limitations in exceptionally complex tasks compared to unsupervised learning.

In the realm of transaction monitoring, a critical challenge arises with supervised machine learning methods. Given the limited approaches in current transaction monitoring practices and their inefficiency, there is a scarcity of labeled data available. Moreover, these methods become obsolete when implementing entirely new monitoring scenarios due to the inability to utilize existing labels. Consequently, supervised machine learning methods prove viable for established scenarios with reliable labels, while unsupervised machine learning methods become essential for cases lacking such labeled data, as expounded in the subsequent chapter.

## 2.2 Unsupervised Learning

Unsupervised learning stands as a potent pattern recognition approach, employing artificial intelligence algorithms to scrutinize datasets devoid of predefined categories or labels for data points. These algorithms autonomously uncover latent relationships within datasets, governed by assumptions about the underlying structural properties, often articulated through algebraic or probabilistic frameworks (Jordan & Mitchell 2015). This methodology proves especially valuable for exploratory data analysis, cross-selling strategies, customer segmentation, and image recognition, showcasing a remarkable capability to discern patterns, similarities, and differences in information, including image recognition.

A noteworthy adaptation involves transforming several supervised machine learning methods into unsupervised counterparts. This is achieved by introducing the concept of artificial class labels, distinguishing "observed" data from synthetically generated data, where the observed data represents the original unlabeled data, and synthetic data are drawn from a reference distribution (Shi & Horvath 2006).

Unsupervised learning, distinct from supervised learning due to the absence of labeled data, signifies a paradigm where there is no predetermined relationship between datasets, and outcomes cannot be anticipated. Characterized by minimal human involvement, unsupervised learning relies on dataset observa-

tions to unveil intrinsic data qualities. Clustering, a fundamental concept in unsupervised learning, facilitates the identification of diverse groupings within a dataset. Two subcategories of unsupervised learning challenges include clustering and dimensionality reduction (Pham & Ruz 2009).

Clustering algorithms strategically group similar data points based on their characteristics, with the goal of identifying distinct clusters within the dataset. In contrast, dimensionality reduction algorithms streamline datasets by reducing the number of input variables while preserving as much original information as possible, thereby aiding in dataset simplification for visualization and analysis (Fujimaki *et al.* 2005).

While unsupervised learning boasts advantages, such as not requiring labeled data and fostering the discovery of novel patterns and relationships, it is not without limitations. The absence of a predefined output may lead to algorithms producing potentially meaningless or inaccurate results. Additionally, evaluating the performance of unsupervised learning algorithms is a complex task due to the lack of a predefined correct output for comparison (Kohoutová 2023). The computational demands and time complexity of these algorithms increase, especially with a growing number of features, contributing to heightened model complexity. Despite being less prevalent in finance compared to supervised learning, unsupervised algorithms find a niche in fraud detection within transaction monitoring, where labeled data is often scarce.

## 2.3 Outlier Detection

Outlier detection comprises a diverse array of techniques, often sharing fundamental similarities but adopting different terminologies, such as outlier detection, novelty detection, anomaly detection, exception mining, noise detection, or deviation detection, as observed in the literature. For the scope of this thesis, the term chosen to encapsulate these techniques is "outlier detection."

In tandem with the varied nomenclature, authors have proposed numerous definitions of an outlier, and no single definition has gained universal acceptance. Aggarwal & Yu (2001) highlights the divergence, noting that outliers might be perceived as noise points lying outside a defined cluster set, or alternatively, they could be points outside the set of clusters yet distinct from the noise. Another perspective, put forth by Hawkins (1980), characterizes an outlier as an observation deviating significantly from others, prompting suspicion that it originated from a distinct mechanism. For the purposes of this thesis,

outliers are defined as data points situated at an abnormal distance from others within a dataset. In the context of transaction monitoring, such outliers can indicate potentially suspicious or fraudulent activities (Kohoutová 2023).

In applications like mobile phone or transaction monitoring, detecting sudden changes in usage patterns is crucial for identifying potential fraud, such as stolen phones or money laundering. Outlier detection analyzes time series usage statistics to achieve this (Ngai *et al.* 2011; Chandola *et al.* 2009). In areas like loan processing or social security benefit payments, an outlier detection system can identify anomalies in applications before approval or payment, ensuring data consistency and integrity (Iglewicz & Hoaglin 1993). Equity or commodity traders use outlier detection to monitor markets for novel trends, indicating buying or selling opportunities (Ahmed *et al.* 2016). In news delivery systems, outlier detection ensures timely updates on changing stories. Outliers in databases may indicate errors or fraudulent cases, requiring different handling based on the application area (Hodge & Austin 2004). For natural anomalies like extreme population features, a robust classification algorithm accommodating outliers is essential. In safety-critical or fraud detection environments, real-time detection and immediate alarms are crucial (Xia *et al.* 2020). Once addressed, anomalous readings may be stored separately for comparison with new cases but are often not included in the main system data, which typically models normality to detect anomalies (Chandola *et al.* 2009).

# Chapter 3

## Methodology

This chapter outlines the methodology and models employed in this study. We begin by introducing and explaining transaction monitoring, detailing its purpose and functionality to provide a clear understanding of why specific models were chosen and how outlier detection is integral to transaction monitoring systems. This foundational knowledge is crucial for grasping the overall logic behind the use of outlier detection.

Following this introduction, we delve into the specific methodologies of each model utilized in this thesis. Each model's methodology is explained in detail, highlighting how it fits into the transaction monitoring framework and its relevance to identifying suspicious activities. By providing a comprehensive overview of both the transaction monitoring process and the models used, this chapter aims to offer a thorough understanding of the research approach and the rationale behind it.

### 3.1 Transaction Monitoring

Implementing Transaction Monitoring systems in compliance with Anti-Money Laundering (AML) policies deters criminals from infiltrating the financial system with illegal funds. Organizations avoid involvement in money laundering due to potential consequences such as significant fines for non-compliance, reputational damage linked to crimes like modern slavery, human trafficking, or drug distribution, and personal liability that could lead to imprisonment.

An AML solution includes a system or set of tools specifically designed to prevent and detect activities related to money laundering, terrorist financing, and other illicit financial transactions. These solutions typically integrate

advanced technologies, established processes, and comprehensive regulatory frameworks to identify suspicious financial activities, monitor transactions, and report anomalies to the relevant authorities. The systematic approach employed by these solutions is illustrated in Figure 3.1. Financial institutions such as banks, credit unions, and brokerage firms, as well as entities involved in financial transactions like casinos and money service businesses, primarily use these solutions.

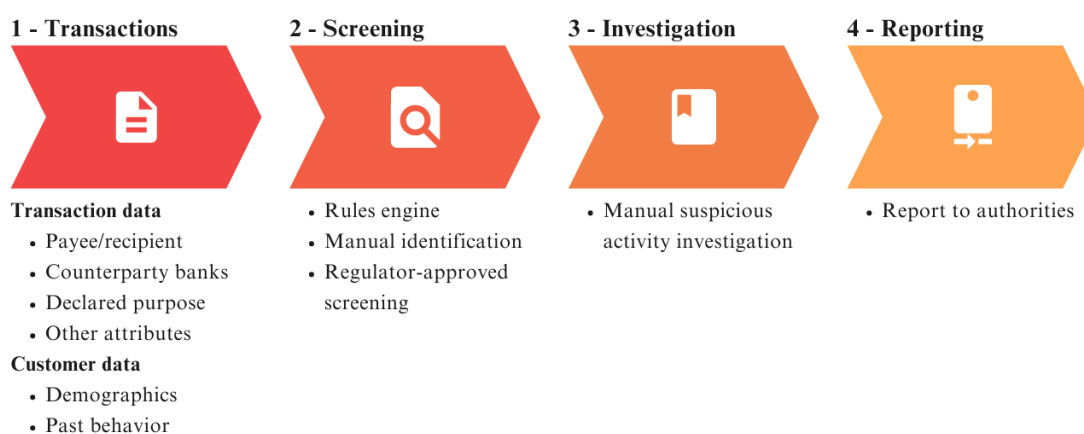


Figure 3.1: Transaction monitoring process

Source: Own elaboration

Implementing an AML solution helps organizations mitigate the risks associated with money laundering, protect their reputation, fulfill regulatory obligations, and contribute to global efforts to combat financial crimes.

AML solutions operate based on a set of rules known as scenarios, which include logical operations such as conjunction, disjunction, and negation. These scenarios are designed to monitor specific behaviors that have been previously identified as criminal activities in terms of AML/CTF.

The overall monitoring process may vary among banks, but it generally follows these basic principles:

1. **Transaction Monitoring** - Transactions are monitored using predefined scenarios.
2. **Suspicious Activity Detection** - Scenarios are employed to detect suspicious activities.
3. **Alert Generation** - The system determines whether to generate an alert. Most systems generate an alert immediately upon detecting suspicious activity, while others require confirmation from multiple scenarios.

4. **Activity Investigation** - Suspicious activities are manually investigated and escalated as necessary.
5. **Case Resolution** - After gathering sufficient information about the entities involved in the transaction, the investigation team makes a final determination and closes the case as either a false positive or a Suspicious Activity Report (SAR). If a SAR is warranted, a form must be completed and submitted, along with any supporting evidence, to the regulatory authority (Monson & Vandermark 2013).

A model situation is shown in Figure 3.2. In this example, money is being laundered within a criminal group, with each member depositing a certain amount into their accounts, which is later transferred to a joint account. A scenario to monitor this particular behavior would be designed to track an unusual number of low-value transactions occurring within a short period (e.g., a week or a day). If the quantity and value of these transactions surpass a predefined threshold, the system would generate an alert for further manual investigation. Otherwise, the behavior would be evaluated as expected.

### MONEY-LAUNDERING SCHEME

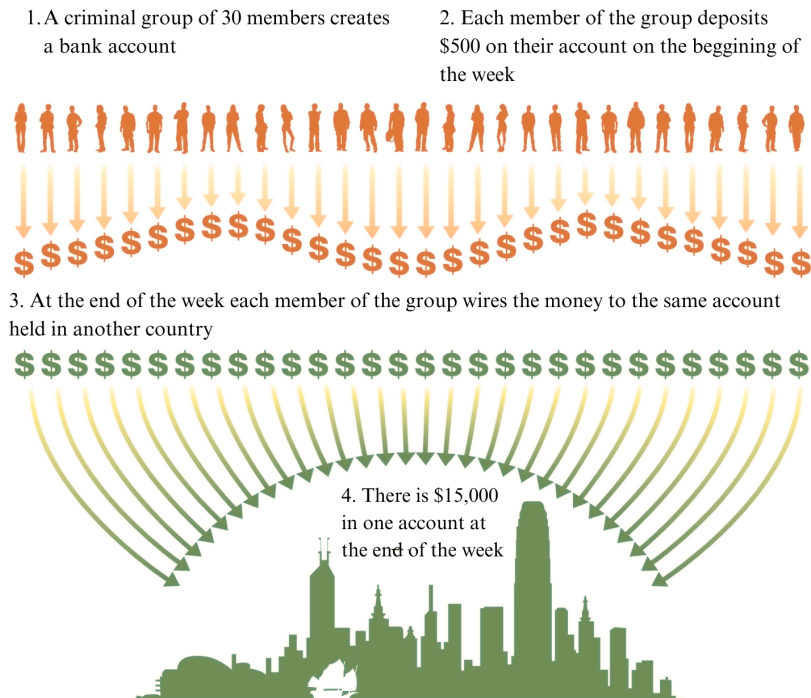


Figure 3.2: Example of scenario

Source: Own elaboration

This is where outlier detection becomes crucial, as correctly determining thresholds is essential for the proper functionality and effectiveness of transaction monitoring scenarios. Thresholds are typically set through tuning analysis, which uses different outlier detection models to find the "right" cut-off point between regular, unproblematic transactions and potentially suspicious ones. It is important to note that tuning must occur regularly, at least once a year, as criminals continually develop new patterns for money laundering, and advances in AI further complicate detection efforts. Additionally, customer behavior changes over time, necessitating regular tuning assessments.

## 3.2 Customer Segmentation

Choosing the correct model is important for another reason. As previously described, transaction monitoring scenarios function based on set thresholds. However, it is not advisable to use a single threshold for the entire customer portfolio, as customers vary in the amount, size, and type of transactions they make. For example, a large enterprise conducts very different transactions compared to a regular student, yet both groups need proper monitoring. Setting thresholds too high to accommodate the substantial financial activities of big corporations may result in insufficient monitoring of students, allowing potentially problematic transactions to go undetected. Conversely, excessively stringent thresholds based on student behavior could lead to an overwhelming number of false positives for transactions involving big corporations. An example illustrating the impact of proper segmentation on monitoring is shown in Figure 3.3. Refined customer segmentation ensures that transactional thresholds align with the risk profiles of each customer group, striking an optimal balance between detection accuracy and false positives.

Customer segmentation involves grouping a customer portfolio into clusters based on similarities in user profiles. By employing appropriate and comprehensive segmentation techniques, financial institutions can significantly enhance their ability to monitor all customers diligently, thereby reducing the likelihood of overlooking malicious transactions. In 2021, the Financial Action Task Force (FATF) recommended behavioral customer segmentation as the best option in transaction monitoring models. Behavioral segmentation combines business understanding with various machine learning algorithms to identify customers' real behavioral patterns and risks, thus improving the efficiency and effectiveness of risk-pattern identification. Through analysis, clusters of customers with

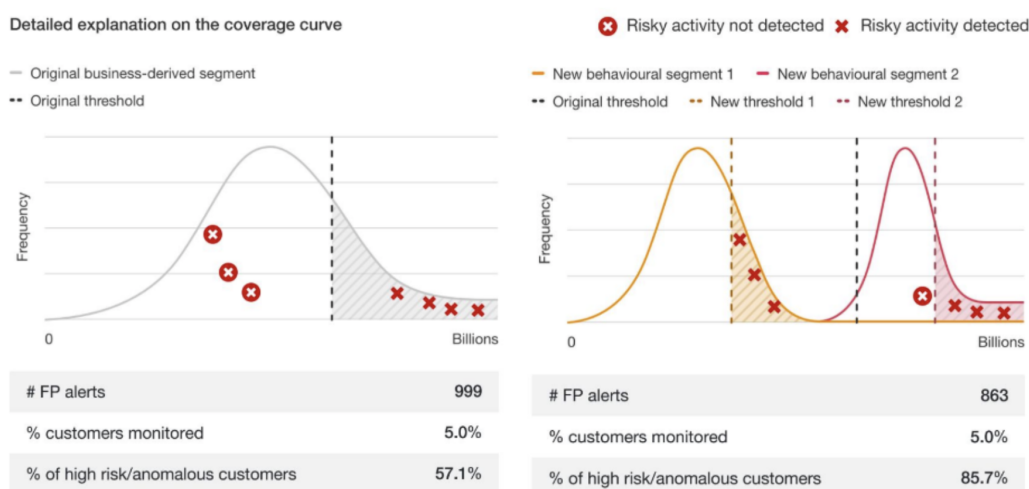


Figure 3.3: Example of TM system coverage with proper segmentation model

Source: Own elaboration

similar behaviors can be defined and applied as a new segmentation model in the transaction monitoring system. This also includes providing explanations and descriptions of the new segments so that AML analysts fully understand the type of customers included in each segment. However, many financial institutions still predominantly employ simple clustering models based primarily on client type, which may not yield optimal efficiency in practice.

### 3.3 Scenarios

AML scenarios are critical components of transaction monitoring systems in financial institutions, designed to detect and prevent suspicious activities related to money laundering and other financial crimes. To ensure their effectiveness, these scenarios require regular updates and revisions. They are developed based on real-world AML/CTF cases investigated by authorities such as FATF and the Egmont Group. These organizations regularly publish reports that highlight new criminal behavior patterns and trends, which inform the creation and adjustment of AML scenarios.

For this analysis, three specific scenarios have been created. These scenarios are designed to cover both low-dimensional and higher-dimensional cases, to test the introduced models under different conditions. The assumption is that simpler models will perform better in low-dimensional cases, while more complex models will excel in higher-dimensional cases.



The first scenario, which will be referred to as Scenario 1, is a simple one. In the Transaction Monitoring (TM) context, this scenario might be considered too simple and, on its own, insufficient to fully represent real criminal behavior. Therefore, it would likely be used in combination with other scenarios. However, we are introducing this scenario to conduct a proper comparative analysis of different outlier detection models in low-to-high dimensions, starting with one dimension. This scenario focuses on *"Transactional amount exceeding a threshold."* It is straightforward: a hit is generated when the amount of a customer's transactions in the past seven days exceeds a certain threshold. This scenario is executed on a weekly basis. The logic is encoded as follows:

---

```
// Scenario Logic
WireDebitAmount_Customer_PastWeek > Threshold_1
```

---

Here, *"WireDebitAmount\_Customer\_PastWeek"* represents the amount of a transaction made by the customer.

The second scenario, which we will refer to as Scenario 2, focuses on *"High daily amount of incoming wire transfers compared to 3-month turnover."* This scenario consists of 2 quantitative parameters to be calibrated. In this scenario, a hit is generated at the end of the day if the following conditions are met:

- The amount of incoming wire transfers to an account during the past day exceeds a threshold.
- The amount of incoming wires to an account exceeds 10% of the average credit turnover of the customer in the past three months.

The scenario logic is encoded as follows:

---

```
// Scenario Logic
WireCreditAmount_Customer_PresentDay > 1,000 * Threshold_1
AND
WireCreditAmount_Customer_PresentDay > Threshold_2 * 0.01 *
(WireCreditAmount_Customer_Past1Month +
WireCreditAmount_Customer_Past2Month +
WireCreditAmount_Customer_Past3Month)
/
(0.001 +
(WireCreditAmount_Customer_Past1Month > 0) +
(WireCreditAmount_Customer_Past2Month > 0) +
```

---

```
(WireCreditAmount_Customer_Past3Month > 0))
```

---

In this structure, *WireCreditAmount\_Customer\_PresentDay* represents the total amount of incoming wire transfers credited to the customer on the given day. *WireCreditAmount\_Customer\_PastXMonth* represents the credit turnover on the customer for each of the past three months. The constant 0.001 is added to ensure the formula holds even if the turnover on the account is zero in any of the months of interest.

The third scenario, referred to as Scenario 3, is named "*Volume and frequency of incoming wire transfers increases rapidly.*" In this scenario, a hit is generated if the customer's credit activity increases compared to the previous 6 months. This scenario involves calibration of 4 quantitative parameters and checks 1 quantitative variable to ensure the account age is at least 6 months. The following conditions must be met:

- The sum of credit transaction amounts in the last 30 days exceeds a threshold.
- The number of credit transactions in the last 30 days exceeds a threshold.
- The sum of credit transaction amounts in the previous 6 closed calendar months is below a threshold.
- The number of credit transactions in the previous 6 closed calendar months is below a threshold.
- The customer must have been with the bank for at least 180 days.

The scenario logic is encoded as follows:

---

```
// Scenario Logic
WireCreditAmount_Customer_Past1Month > Threshold_1
AND
WireCreditNumber_Customer_Past1Month > Threshold_2
AND
(WireCreditAmount_Customer_Past1Month +
WireCreditAmount_Customer_Past2Month +
WireCreditAmount_Customer_Past3Month +
WireCreditAmount_Customer_Past4Month +
WireCreditAmount_Customer_Past5Month +
WireCreditAmount_Customer_Past6Month) < Threshold_3
```

---

```
AND
(WireCreditNumber_Customer_Past1Month +
WireCreditNumber_Customer_Past2Month +
WireCreditNumber_Customer_Past3Month +
WireCreditNumber_Customer_Past4Month +
WireCreditNumber_Customer_Past5Month +
WireCreditNumber_Customer_Past6Month) < Threshold_4
AND
NumberDays_Customer_Open >= 180
```

---

In this structure, *WireCreditAmount\_Customer\_PastXMonth* represents the total amount of wire credit transactions credited to the customer in each of the past months of interest, and *WireCreditNumber\_Customer\_PastXMonth* denotes the number of wire credit transactions in each respective month. *NumberDays\_Customer\_Open* indicates the duration of the customer's relationship with the bank in days.

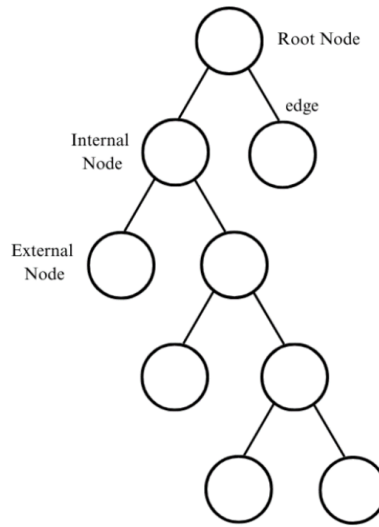
## 3.4 Isolation Forest

One notable algorithm for outlier detection is the Isolation Forest. As an unsupervised machine learning method, it identifies observations as outliers based on their separability from the overall data population. The Isolation Forest employs a binary tree structure, iteratively partitioning the data into two parts until a specific tree height is reached or further splitting is impractical, yielding a structure known as an iTree (Liu *et al.* 2012).

In the iTree, samples closer to the root or at the beginning of the split have a higher potential for being anomalous, while normal samples may require more iterations for isolation. Each node in the iTree can be an external node with no children or an internal node with two children, as shown in Figure 2.1. The path length, which indicates the number of edges from the root node to the external node, characterizes the isolated sample (Chen & Wu 2018).

In the realm of data-induced random trees, instances undergo iterative partitioning until complete isolation is achieved. This stochastic partitioning leads anomalies along shorter paths due to their infrequent occurrence, resulting in fewer partitions and consequently shorter paths within the tree structure (Zhang *et al.* 2015). Moreover, instances with distinctive attribute values are more likely to be segregated early in the partitioning process. Therefore, when

Figure 3.4: Isolation Forest separation example



Source: Own elaboration

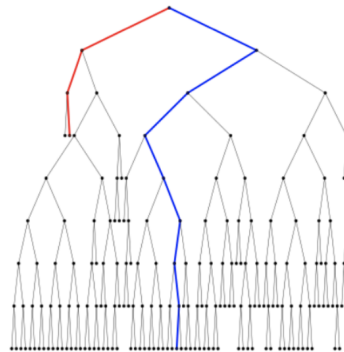
a set of random trees yields shorter path lengths for specific data points, those points are highly likely to be anomalies (Ahmed *et al.* 2016).

With that being said, let  $X = \{X_1, \dots, X_n\}$  be a set of  $d$ -dimensional points and  $X' \subset X$  a subset of these data points. The Isolation Tree (iTree) is structured as a data model where nodes are defined as either external nodes with no children or internal nodes with a "test" consisting of an attribute  $q$  and a split value  $p$ , determining the traversal of a data point to either  $T_l$  or  $T_r$ . The iTree is built by recursively partitioning  $X'$  through random selection of an attribute  $q$  and a split value  $p$ , continuing until each node contains only one instance or all instances at the node share the same attribute values (Liu *et al.* 2008; Zhang *et al.* 2015). Once fully grown, each point in  $X$  is isolated at one of the external nodes. Anomalous points, with shorter path lengths in the tree, are easier to isolate. The path length  $h(x)$  of a point  $x$  is defined as the number of edges  $x$  traverses from the root node to an external node in an iTree. During the evaluation stage, the mean  $h(x)$  in the ensemble of iTree is computed for each value  $x$ , and an average  $h(x)$  is used to calculate the anomaly score.

### 3.4.1 Masking and Swamping Phenomena

Anomaly detection faces challenges in accurately labeling rare events, which can lead to issues such as swamping and masking. Swamping occurs when normal events are incorrectly labeled as anomalies, while masking arises when data

Figure 3.5: Isolation Forest



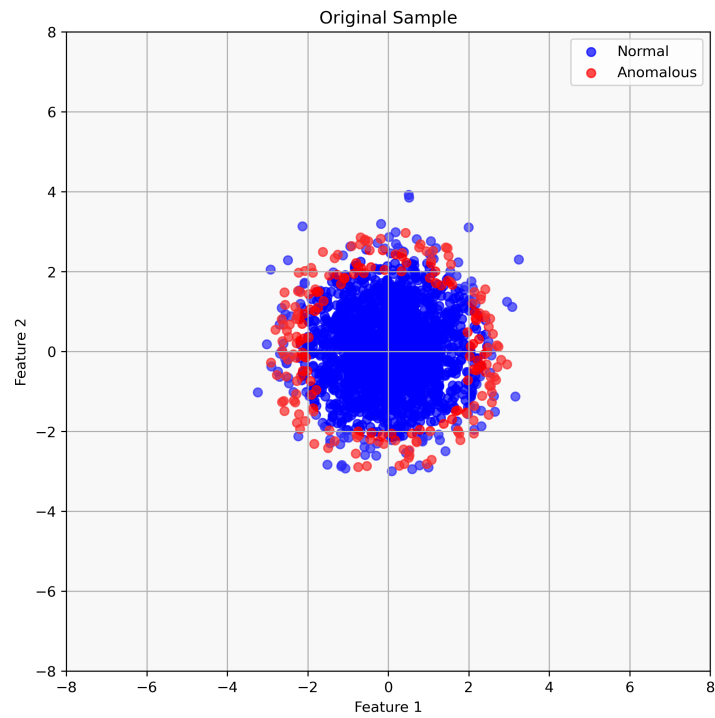
Source: Hariri *et al.* (2018)

points from different clusters merge, making it difficult to detect outliers. These challenges become more pronounced in larger datasets and underscore the need for robust anomaly detection techniques to ensure reliable performance.

To address these obstacles, Isolation Forest (IF) leverages a unique feature — the ability to construct a partial model through sub-sampling. Sub-sampling helps control the dataset size, enabling IF to better isolate instances of anomalies. Moreover, each isolation tree can specialize because each sub-sample may contain a different set of anomalies or none at all. This characteristic of IF is illustrated in Figure 3.6 using an artificial dataset where normal points surround anomaly clusters. In Figure 3.7, a sub-sample of 256 instances is taken from the original data, resulting in clear identification of anomaly clusters and improved anomaly detection.

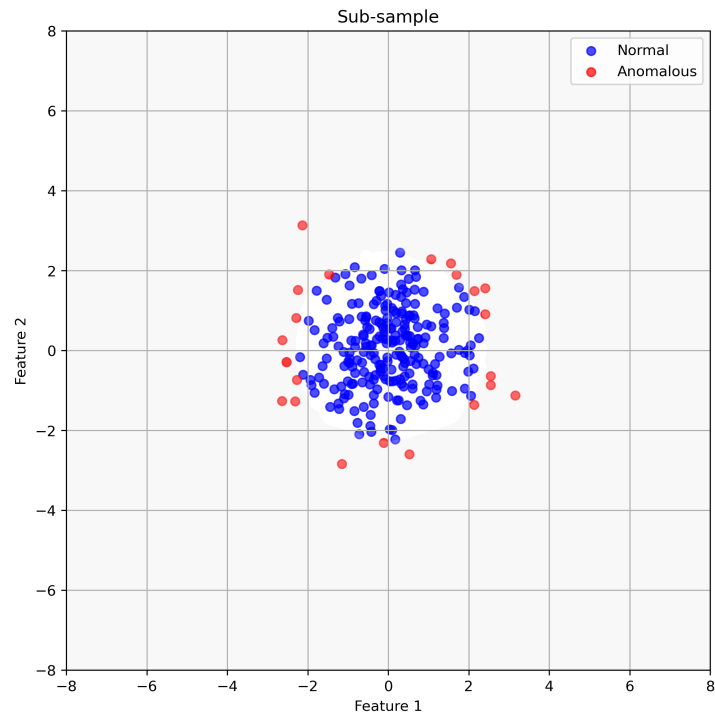
For performance evaluation, the Area Under the Curve (AUC) metric is utilized, which quantifies the ability of a binary classifier to distinguish between classes. The AUC for IF using the entire sample is 0.83, whereas the AUC for a sub-sample of size 128 reaches 0.98. This demonstrates IF's enhanced capability in anomaly detection through reduced sub-sampling.

Figure 3.6: Original dataset



Source: Own elaboration

Figure 3.7: Example of subsampling



Source: Own elaboration

### 3.4.2 Anomaly Score

The anomaly score, also known as the anomaly measure or anomaly probability, is a numerical value that quantifies the degree of abnormality of a data point or observation in comparison to the rest of the dataset. In the context of anomaly detection algorithms, such as Isolation Forest, the anomaly score indicates how likely a data point is to be an anomaly or outlier.

The anomaly score, denoted as  $s(x, i)$ , is calculated as the average  $h(x)$  in IF normalized by the average path of unsuccessful searches in a Binary Search Tree (BST). The formula for  $c(i)$ , which represents the average of  $h(x)$  given  $i$ , is used to normalize  $h(x)$  as follows:

$$c(i) = \begin{cases} 2H(i-1) - \frac{2^{(i-1)}}{n} & \text{for } i > 2 \\ 1 & \text{for } i = 2 \\ 0 & \text{otherwise} \end{cases}$$

where:

- $H$  is the harmonic number, estimated as  $H(.) = \ln(.) + \gamma$  (Euler's Constant),
- $n$  is the testing data size,
- $i$  is the size of the sample set.

The anomaly score allows anomaly detection algorithms to rank data points based on their abnormality level, making it easier to set a threshold for identifying outliers. By specifying a threshold for the anomaly score, analysts can determine which data points are considered anomalies and trigger alerts or further investigation.

Lower anomaly scores indicate a higher likelihood of being normal, while higher anomaly scores suggest a higher probability of being an anomaly.

The formula for the anomaly score,  $s(x, i)$ , uses the average  $h(x)$  from a collection of iTrees and is expressed as:

$$s(x, i) = 2^{-\frac{E(h(x))}{(n)}} \quad (3.1)$$

where:

- $E(h(x))$  is an average value of  $h(x)$  from a collection of iTrees.

From that we can draw a following conclusions:

- scores close to 1 indicates an anomaly,
- scores much smaller than 0.5 indicates normal observations,
- if all scores are close to 0.5 than the entire sample does not seem to have clearly distinct anomalies.

### 3.5 Clustering

Clustering, an unsupervised learning technique, finds application across diverse fields, including engineering, computer science, life sciences, medical sciences, earth sciences, social sciences, and economics (Wang *et al.* 2018). Despite its versatility, clustering introduces confusion due to varying terminologies and objectives. Algorithms tailored to specific domains often make assumptions that favor the particular application of interest, influencing their performance in other scenarios. For instance, the K-Means algorithm, relying on Euclidean measure, tends to generate hyperspherical clusters. If real clusters take different geometric forms, K-Means may become less effective, necessitating alternative approaches (Xu 2005; Jain *et al.* 2020).

In general, clustering methods fall into three groups: partitioning, hierarchical, and density-based clustering (Kaufman & Rousseeuw 2009). This thesis focuses on partitioning clustering, specifically K-Means, as the distance measure aligns well with the purpose throughout. Partitioning aims to maximize similarity within a cluster while minimizing similarity to other clusters (Henig *et al.* 2015). A clustering method's efficacy is determined by well-defined clusters with high intra-class similarity. Instances are organized into subsets, represented as  $C = C_1, \dots, C_k$  of  $S$ , where  $S = \bigcup_{i=1}^k C_i$  and  $C_i \cap C_j = \emptyset$  for  $i \neq j$ . The goal is to cluster the dataset into  $K$  different clusters, with  $K$  as a design parameter. The set  $\mathcal{Z} = \mu_1, \dots, \mu_K$  with  $\mu_1, \dots, \mu_K \in \mathcal{R}^{\mathbb{D}}$  represents the center of each cluster within  $\mathbb{K}$ . The objective is to assign each data point to a cluster center, minimizing the overall distance (Wang *et al.* 2018).

Introducing an indicator variable,  $r_{nk}$ , such that

$$r_{nk} = \begin{cases} 1 & \text{if data point } x_n \text{ is assigned to cluster } k \\ 0 & \text{otherwise,} \end{cases}$$

the clustering problem can be written as



$$\min_{\mu_k, r_{nk}} J = \sum_{n=1}^N \sum_{k=1}^K r_{nk} \|x_n\| \quad (3.2)$$

$$\text{s.t. } r_{nk} \in \{0, 1\} \quad (3.3)$$

$$\sum_{k=1}^K r_{nk} = 1 \forall n. \quad (3.4)$$

### 3.5.1 K-Means

The K-Means algorithm provides a simple and efficient solution for clustering datasets. This iterative optimization method approximates a clustering problem through two steps, alternating between optimizing with respect to  $\mu_k$  and  $r_{n,k}$  while holding the other parameter constant. The process begins with the random initialization of  $\mu_k$ , followed by optimization with respect to  $r_{n,k}$  and  $\mu_k$  held fixed. Since the clustering problem's objective function is linear with respect to  $r_{n,k}$  and independent of  $n$ , optimization for each  $n$  occurs in the following way:

$$r_{nk} = \begin{cases} 1 & \text{if } k = \arg \min_j \|x_n - \mu_j\|^2 \\ 0 & \text{otherwise.} \end{cases}$$

The optimization then proceeds with respect to  $\mu_k$ , keeping  $r_{n,k}$  fixed. The clustering problem's objective function becomes quadratic and convex, allowing the minimum to be found by equating the derivative to zero (Gustavson 2019):

$$\frac{\partial J}{\partial \mu_k} = 2 \sum_{n=1}^N r_{nk} (x_n - \mu_k) = 0. \quad (3.5)$$

This yields the optimal  $\mu_k$  as:

$$\mu_k = \frac{\sum_n r_{nk} x_n}{\sum_n r_{nk}}. \quad (3.6)$$

The optimization continues iteratively until convergence is reached, resulting in

$$r_{nk}, \forall n \in 1, \dots, N, k \in 1, \dots, K. \quad (3.7)$$

and

$$Z = \{\mu_1, \dots, \mu_K\}. \quad (3.8)$$

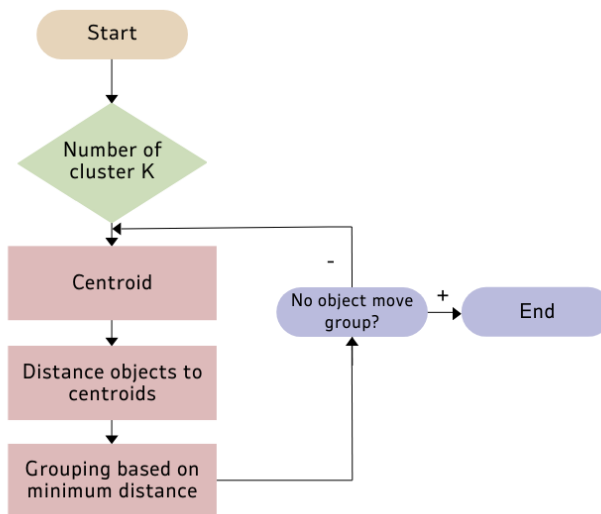
Due to the reduction in the objective function at each step, the method

guarantees convergence. It's crucial to note that K-Means may converge to local minima, and the algorithm doesn't ensure convergence to the global minimum.

### 3.5.2 Euclidian Distance

The fundamental concept behind the K-Means algorithm involves defining  $k$  centroids, each designated for a specific cluster. Placing these centroids strategically is essential, as their location significantly influences the outcome. Ideally, centroids should be positioned as far away from each other as possible. The process proceeds by associating each point in the dataset with its nearest centroid. Once all points are assigned, the initial grouping is completed, and  $k$  new centroids are recalculated. This initiates a loop where each data point is then re-associated with the nearest new centroid. The loop continues until the  $k$  centroids no longer change their positions, signifying the algorithm's convergence (Stojanovic *et al.* 2022). Ultimately, the objective is to minimize the clustering problem (Bora & Gupta 2014).

Figure 3.8: K-Means steps



Source: Own elaboration

In the K-Means algorithm, the distance between each dataset point and every analyzed centroid is calculated. Points are then assigned to the centroid with the minimum distance. This distance calculation is pivotal to the clustering algorithm, and various techniques are available for its computation. The choice of a technique depends on factors such as the data's properties and the dataset dimensions. For this purpose, we employ the Euclidean distance measurement defined as:

$$\sqrt{\sum_{j=1}^k (a_j - b_j)^2} \quad (3.9)$$

Euclidean (and squared Euclidean) distances are typically computed from raw data rather than standardized data. An advantage of this method is that the distance between any two objects remains unaffected by the addition of new objects to the analysis, even if they are outliers. However, differences in scale among dimensions can significantly impact distances. Transforming dimensions to similar scales is often recommended to mitigate this effect.

While effective, the K-Means algorithm's speed can be relatively slow as it computes the Euclidean distance for every data point in each iteration. However, faster convergence can be achieved by initializing  $\mu_k$  from a random subset of  $K$  data points within the dataset (Gustavson 2019).

In subsequent tests on transaction monitoring data, we will observe that the algorithm may not be highly suitable for assessing outliers in complex scenarios, particularly in the banking sector. The extensive iterations required for diverse customer groups and risk levels make it impractical and costly for institutions. The algorithm's efficiency is, therefore, most applicable in specific scenarios and situations.

## 3.6 Copulas

The paper by Zheng Li (2020) introduced a novel Copula-based Outlier Detection technique known as COPOD. Before this development, several prominent outlier detection methodologies had been proposed, predominantly centered around clustering principles. These methodologies typically involved computing distances between data points and flagging those exhibiting significant deviations from their neighboring points. In contrast, COPOD diverges from this paradigm by leveraging a statistical construct known as copula. A copula function is a bridge between univariate marginal distributions, facilitating the construction of their corresponding joint multivariate distribution. This feature proves particularly advantageous in detecting outliers within financial time series data. Empirical studies have consistently highlighted the non-normal nature of financial data, often characterized by skewness, leptokurtosis, and symmetric dependencies as in Joe (2014). Recent scholarly contributions underscore the growing recognition of copulas in financial modeling, driven partly

by the inadequacies of assuming multivariate normality, especially evident in transactional monitoring scenarios where multidimensional data are prevalent (Patton 2012).

COPOD exploits copulas to dissect the interrelationships among multiple features, even in scenarios where the underlying distributions of individual features are unknown (Faesel 2022). Formally, a  $d$ -variate copula, denoted as  $C : [0, 1]^d \rightarrow [0, 1]$ , represents the empirical cumulative distributive function (CDF), which describes the probability at each point that a random variable  $(U_1, U_2, \dots, U_d)$  drawn from Uniform(0,1) marginals, falls below or equals a specified point

$$C_U(u) = \mathbb{P}(U_1 \leq u_1, \dots, U_d \leq u_d) \quad (3.10)$$

where  $P(U_j \leq u_j) = u_j$  for  $j \in 1, \dots, d$  and  $u_j \in [0, 1]$ . These uniform distributions can be transformed into desired distributions via inverse sampling:

$$X_j = F_j^{-1}(U_j) \sim F_j \quad (3.11)$$

Sklar's theorem, formulated by Sklar (1959), underscores the ubiquity of copulas by revealing that any joint distribution function  $F(x_1, \dots, x_d)$  with marginal distributions  $F_1, \dots, F_d$  can be expressed in terms of copulas:

$$F(x) = C(F_1(x_1), \dots, F_d(x_d)) \quad (3.12)$$

This theorem offers considerable modeling flexibility for high-dimensional datasets, as it permits the separate modeling of each dimension with a guaranteed method for linking marginal distributions to form joint distributions. When the marginals are continuous, the copula function can be uniquely determined. By substituting the inverse transformation from equation 3.11 into equation 3.10, the copula equation can be expressed in terms of joint and inverse cumulative distribution functions:

$$\begin{aligned} C_U(u) &= \mathbb{P}(F_{X_1}(X_1) \leq u_1, \dots, F_{X_d}(X_d) \leq u_d) \\ &= \mathbb{P}(X_1 \leq F_{X_1}^{-1}(u_1), \dots, X_d \leq F_{X_d}^{-1}(u_d)) \\ &= F_X(F_{X_1}^{-1}(u_1), \dots, F_{X_d}^{-1}(u_d)) \end{aligned} \quad (3.13)$$

Together, these insights from Sklar's theorem underscore the existence of a copula for any given multivariate continuous distribution and provide a systematic approach for constructing copulas (Zheng Li 2020).

### 3.6.1 Theoretical Framework of COPOD

The theoretical framework of COPOD involves several key components:

#### A. Empirical Copula

An empirical copula is a non-parametric estimator of the copula function based on observed data, which is used by COPOD. It provides a way to estimate the underlying dependence structure among variables without making assumptions about the functional form of the copula Genest & Favre (2007). Let  $X$  denote a dataset comprising  $n$  observations in  $d$  dimensions. For clarity, we denote the  $i$ th observation of the  $j$ th dimension as  $X_{j,i}$ . The empirical CDF,  $\hat{F}(x)$  is calculated as:

$$\hat{F}(x) = \frac{1}{n} \sum_{i=1}^n I(X_i \leq x) \quad (3.14)$$

This transforms the original data into ranks, effectively converting them into uniform marginals. With the rank-transformed data  $U_{i,j}$ , the joint empirical distribution function is computed

$$\hat{C}_n(u_1, u_2, \dots, u_d) = \frac{1}{n} \sum_{i=1}^n I(\tilde{U}_{1,i} \leq u_1, \dots, \tilde{U}_{d,i} \leq u_d) \quad (3.15)$$

This function gives the probability that each variable falls below or equals a specified rank  $u_j$  for  $j = 1, 2, \dots, d$ . The empirical copula, denoted as  $\hat{C}_n(u_1, u_2, \dots, u_d)$ , is then obtained from the joint empirical distribution function. This serves as an estimator for the true copula function underlying the multivariate distribution Nelsen (2006).

#### B. Tail Probability Estimation

COPOD uses the empirical copula to approximate the tail probability of observing extreme events in the dataset. By analyzing the copula-based tail probability, COPOD identifies outliers that deviate significantly from the expected dependence structure captured by the copula. To achieve this, COPOD estimates the tail probabilities associated with observing extreme points in the data. We assume, that  $x_i$  is distributed according to some  $d$ -variate distribution function  $F_X$ , the left and right tail probabilities  $F_X(x_i)$  and  $1 - F_X(x_i)$  are computed, representing the likelihood of observing values as extreme as  $x_i$  on either end of the distribution. In case  $x_i$  is an outlier, the probability of observing a point at least as extreme as  $x_i$  should be small Zheng Li (2020). By

quantifying the rarity of extreme observations through tail probabilities, COPOD identifies potential outliers that deviate significantly from the expected behavior of the dataset.

Building on the estimation of tail probabilities, COPOD employs computational techniques to empirically compute these probabilities for each observation in the dataset. This involves leveraging the empirical copula function to approximate the tail probabilities based on the observed data. By systematically evaluating the tail behavior of the dataset across multiple dimensions, COPOD gains a comprehensive understanding of outlier characteristics and their underlying distributional properties.

In high-dimensional spaces, the probability of observing extreme events tends to decrease exponentially as the dimensionality of the dataset increases. This phenomenon, known as the curse of dimensionality Bellman (1961), poses a challenge for outlier detection algorithms like COPOD. To mitigate the impact of diminishing tail probabilities, COPOD adopts a logarithmic transformation of the tail probabilities. If we consider equation 3.15 in both low dimensional and high dimensional settings, we see, that as dimensionality increases, the probability of  $\tilde{U}_{j,i} \leq u_j, \forall j$  decreases exponentially (Zheng Li 2020). By taking the negative logarithm of the tail probabilities, COPOD ensures that the probabilities remain relevant and informative even in high-dimensional settings, thereby enhancing the accuracy of outlier detection.

$$\begin{aligned} -\log(\hat{C}(u)) &= -\log(\mathbb{P}(\tilde{U}_{1,i} \leq u_1) \times \dots \times \mathbb{P}(\tilde{U}_{d,i} \leq u_d)) \\ &= -\sum_{j=1}^d \log(\mathbb{P}(\tilde{U}_{j,i} \leq u_j)) = -\sum_{j=1}^d \log(u_j) \end{aligned} \quad (3.16)$$

### C. Skewness correction

The skewness of the dataset is essential in determining whether to use left or right tail probabilities for outlier detection. When outliers exhibit skewed behavior, traditional methods relying solely on one tail may produce suboptimal results. COPOD addresses this by introducing a skewness correction mechanism that adapts the selection of tail probabilities based on the skewness of each dataset dimension. By dynamically adjusting the tail probabilities, COPOD enhances its ability to detect outliers across various distributional characteristics, improving robustness and effectiveness.

In transaction monitoring, the expertise of analysts is crucial. Typically, the

focus is on the upper-tail for detecting anomalies. For instance, in a scenario where transactions exceeding a certain amount and their frequency surpass specified thresholds, only the upper-tail is relevant. Conversely, a scenario that monitors structured cash amounts would focus on lower-tail probabilities.

By understanding and applying the appropriate tail probabilities, analysts can more effectively monitor and detect potential money laundering activities, ensuring that the transaction monitoring system operates optimally and accurately.

# Chapter 4

## Data

In this section, a comprehensive description of the data, along with its implementation will be provided. Initially, we detail the creation process of synthetic data designed to mimic the transactional data structure used in transaction monitoring. These data were created to follow the behavior of a specific customer group. Following this, we focus on the examination of anonymized data obtained from real-world institutions, along with their advantages and limitations.

### 4.1 Artificial Data

Constructing artificial data is straightforward since we do not need to navigate complex layers of data typically obtained from real financial institutions, which would require detailed descriptions of their construction, which will be further introduced in the section dedicated to real-world data. However, to grasp the underlying structure of this data, a few key points are worth mentioning.

For instance, as previously discussed, customer segmentation plays a pivotal role in the transaction monitoring pipeline. Outlier detection models are specifically applied to transactional data within defined customer segments to ensure accurate assessment, avoiding misleading outliers from customers with vastly different behavioral patterns. In constructing artificial data, this segmentation aspect is unnecessary, as the data inherently represents a synthetic segment of customers with uniform behavior.

That being said, the default transactional data are designed with the following characteristics based on industry knowledge to reflect an average customer segment:



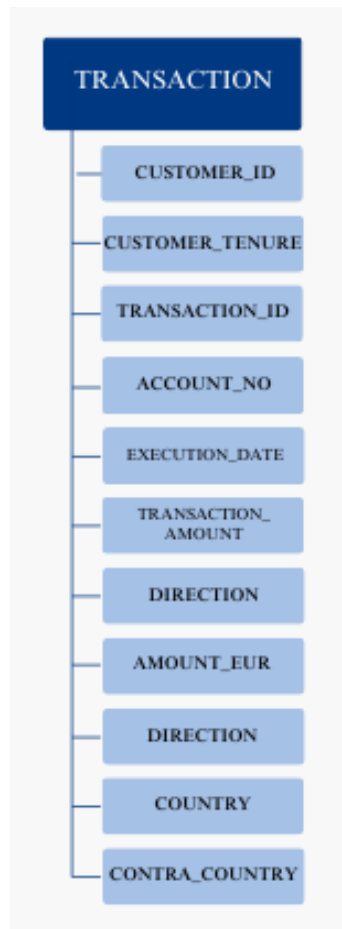
- 100,000 transactions
- 500 unique customers
- 10 different transaction types (e.g., wire, cash, etc.)
- 25% of transactions are international (with 10% involving high-risk countries)
- Spanning a one-year time period
- Chi-squared distribution (6 degrees of freedom) ( $\chi_6^2$ )

The default data settings are consistently applied throughout the analysis. The structure of the table is illustrated in the conceptual schema in Figure 4.1, and the distribution of transactions is depicted in Figure 4.2. The adoption of a chi-squared distribution with 6 degrees of freedom is consistent with one of the standard transaction distributions described in earlier chapters. This choice underscores a critical aspect of our analysis: legitimate transactions, which may not involve any illicit activity, can still be flagged as suspicious and identified as outliers in our models during transaction monitoring. For the purposes of this analysis, transaction amounts are considered in EUR. Customer tenure is randomly assigned, with 1% of customers having a tenure of fewer than 180 days.

The columns and their specifications are as follows:

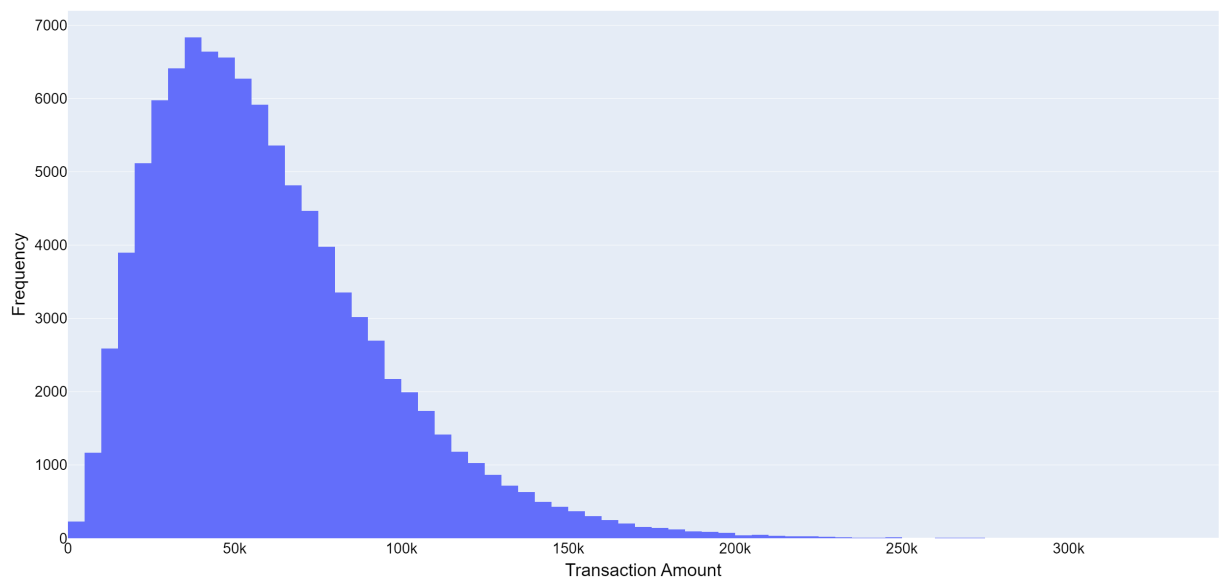
- **CUSTOMER\_ID**: Links the transaction to a specific customer.
- **CUSTOMER\_TENURE**: Number of days the customer has been with the institution.
- **TRANSACTION\_ID**: Unique identifier for each transaction.
- **ACCOUNT\_NO**: Links the transaction to a specific account.
- **EXECUTION\_DATE**: Date when the transaction occurred.
- **TRANSACTION\_AMOUNT**: Amount of the transaction converted to EUR (as scenarios typically operate in a single currency).
- **DIRECTION**: Transaction direction (credit or debit).
- **COUNTRY**: Customer's country.

Figure 4.1: Conceptual schema of the artificial data



Source: Own elaboration

Figure 4.2: Default distribution of artificial transactions



Source: Own elaboration

- **CONTRA\_COUNTRY**: Counterparty's country.
- **INTERNATIONAL\_FLAG**: Boolean indicating if the transaction is international.
- **CASH\_FLAG**: Boolean indicating if the transaction involves cash.
- **BUSINESS\_TYPE**: Type of transaction (wire, cash, etc.).

To meet specific scenario requirements and ensure applicability, the data must be further adjusted so that each scenario has its unique dataset with outliers that correspond to the specific suspicious activities targeted by the scenario. Therefore, each dataset includes some outliers, whose amounts were selected randomly to maintain the randomness characteristic of real transactional data. From a business perspective, as it is not feasible to determine the exact number of outliers with certainty, the choice is informed by business experience and the targeted number of transactions to be considered outliers. Financial institutions typically cannot investigate every flagged transaction due to the cost implications and must balance the cost of investigations with potential regulatory penalties for insufficient transaction monitoring coverage. As such, the typical percentage of outliers varies between 5-15%. For each scenario, we will select a random percentage with a mean of 10 and a standard deviation of 5. The detailed approach for each scenario will be further explained in Section 4.3. The outliers will be either increased or decreased by a factor known as the outlier factor. This factor is determined randomly, with an average increase set to 3 and a standard deviation of 2.

## 4.2 Real-world Data

To ensure streamlined analysis and compatibility across different Transaction Monitoring systems, our data structure remains independent of any specific TM system and adheres strictly to the analytical requirements.

For this use case, we collected anonymized data from a real-world, medium-sized financial institution spanning 12 months. This timeframe ensures that our models can capture seasonality and other annual trends effectively. A 12-month period is typically sufficient for training transaction monitoring models, striking a balance between efficiency and the ability to gather essential information.

In contrast to artificially created data, the real-world data we employ includes all information provided by the institution. This encompasses details

about alerts generated by scenarios, although these are not the primary focus of our analysis, as well as data on customers who did not conduct transactions during the observed period, necessitating additional data preprocessing.

To facilitate data representation and management, we have identified and defined six distinct tables that comprehensively represent our input data. Figure 4.3 below illustrates the conceptual schema of these tables which are:

- transactional
- alert
- customer
- threshold
- alert-scenario mapping
- alert-feature mapping

The obtained tables contain the following information:

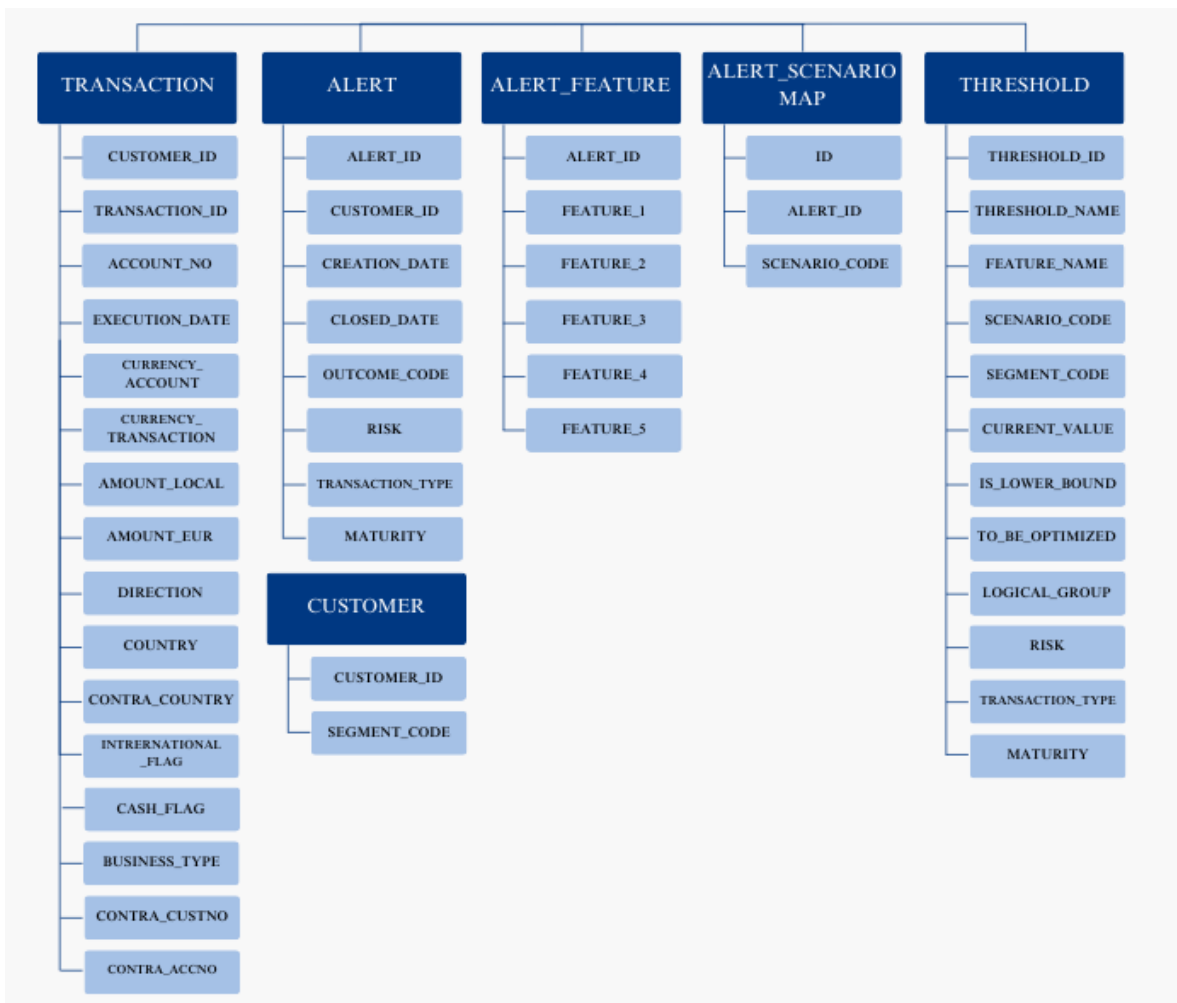
- 8,879,643 transactions
- 3,493,688 customers
- 17 segments
- 35 distinct scenarios
- 6,120 alerts

#### *Transactional Table*

The Transactional table includes essential information crucial for our analysis, as transactions are the primary input for training the models in this research. It encompasses details about all transactions made during the observed period at the institution. The columns presented in the Transactional table are:

- **CUSTOMER\_ID**: Links the transaction to the particular customer.
- **TRANSACTION\_ID**: Unique identifier of the transaction.
- **ACCOUNT\_NO**: Links the transaction to the particular account.
- **EXECUTION\_DATE**: Date when the transaction was executed.

Figure 4.3: Conceptual scheme of the real-world data



Source: Own elaboration

- **CURRENCY\_ACCOUNT**: Currency in which the account making the transaction is held.
- **CURRENCY\_TRANSACTION**: Currency in which the transaction was made.
- **AMOUNT\_LOCAL**: The amount of the transaction in the currency of the transaction.
- **AMOUNT\_EUR**: The amount of the transaction converted to EUR (as scenarios typically function in one currency).
- **DIRECTION**: Direction of the transaction (credit or debit).
- **COUNTRY**: Country of the customer.
- **CONTRA\_COUNTRY**: Country of the counterparty.
- **INTERNATIONAL\_FLAG**: Boolean value indicating if it's an international transaction.
- **CASH\_FLAG**: Boolean value indicating if it's a cash transaction.
- **BUSINESS\_TYPE**: Type of transaction (wire, cash, etc.).
- **CONTRA\_CUSTNO**: Unique identifier of the counterparty (if applicable).
- **CONTRA\_ACCNO**: Account number of the counterparty.

#### *Alert table*

The dataset includes a comprehensive record of alerts generated throughout the analyzed period. It encompasses crucial details such as the unique alert identifier, creation date, closed date (if applicable), investigation outcome, and additional fields like customer maturity and transaction type.

During the investigative process, each alert may result in one of three outcomes:

- **False Positive (FP)**: Alerts where no substantial evidence of suspicious activity is found.
- **Worthy Alert (WA)**: Alerts escalated for further investigation, but without sufficient evidence to confirm suspicious activity, resulting in closure as WA.

- Suspicious Activity Report (SAR): Alerts thoroughly investigated and substantiated as suspicious, subsequently reported to regulatory authorities.

Key fields in the Alert table include:

- **ALERT\_ID**: Unique identifier of the alert within the TM system.
- **CUSTOMER\_ID**: Key linking to the customer table.
- **CREATION\_DATE**: Date when the alert was generated.
- **CLOSED\_DATE**: Date when the alert was closed, if applicable.
- **OUTCOME\_CODE**: Outcome of the investigation (provided only if the alert has been closed).
- **RISK**: Risk category assigned to the transaction.
- **TRANSACTION\_TYPE**: Type of transaction associated with the alert.
- **MATURITY**: Length of the customer's relationship with the institution at the time of the transaction.

#### *Customers Table*

The Customers table contains essential customer data, excluding personally identifiable information necessary for analysis. It includes information about customer segments, crucial for accurately classifying customers based on similar patterns of behavior. These segments were created using behavioral segmentation, an approach recommended by the FATF since 2021 as the optimal choice, with its implementation detailed in a previous chapter. The table features the following key columns:

- **CUSTOMER\_ID**: This column serves as a unique identifier for each customer. If actual customer IDs are withheld by the bank, data can be anonymized, and new identifiers can be generated.
- **SEGMENT\_CODE**: This column provides a unique identifier for different customer segments, such as Private Individuals, Legal Entities, SMEs, and others.

*Alert-Feature Mapping Table*

The Alert-Feature Mapping table records variables calculated by scenarios for each alert. The number of attributes (features) varies based on the scenarios being optimized and their respective thresholds. The columns in this table include:

- **ALERT\_ID**: A unique identifier of an alert in the TM system.
- **FEATURE\_X**: Variable number X.

*Alert-Scenario Mapping Table*

The Alert-Scenario Mapping table associates alerts with the respective scenarios that generated them. In some systems, an alert may be triggered by multiple scenarios. This table includes the following columns:

- **ALERT\_ID**: A unique identifier of an alert in the TM system.
- **SCENARIO\_CODE**: A unique identifier of the scenario that triggered the alert.

*Threshold Table*

The Threshold table stores the configuration for each scenario in the form of thresholds. It includes the following columns:

- **THRESHOLD\_ID**: A unique identifier of the threshold amount.
- **THRESHOLD\_NAME**: A unique code assigned to the threshold.
- **FEATURE\_COLUMN**: The name of the column containing the threshold value.
- **SCENARIO\_CODE**: A unique identifier of the scenario to which the threshold belongs.
- **SEGMENT\_CODE**: A unique identifier of the segment to which the threshold is applicable.
- **CURRENT\_VALUE**: The current value of the threshold.
- **IS\_LOWER\_BOUND**: A Boolean value indicating whether the threshold is a lower bound (1) or an upper bound (0).



- **TO\_BE\_OPTIMIZED**: A Boolean value indicating whether the threshold value will be reconfigured.
- **LOGICAL\_GROUP**: A unique identifier of the logical group. Thresholds within the same logical group have a condition 'AND' between them.
- **RISK**: The risk category for which the threshold is defined.
- **MATURITY**: The maturity of the client for whom the threshold is configured – time the customer has spent with the institution.

### 4.3 Data Preparation

Figure 4.4 illustrates a comprehensive six-step methodology for the parametrization process using the current segmentation model. The analytical focus is on achieving an optimal balance between compliance costs, primarily reducing false positives, and the risks of non-compliance, where potential instances of suspicious activity might go undetected.

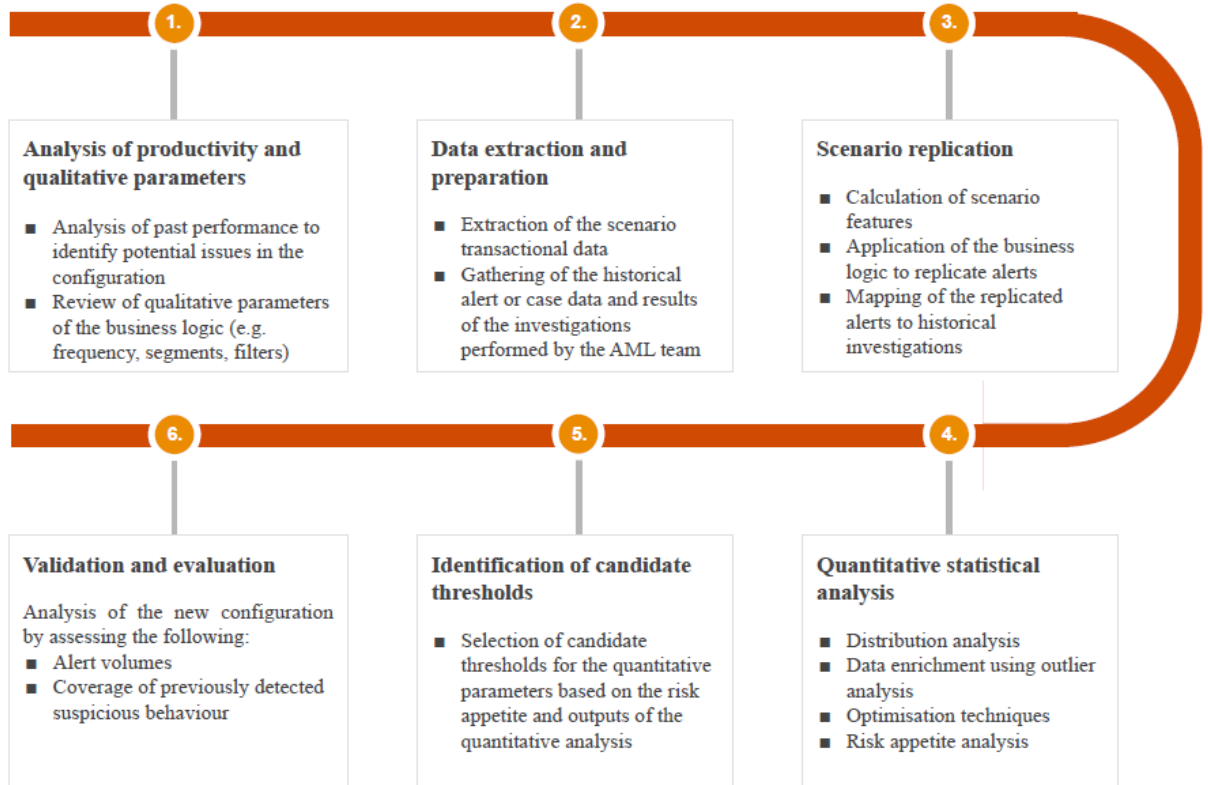
The preceding section described the initial data required for analysis. However, further preprocessing is necessary to apply this data to outlier detection models and to align it with the parametrization process described above. The first step in this preprocessing phase involves consolidating the data into a single table that contains all necessary information. This step is unnecessary for artificial data, as it has been structured accordingly from inception. For real-world data, all pertinent information scattered across various tables must be integrated into a transactional table, which serves as the primary source for feature computation.

Given that the transactional data encompasses the entire customer portfolio, segmentation information from the customer table will be used to differentiate transactions by specific segments.

Figure 4.5 displays transaction counts across different segments and risk categories. For the analysis, we will focus on the Individuals segment and the Medium-risk category, chosen for its adequate transaction volume and relatability to the average reader's understanding of individual transactional behavior.

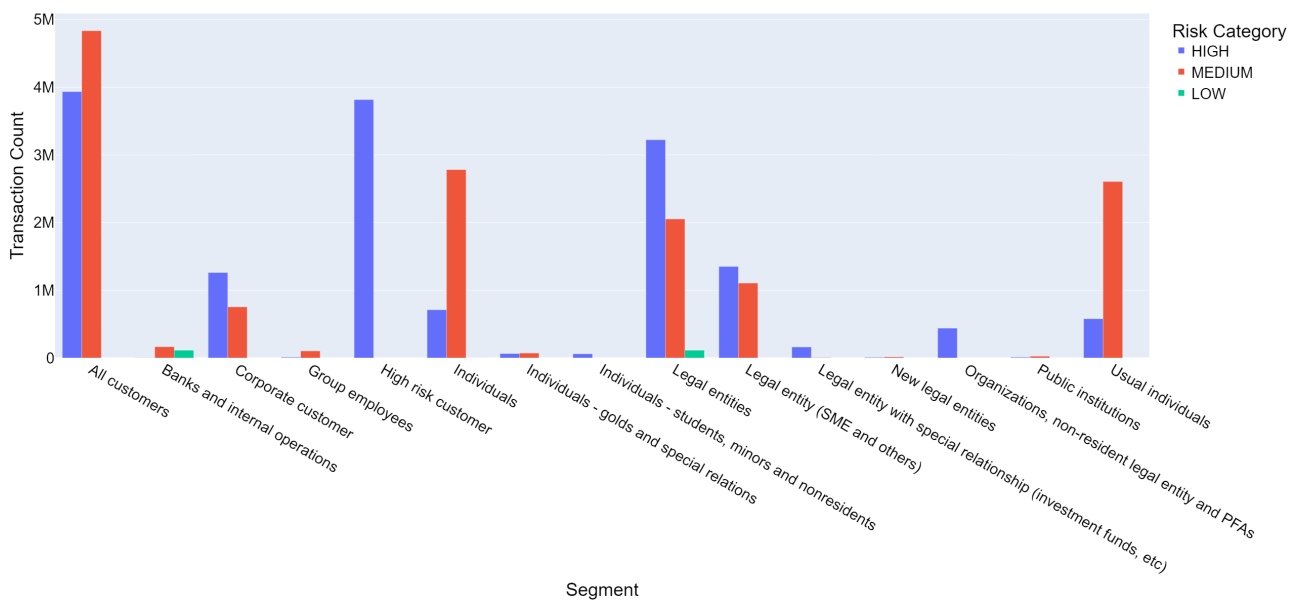
Next step is to integrate tables containing information about generated alerts. It is notable that this institution generated a minimal number of alerts across all scenarios, which suggests that thresholds may have been set too high.

Figure 4.4: Methodology of the parametrization process



Source: Own elaboration

Figure 4.5: Transaction count by segment and risk category



Source: Own elaboration

Moreover, the same thresholds were applied across all customer in the portfolio as visible in the figure 4.5, the bank employs the general segment which includes all the customers in its portfolio "All Customers". This likely contributed to underperformance in scenario detection, given the diverse behavioral patterns within this large segment.

As a result, the dataset provided lacks labeled instances because no alerts were generated, precluding conventional quantitative evaluation. This limitation will be further discussed in Chapter 6. Consequently, the evaluation of this dataset will primarily adopt a qualitative approach, enabling an initial exploration of model performance in the absence of established ground truth. Quantitative metrics will be assessed solely on the artificial dataset, where labels are generated during outlier creation.

After completing the initial data preprocessing phase, we proceed with specific preprocessing tailored to each scenario. In the first scenario, the objective is straightforward: identifying weekly debit transactions that significantly exceed a predefined threshold. To prepare the dataset for the outlier detection model, we focus solely on the relevant features. Specifically, we calculate the total sum of debit transactions made by customers on a weekly basis across the entire period. This aggregated sum will serve as the input feature for our analysis. For the artificial data, we must manually introduce outliers. Here's the approach: we randomly select a number of transactions to designate as outliers, adhering to the earlier mentioned percentage range (typically between 5-15%). Each selected outlier transaction amount is increased by an outlier factor.

The second scenario will consist of two features, as we need to identify two potential thresholds. We first need to calculate the parameters of the scenario. In this case, we have four parameters to calculate:

- *WireCreditAmount\_Customer\_PresentDay*,
- *WireCreditAmount\_Customer\_Past\_n\_Month*,

where  $N = 3$ . To calculate these parameters, we will first aggregate the amounts of wire credit transactions made by each customer on a given day. Then, for each day, we will create 30, 60, and 90-day lookback periods and calculate the amounts of wire credit transactions made by each customer during these periods. Using these parameters, we will derive the final features.

To illustrate how the features are calculated, we can modify the scenario syntax into Equation 4.3:

$$\begin{aligned}
X_1 &> 1000 \times T_1 \\
X_1 &> T_2 \times 0.01 \times \left( \frac{Y_1 + Y_2 + Y_3}{0.001 + (Y_1 > 0) + (Y_2 > 0) + (Y_3 > 0)} \right)
\end{aligned} \tag{4.1}$$

where:

- $X_1 = \text{WireCreditAmount\_Customer\_PresentDay}$ ,
- $T_1 = \text{Threshold\_1}$ ,
- $T_2 = \text{Threshold\_2}$ ,
- $Y_n = \text{WireCreditAmount\_Customer\_Past\_n\_Month}$ ,

Next, we will manipulate the equation to isolate both thresholds:

$$\begin{aligned}
\frac{X_1}{1000} &> T_1 \\
\frac{X_1}{0.01 \times \left( \frac{Y_1 + Y_2 + Y_3}{0.001 + (Y_1 > 0) + (Y_2 > 0) + (Y_3 > 0)} \right)} &> T_2
\end{aligned} \tag{4.2}$$

These two expressions will be our features:

$$\begin{aligned}
\frac{X_1}{1000} &= F_1 \\
\frac{X_1}{0.01 \times \left( \frac{Y_1 + Y_2 + Y_3}{0.001 + (Y_1 > 0) + (Y_2 > 0) + (Y_3 > 0)} \right)} &= F_2
\end{aligned} \tag{4.3}$$

where  $F1 = \text{Feature 1}$  and  $F2 = \text{Feature 2}$ . Similarly to the previous scenario, we will further adjust these results for the artificial dataset to create outliers. In this instance, we will separately increase both features using a random outlier factor.

The final scenario will consist of four features, as we are identifying four thresholds. Similar to previous scenarios, we first need to calculate the parameters, which are:

- $\text{WireCreditAmount\_Customer\_Past\_n\_Month}$ ,
- $\text{WireCreditNumber\_Customer\_Past\_n\_Month}$ ,
- $\text{NumberDays\_Customer\_Open} \geq 180$ .

Where  $N = 6$ . In this case, the parameter *NumberDays\_Customer\_Open* is used for filtering the initial dataset rather than for calculation. Since the scenario partially focuses on transactions with a 180-day lookback, we need to consider only customers who have been clients of the bank for at least the last 180 days. Therefore, this parameter will be used to remove customers who do not meet this condition.

For the remaining customers, we will create 30, 60, 90, 120, 150, and 180-day lookback periods and calculate the amounts and counts of wire credit transactions made by each customer during these periods. Using these parameters, we will derive the final features.

As before, we will modify the scenario syntax into Equation 4.4:

$$\begin{aligned}
 X_1 &> T_1 \\
 Y_1 &> T_2 \\
 (X_1 + X_2 + X_3 + X_4 + X_5 + X_6) &< T_3 \\
 (Y_1 + Y_2 + Y_3 + Y_4 + Y_5 + Y_6) &< T_4
 \end{aligned} \tag{4.4}$$

where:

- $T_n = \text{Threshold}_n$ ,
- $X_n = \text{WireCreditAmount\_Customer\_Past}_n\text{\_Month}$ ,
- $Y_n = \text{WireCreditNumber\_Customer\_Past}_n\text{\_Month}$ ,

From this, we can derive the features as follows:

$$\begin{aligned}
 X_1 &= F_1 \\
 Y_1 &= F_2 \\
 (X_1 + X_2 + X_3 + X_4 + X_5 + X_6) &= F_3 \\
 (Y_1 + Y_2 + Y_3 + Y_4 + Y_5 + Y_6) &= F_4
 \end{aligned} \tag{4.5}$$

where  $F_n$  are the four features. Additionally, for the artificial dataset, we will create the outliers by increasing Feature 1 and Feature 2 by a random outlier factor and, conversely, decreasing Feature 3 and Feature 4 by the same random outlier factor.

With this approach, we have created six feature datasets: three from artificial data with manually added outliers and three from real-world unlabeled data. Each dataset is tailored to a particular scenario and its outlier detection requirements.

# Chapter 5

## Model Implementation

In this section, we delineate the process of constructing each of our models and determining the significance of the variables. Additionally, we explain the business logic and underlying motivations for the adopted methodologies. The models will be applied to the dataset described in the preceding chapter.

For the purpose of this thesis, the primary evaluation and tuning of the models will be conducted on synthetic data, since, in real-world scenarios, labeled transactions are often unavailable or inconclusive, as will be further discussed in Chapter 6.

This thesis employs two approaches for model tuning. In the context of transaction monitoring, the objective is to maximize the correct classification of transactions as fraudulent, which we define as outliers. To achieve this, the institutions adopt the cost-sensitive approach presented by Elkan (2001). The goal is to enhance the performance of the models on the imbalanced sample. While it might seem straightforward to choose a performance metric for transaction monitoring, this is not the case. Ideally, marking all suspicious transactions as potentially fraudulent would be logical. However, in reality, each flagged transaction undergoes a review process by an employee, which varies by transaction but is typically time-consuming and costly for the institution. Moreover, if too many transactions are flagged, the investigation team might be unable to thoroughly investigate all cases, necessitating either additional hires or incurring high fees from regulatory bodies.

Our analysis aims to balance two performance metrics: sensitivity and specificity. Sensitivity refers to a model's ability to correctly identify fraudulent transactions, measuring the proportion of frauds labeled as fraudulent, thus correlating with a low false negative rate. Specificity, conversely, refers to an

algorithm’s ability to correctly identify genuine transactions, measuring the proportion of valid transactions labeled as valid, thus correlating with a low false positive rate. Our aim is to achieve high sensitivity while maintaining sufficiently high specificity. Precision is also considered as a complementary performance measure, describing the proportion of true positives among all transactions labeled as fraudulent, where higher precision indicates a low false positive rate.

Furthermore, we examine the Area Under the ROC Curve (AUC). In our context, a high AUC is necessary but not sufficient to conclude that our model performs well due to the significant imbalance between fraudulent and genuine transactions in our dataset. As described in Fayzrakhmanov *et al.* (2020), the usage of ROC curve might be suboptimal in such cases. A high AUC might be misleading if our model predominantly labels transactions as legitimate, rendering it ineffective. Nonetheless, AUC cannot be disregarded entirely as there remains a risk of mislabeling both fraudulent and legitimate transactions. Thus, while ensuring that our models achieve high AUC, we must also consider other performance metrics.

Drawing upon our findings from the real-world dataset, we provide a detailed rationale for selecting an appropriate threshold, informed by the outcomes of various outlier detection models and aligned with the established business criteria. The objective is to determine the number of potential alerts each algorithm identifies and to evaluate the performance and suitability of each model based on these figures.

However, merely assessing the outliers identified by the model is insufficient. The nature of outliers can vary significantly throughout the feature space, necessitating a more nuanced approach to threshold selection. In this context, we aim to identify the tail value in the data where outliers become more prevalent or where the entirety of the data can be classified as outliers across all feature spaces.

To achieve this, the identified outliers from each model are first applied separately to each feature. This step involves sorting the data by each feature’s values to determine the distribution of outliers within individual feature spaces. By doing so, we can assess where outliers begin to cluster and at what point they become more common.

After this feature-wise assessment, the sorted data are merged to identify a tail value point. This tail value is critical as it represents the threshold where outliers are consistently flagged across multiple features. By examining

these tail values, we can pinpoint the threshold where outliers begin to emerge uniformly across the entire feature space.

This comprehensive approach ensures that the selected threshold accurately reflects the areas of the data where outliers are most likely to occur, providing a robust basis for evaluating the appropriateness of the results. By considering the tail value across all features, we ensure that our outlier detection is both thorough and aligned with the practical needs of the business, leading to more reliable and actionable insights.

It is important to note that after the initial threshold proposal, the final choice of thresholds may still differ based on the institution's risk appetite. Even though some thresholds might seem optimal from a purely analytical perspective, they might not align with the institution's risk tolerance and operational strategies. Hence, the ultimate threshold selection must balance analytical rigor with the institution's specific risk management policies.

## 5.1 Isolation Forest

We employ Isolation Forest as our initial model for detecting outlier transactions. For this analysis, we use the *IsolationForest* package. The outlier detection process with Isolation Forest comprises two stages. The first stage, the training stage, involves building isolation trees using subsamples of the training set. The second stage, the testing stage, involves passing the test instances through the isolation trees to obtain an anomaly score for each instance.

In the training phase, isolation trees (iTrees) are built by recursively splitting the training dataset until each instance is isolated or a predetermined tree height is achieved, resulting in a partial model. The number of splits required to isolate a sample corresponds to the path length  $l$  from the root to the terminal node, determined by the subsampling size  $\psi$  :  $l = \text{ceiling}(\log_2 \psi)$ . The average path length across a forest of such trees acts as a measure of normality, forming the basis of our decision function. We focus on data points with shorter-than-average path lengths, as these are more likely to be anomalies.

The Isolation Forest algorithm relies on two primary parameters: the subsampling size  $\psi$  and the number of trees  $t$ . The subsampling size dictates the amount of training data used. Once  $\psi$  reaches an optimal value, further increases do not improve detection performance but do increase processing time and memory requirements. Empirical evidence suggests that setting  $\psi$  to 256 generally suffices for effective anomaly detection across various datasets (Liu



*et al.* 2015). Thus, we adopt  $\psi = 256$  as our default value. The number of trees  $t$  determines the size of the ensemble. Path lengths typically stabilize well before  $t = 100$  (Liu *et al.* 2015), so we use  $t = 100$  as our default value unless specified otherwise.

At the conclusion of the training process, a collection of trees is prepared for the evaluation phase. An anomaly score  $s$  is calculated from the expected path length  $E(h(x))$  for each test instance by passing it through the isolation trees. The path length  $h(x)$  is the number of edges  $e$  from the root to the terminal node. If  $x$  terminates at an external node with  $Size > 1$ , the value returned is  $e$  plus an adjustment  $c(Size)$ . Once  $h(x)$  is obtained for each tree, an anomaly score  $s(x, \psi)$  is computed as in Equation 3.1. The complexity of this process is  $O(nt \log \psi)$ , where  $n$  is the testing data size. To find the top  $m$  anomalies, the data is sorted by  $s$  in descending order, and the first  $m$  instances are identified as anomalies.

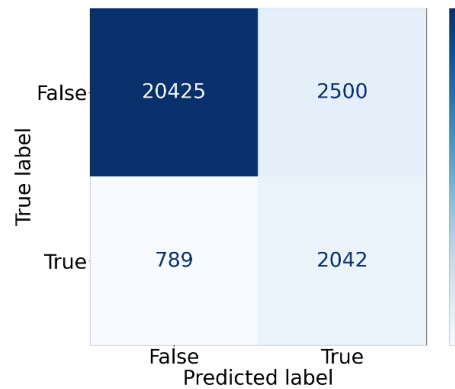
We begin with the results of Scenario 1 on the artificial dataset. The predictive performance is illustrated in Figure 5.1. The Isolation Forest correctly identified 2,042 outliers while generating 20,425 false positives. This model achieved a specificity of 89%, indicating its capacity to correctly classify 89% of potentially genuine transactions. Despite this, it underperformed compared to the K-Means and Copulas algorithms in terms of specificity.

However, the Isolation Forest demonstrated a sensitivity of 72.1%, outperforming its peers in this metric. Notably, while it identified 2,500 transactions as potentially fraudulent, it misclassified only 789 transactions as genuine, which may have been fraudulent. This scenario exemplifies the trade-off between true positives and false positives that many financial institutions must navigate. Although the Isolation Forest excels in identifying fraudulent transactions, its accuracy in correctly identifying genuine transactions is limited. This limitation could lead to increased costs for financial institutions due to the need for additional investigative efforts.

Figure 5.2 presents the ROC curve for the Isolation Forest model, with an area under the curve (AUC) of 0.86. This AUC is slightly better compared to other models, with COPOD showing a similar performance. Furthermore, the average prediction time for the Isolation Forest was 2.29 seconds. While this is shorter than the K-Means model, it is significantly longer than COPOD's 0.02 seconds, especially given the relatively small dataset used in this scenario. This indicates that although the Isolation Forest's performance is satisfactory, its processing time is suboptimal. This inefficiency could become problematic

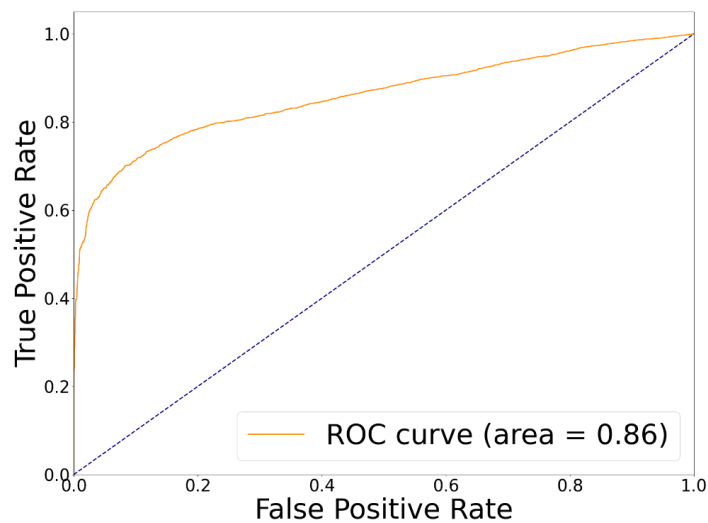
with larger datasets, given the extensive number of results required to tune the entire customer portfolio and all scenarios.

Figure 5.1: Confusion Matrix of Isolation Forest - Scenario 1



Source: Own elaboration

Figure 5.2: ROC Curve of Isolation Forest - Scenario 1



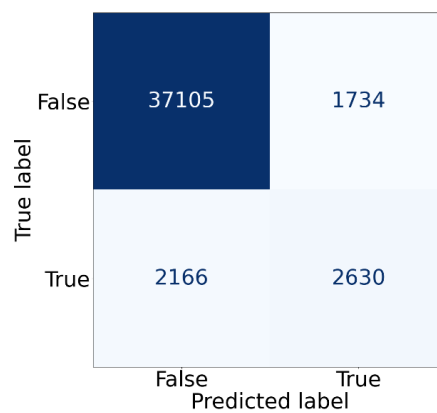
Source: Own elaboration

The results of the second scenario, involving two dimensions, are presented in Figure 5.3. In this scenario, the Isolation Forest correctly identified 2,630 fraudulent transactions while generating 37,105 false positives. Although the Isolation Forest's specificity increased to 95.5%, it still underperformed compared to other models, though the performance gap was not significant. Conversely, the model's sensitivity decreased to 54.8%, which, although similar to the performance of the other models, is unsatisfactory. Unlike the one-dimensional scenario, the two-dimensional model falsely identified more trans-

actions as genuine rather than fraudulent. This misclassification could lead to penalties due to insufficient monitoring of fraudulent transactions.

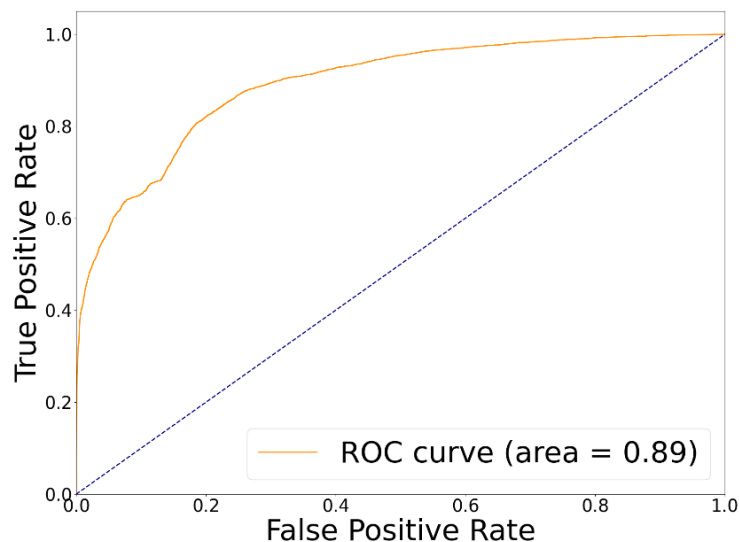
The ROC curve for the two-dimensional Isolation Forest model is shown in Figure 5.4, with an AUC of 0.89. This value remains comparable to those of the other two models. However, the processing time increased significantly to 6.32 seconds. This finding underscores the previously mentioned concern regarding high processing times, which could pose significant challenges for larger financial institutions.

Figure 5.3: Confusion Matrix of Isolation Forest - Scenario 2



Source: Own elaboration

Figure 5.4: ROC Curve of Isolation Forest - Scenario 2



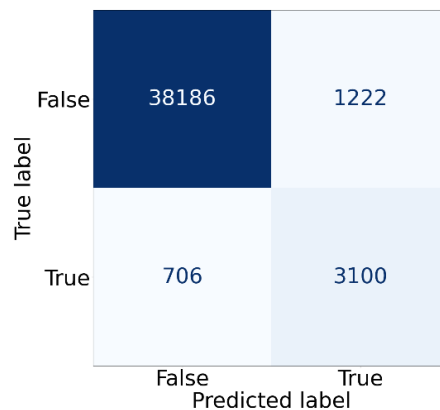
Source: Own elaboration

The four-dimensional analysis, represented as Scenario 3, is shown in Figure 5.5. In this scenario, the Isolation Forest significantly outperformed both

the K-Means and Copulas models, although its results remained close to those of COPOD. Notably, in higher dimensions, the Isolation Forest's specificity increased to 96.9%, and its sensitivity rose to 81.5%, indicating improved performance with higher-dimensional data. The model correctly identified 3,100 fraudulent transactions and misclassified only 706 transactions as false positives. Additionally, it classified 38,186 transactions as genuine, with only 1,222 incorrectly classified as potentially fraudulent. These results suggest that the Isolation Forest model yields satisfactory outcomes in higher dimensions.

Figure 5.6 illustrates the ROC curve for the four-dimensional Isolation Forest model, which boasts an impressive AUC of 0.98. This performance surpasses that of the K-Means model and matches the results of COPOD. Nonetheless, the processing time for the Isolation Forest model is still a concern, particularly when compared to COPOD, which keeps its processing times under 0.2 seconds.

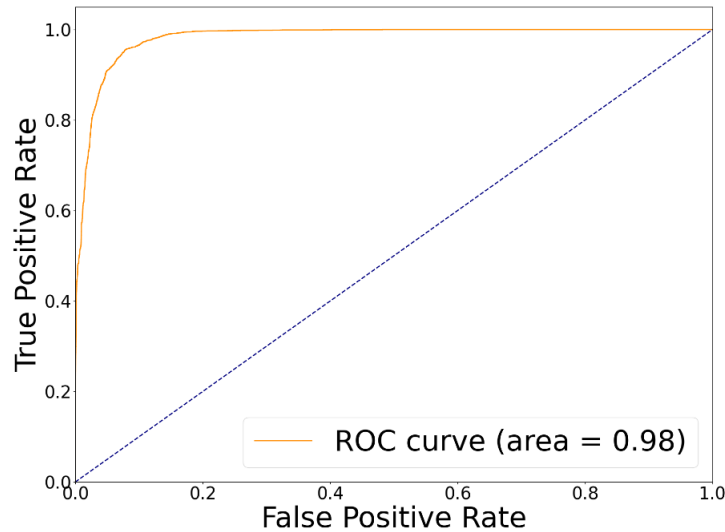
Figure 5.5: Confusion Matrix of Isolation Forest - Scenario 3



True label	False	38186	1222
	True	706	3100
		False	True
		Predicted label	

Source: Own elaboration

Figure 5.6: ROC Curve of Isolation Forest - Scenario 3



Source: Own elaboration

As we transition to analyzing real-world data, we face the challenge of working without labeled data, making it difficult to directly assess the correct classification of the models. Instead, we evaluate the models based on business needs and the rationale behind the chosen thresholds. The Table 5.1 shows the number of transactions evaluated as fraudulent and genuine in each scenario:

Scenario	Total features	Outliers	Outliers (%)	Time (s)
1	211,499	29,000	13.71%	24.1952
2	379,969	58,389	15.37%	31.3524
3	328,674	36,332	11.05%	28.0342

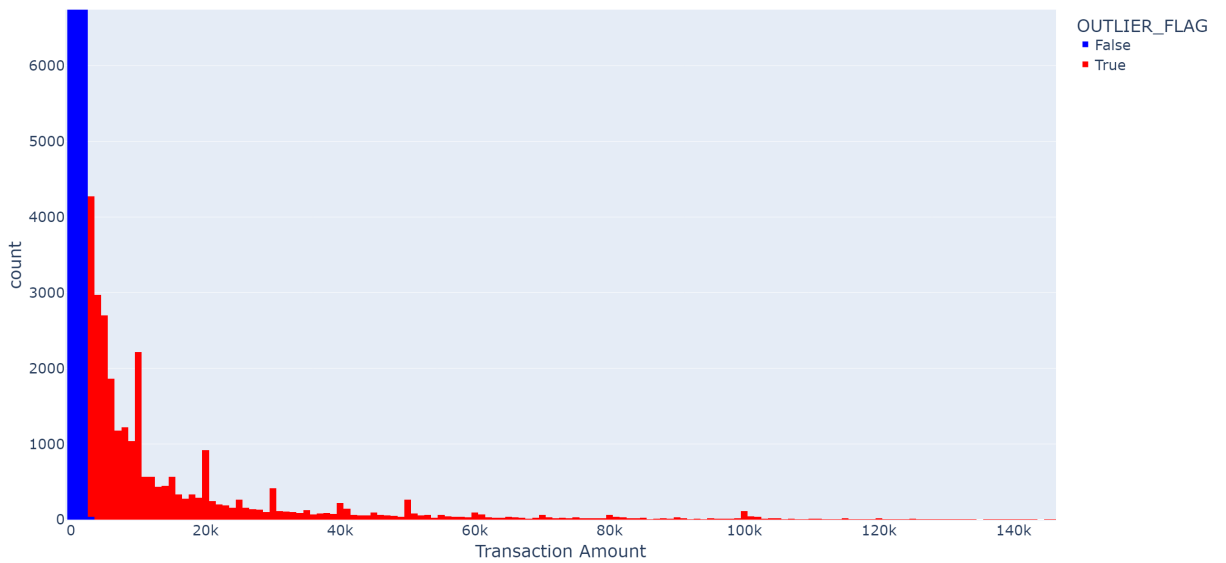
Table 5.1: Performance metrics of Isolation Forest on real-world data

In Scenario 1, the process is relatively straightforward. We assess the threshold at the point where transactions start appearing as outliers according to the model's feature distribution. Figure 5.7 illustrates this distribution, zoomed in for clarity due to the large number of transactions and the extensive tail.

Based on this figure, the ideal cut-off point would be a threshold value of 1,500 EUR. This means that from the total 211,499 transactions, the system would identify 29,000 alerts. It is crucial to note that this scenario was introduced as an example of a one-dimensional analysis and, in practice, would be more complex or used in combination with other scenarios, significantly reducing the number of alerts to avoid a high number of false positives.

Given the impracticality of investigating 29,000 alerts, we would look further

Figure 5.7: Feature Distribution - Scenario 1 [zoomed in]



*Source:* Own elaboration

down the distribution to find an optimal cut-off point that aligns with the number of alerts the institution can feasibly investigate. In the context of a one-dimensional scenario, the benefit of using Isolation Forest (similarly to the COPOD algorithm) is the ability to choose the contamination level at the outset. This means that Isolation Forest can identify the number of alerts that fit the institution’s risk-based approach and cost considerations.

For Scenarios 2 and 3, the situation is more complex due to the multi-dimensional nature of the data. Here, assessing the threshold based on a single distribution function is not feasible. Additionally, setting the contamination level is less straightforward because we need to evaluate thresholds separately for each feature. This requires further preprocessing of the results as previously explained in the model implementation section.

The post-preprocessing results are shown in Table 5.2, indicating a decrease in the number of potential alerts:

Scenario	Total transactions	Outliers after applying the threshold	Outliers (%)
2	379,969	5,227	1.38%
3	328,674	14,530	4.42%

Table 5.2: Outliers detected by Isolation Forest after applying the thresholds to scenarios

In Scenario 2, the number of alerts decreased to 5,277, representing 1.38% of the total features, if we would apply the respective thresholds at the as-

sessed cut-off points. This proportion is manageable for most institutions to investigate and aligns well with average institutional needs.

For Scenario 3, the number of alerts is higher, totaling 14,530. This increase is due to the scenario being run daily, leading to potential double counting, particularly in the first half of each month. However, many of these alerts would eventually be merged as duplicates, reducing the final count to a more manageable number suitable for threshold proposal.

Even though the overall results seem satisfactory, there is a significant downside to using Isolation Forest in this analysis: its high processing times. These processing times are notably higher than those of our other models, which raises concerns about its efficiency and practicality for real-world applications. High processing times can limit the model's usability. Therefore, while Isolation Forest demonstrates strong performance in identifying outliers, its extended processing duration makes it a less efficient choice compared to faster alternatives like COPOD, which maintain lower processing times while delivering comparable results. This efficiency issue must be considered when choosing the most suitable model for practical deployment in a business setting.

## 5.2 K-Means

The utilization of K-Means for anomaly detection hinges critically upon the concept of distance. Anomalies are construed as data points that deviate significantly from the predominant cluster within a dataset. By leveraging the distance between each data point and its nearest cluster centroid, we can effectively pinpoint instances that lie substantially distant from the established clusters. A fundamental requirement of the K-Means algorithm is the pre-specification of the number of clusters, posing a primary challenge in its application, particularly in the realm of transactional monitoring.

The K-Means algorithm operates through several stages. Initially, the number of clusters  $K$  is specified. Next, centroids are either chosen sequentially or randomly from the initial dataset to form  $K$  clusters. Then, distances from each data point to the  $K$  centroids are computed. Subsequently, each centroid is recalculated based on the mean values of the data points assigned to it. Data points are then reassigned to clusters based on the smallest distance to centroids. Finally, steps 3 through 5 are iterated until there are no further changes in cluster assignments.

In this analysis, we employ the *KMeans* package, a tool that necessitates

the specification of parameter  $K$ , denoting the number of disjoint clusters  $C$ , each characterized by its centroid  $\mu_j$ —often referred to as the cluster means. Notably, these centroids are not typically data points from the set  $X$ , although they reside within the same feature space.

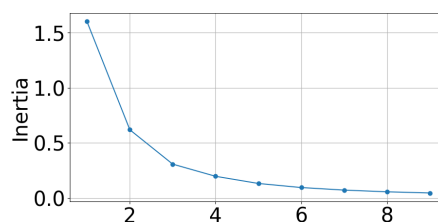
The objective of the K-Means algorithm centers on the minimization of inertia, a metric employed to determine the optimal number of clusters. Inertia quantifies the sum of squared distances between each data point  $X$  and its corresponding centroid  $\mu_j$  within a cluster. A desirable model exhibits both low inertia and a conservative number of clusters  $K$ . Nonetheless, a tradeoff exists, as increasing  $K$  tends to decrease inertia. To ascertain the optimal  $K$  for a given dataset, we apply the Elbow method, which identifies the point where the reduction in inertia begins to plateau.

In transaction monitoring, initially considering two clusters—“one for potentially fraudulent transactions and one for non-fraudulent ones”—seems intuitive. However, practical analysis may reveal that two clusters are inadequate. If the number of identified fraudulent transactions is too small, distinguishing them from non-fraudulent ones becomes challenging, requiring more clusters for better separation. Conversely, an excessive number of fraudulent transactions could overwhelm the algorithm, reducing its effectiveness.

Therefore, selecting the optimal number of clusters for transaction monitoring involves balancing these factors. It requires careful consideration of dataset characteristics, such as the prevalence of fraudulent transactions, to ensure effective anomaly detection without compromising accuracy.

The one-dimensional scenario begins with determining the optimal number of clusters for analysis. Figure 5.8 illustrates the inertia results, which guide the selection process. Utilizing the elbow method, the optimal number of clusters is identified at the point where improvements in the inertia value taper off. Based on this approach, the chosen  $K$  falls between 2 and 4. Given the objective of distinguishing outliers from non-outliers, we can determine the optimal  $K$  to be 2.

Figure 5.8: Inertia - Scenario 1



Source: Own elaboration



With  $K = 2$ , the model is fitted, and the results are shown in the confusion matrix in Figure 5.9. The K-Means algorithm correctly identified 1,619 fraudulent transactions and classified 1,212 as genuine, resulting in a sensitivity of 57.2%. This performance is significantly worse than the other models and could result in substantial regulatory penalties. However, the K-Means algorithm achieved a specificity of 99%, correctly classifying the majority of genuine transactions and falsely identifying only 222 as fraudulent. This indicates that while the model minimizes the time spent investigating false positives, it risks leaving a significant number of fraudulent transactions undetected, posing severe risks to the institution.

Figure 5.10 presents the ROC curve for the K-Means model, which has an AUC of 0.72, lower than the results achieved by the other models. Like the Isolation Forest, the K-Means model also experiences relatively long processing times, which can be problematic in a transactional monitoring context. Furthermore, the requirement to accurately determine the number of clusters contributes to its time consumption, making it the most time-consuming model among those evaluated.

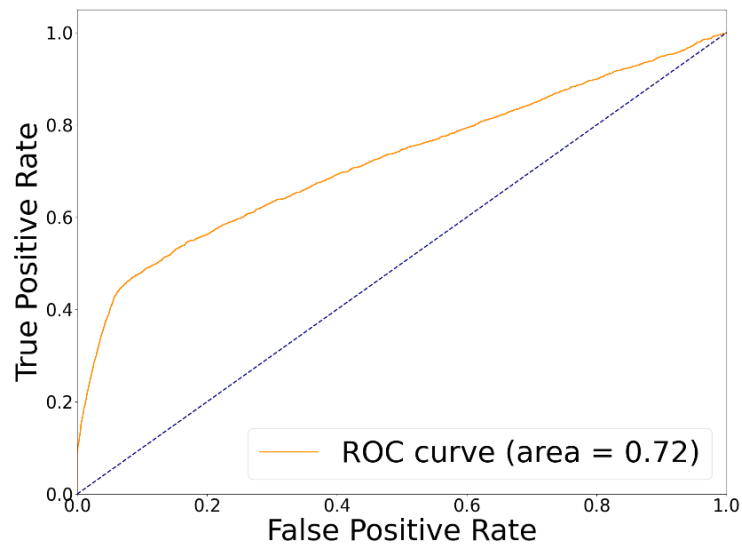
Figure 5.9: Confusion Matrix of K-Means - Scenario 1

A confusion matrix for the K-Means model. The vertical axis is labeled 'True label' with categories 'False' and 'True'. The horizontal axis is labeled 'Predicted label' with categories 'False' and 'True'. The matrix cells contain the following counts: True label False, Predicted label False: 22703; True label False, Predicted label True: 222; True label True, Predicted label False: 1212; True label True, Predicted label True: 1619.

True label	Predicted label	
	False	True
False	22703	222
True	1212	1619

Source: Own elaboration

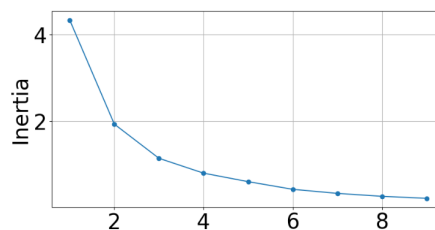
Figure 5.10: ROC Curve of K-Means - Scenario 1



Source: Own elaboration

In the two-dimensional scenario, the optimal number of clusters is again determined using the elbow method, as depicted in Figure 5.11. The chosen  $K$  remains at 2.

Figure 5.11: Inertia - Scenario 2

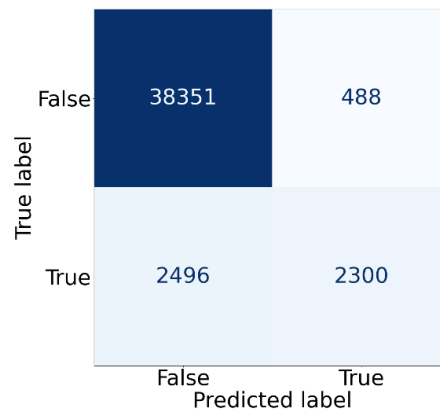


Source: Own elaboration

The results shown in Figure 5.12 indicate that the K-Means model maintained high specificity at 98.7%, outperforming the other models with only 488 genuine transactions falsely identified as fraudulent. However, sensitivity decreased to 48%, correctly identifying 2,300 fraudulent transactions and leaving 2,496 undetected. This trade-off is unfavorable for financial institutions, as more fraudulent transactions remain undetected, rendering this approach ineffective for transaction monitoring.

Figure 5.13 displays the ROC curve for Scenario 2, which has an AUC of 0.84. While this is lower compared to the other models, the difference is not substantial. However, the processing time increased to 7.12 seconds, which is

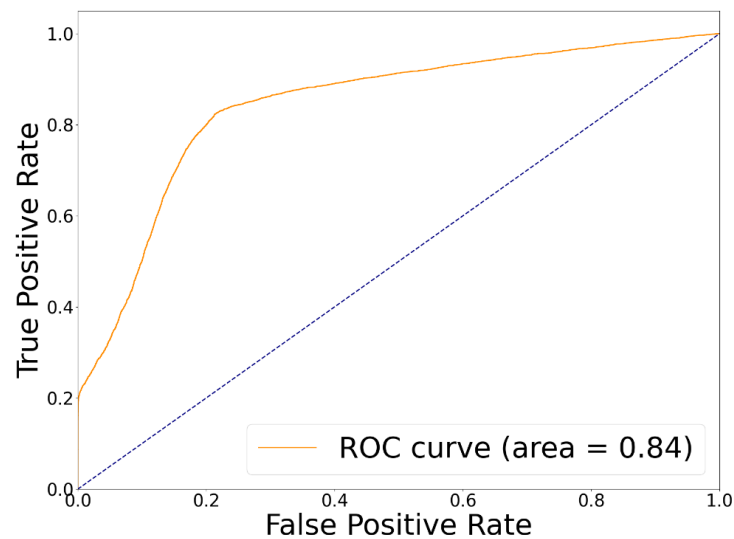
Figure 5.12: Confusion Matrix of K-Means - Scenario 2



Source: Own elaboration

notably longer than COPOD's 0.1 seconds. This indicates that the K-Means model is not ideal for this scenario.

Figure 5.13: ROC Curve of K-Means - Scenario 2

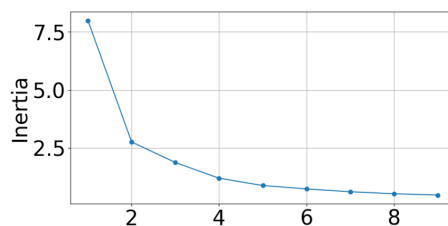


Source: Own elaboration

In the four-dimensional scenario, the inertia analysis shown in Figure 5.14 indicates that the optimal number of clusters can remain at 2.

The results presented in Figure 5.15 demonstrate the poorest performance of all models. The K-Means model correctly identified only 2 fraudulent transactions, leaving 3,804 fraudulent transactions undetected, with a specificity of only 0.001%. For genuine transactions, only 12,242 were correctly classified, while 27,166 were falsely identified as fraudulent. This would significantly

Figure 5.14: Inertia - Scenario 3



Source: Own elaboration

increase the workload of the investigation team, with almost none of the transactions leading to SARs.

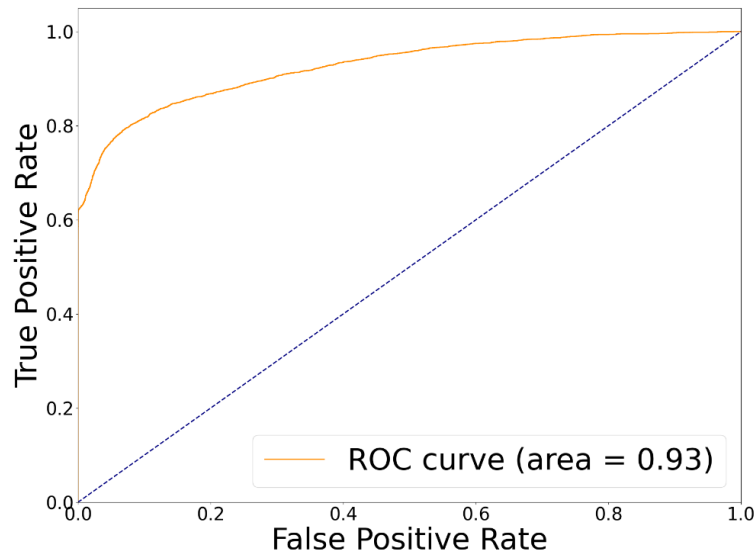
Figure 5.16 illustrates the ROC curve for Scenario 3, which has an AUC of 0.93. Although this is slightly lower than the results from the other models, the difference is not substantial. However, the AUC alone cannot compensate for the lower performance in specificity and sensitivity observed in previous results. Additionally, with a processing time of 4.34 seconds — the highest among all models — this further underscores the inefficiency of the K-Means model for this task.

Figure 5.15: Confusion Matrix of K-Means - Scenario 3

	False	True
False	12242	27166
True	3804	2
	False	True
	Predicted label	

Source: Own elaboration

Figure 5.16: ROC Curve of K-Means - Scenario 3

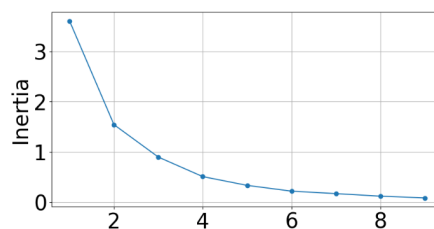


Source: Own elaboration

Moving to the real-world dataset, the first step across all scenarios was to determine the optimal number of clusters  $K$ . Surprisingly, scenarios 1 and 3 yielded unexpected results during the assessment of  $K$ , as depicted by their inertia in Figures 5.17 and 5.18. Initially, it appeared that, similar to the artificial data, the optimal number of clusters would vary between 2 and 4. However, upon applying the model to each variant, we found that the number of identified outliers is very low and did not change significantly with additional clusters.

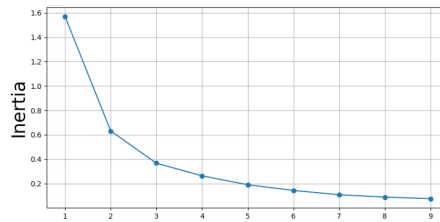
After experimenting with different  $K$  values, we decided to adopt  $K = 5$  across all variants and identify all the points in the smaller cluster as outliers. This decision was based on observing that increasing the number of clusters beyond 5 did not notably alter the detection of outliers, indicating diminishing returns in terms of identifying additional anomalies.

Figure 5.17: Inertia - Scenario 1



Source: Own elaboration

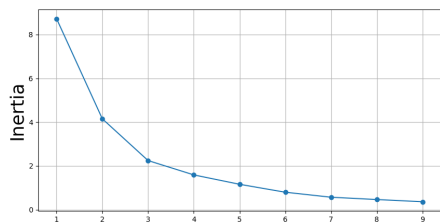
Figure 5.18: Inertia - Scenario 3



Source: Own elaboration

In contrast, scenarios 2 in two dimensions required a different approach. Here, experimenting with  $K = 2$  resulted in a substantial proportion of potential alerts being identified. Therefore, we retained  $K = 2$  for this scenario, as it provided a high-enough number of alerts without introducing excessive noise.

Figure 5.19: Inertia - Scenario 2



Source: Own elaboration

The results before final preprocessing are summarized in Table 5.3. In scenario 1, K-Means identified 3,143 alerts, representing 1.49% of the features triggering an alert, which aligns well with industry standards.

Scenario	Total features	Outliers	Outliers (%)	Time (s)
1	211,499	3,143	1.49%	1.5906
2	379,969	253,666	66.76%	1.0054
3	328,674	3,846	1.17%	3.1174

Table 5.3: Performance metrics of K-Means on real-world data

For multi-dimensional scenarios 2 and 3, detailed results are presented in Table 5.4. Notably, scenario 2 initially identified over 66% of features as outliers, which was reduced after preprocessing but still remains too high for practical use. Conversely, scenario 3 showed a very low proportion of outliers post-preprocessing (0.05%), suggesting that the analysis may have missed significant anomalies.

These findings underscore that K-Means does not appear to be a suitable option in any dimension for effectively identifying outliers in this real-world

Scenario	Total transactions	Outliers after applying the threshold	Outliers (%)
2	379,969	162,384	42.74%
3	328,674	151	0.05%

Table 5.4: Outliers detected by K-Means after applying the thresholds to scenarios

dataset. The algorithm’s inability to consistently provide meaningful and actionable alerts across scenarios raises concerns about its suitability for robust anomaly detection in financial monitoring and other high-stakes applications.

### 5.3 Copulas

The last chosen model for our analysis is COPOD, which we will implement using the *COPOD* package. COPOD is recognized for its efficiency, particularly in handling high-dimensional datasets. It follows a structured three-stage process designed to produce outlier scores for a given  $d$ -dimensional input dataset  $X = \{X_1, \dots, X_n\}$ . These outlier scores, denoted as  $O(X)$ , range between  $(0, \infty)$  and are used comparatively. It’s important to note that the score  $O(X_i)$  does not represent the probability of  $X_i$  being an outlier, but rather indicates how likely  $X_i$  is compared to other points in the dataset: the higher the score, the greater the likelihood of  $X_i$  being an outlier.

In the initial step, COPOD fits  $d$  empirical left-tail cumulative distribution functions (CDFs) and  $d$  empirical right-tail CDFs by considering the negative values of  $X$ . In the second step, it computes the empirical copula observations  $\hat{U}_{d,i} = \hat{F}_d(x_i)$  and  $\hat{V}_{d,i} = \hat{F}_d^-(x_i)$  for each  $X_i$ . Subsequently, it calculates the probability of observing a point as extreme or more extreme than  $x_i$  along each dimension. The outlier score is determined as the maximum of the negative logarithm of these probabilities from the left-tail empirical copula, right-tail empirical copula, and skewness-corrected empirical copula.

Intuitively, smaller tail probabilities correspond to larger negative logarithm values, leading to the identification of outliers based on low probabilities in either the left or right tail. This approach allows for flexible outlier assessment based on specified thresholds: higher thresholds focus on right-tail probabilities, while lower thresholds emphasize left-tail probabilities.

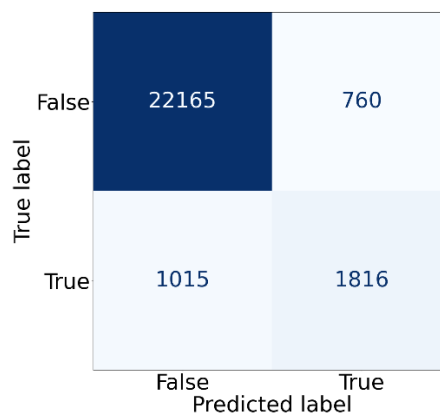
In addition to its comprehensive outlier detection methodology, COPOD stands out as a parameter-free model. This characteristic simplifies model fit-

ting, eliminating the need for parameter estimation and enhancing its practical usability.

The predictive performance of the COPOD algorithm for Scenario 1 is illustrated in Figure 5.20. The model accurately identified 22,165 genuine transactions, misclassifying only 760 as fraudulent, resulting in a specificity of 96.7%. On the other hand, COPOD correctly identified 1,816 fraudulent transactions, with 1,015 false negatives, yielding a sensitivity of 64.1%. While this performance is average among its peers, it still showcases the model's reliability in distinguishing genuine transactions from fraudulent ones.

The ROC curve for COPOD in Scenario 1, shown in Figure 5.21, exhibits an AUC of 0.82, closely matching the performance of the Isolation Forest. Notably, COPOD's processing time is significantly lower at only 0.02 seconds, outperforming the other models in terms of speed. This efficiency makes COPOD particularly attractive for real-time transaction monitoring where rapid processing is critical.

Figure 5.20: Confusion Matrix of COPOD - Scenario 1



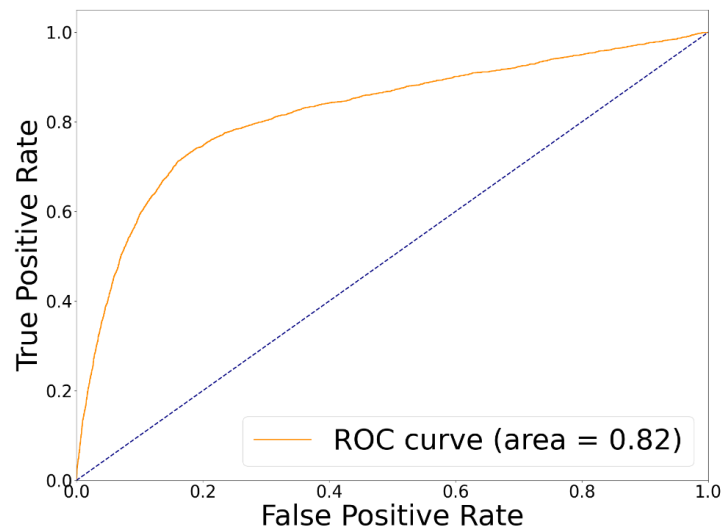
A confusion matrix for the COPOD model in Scenario 1. The matrix is a 2x2 grid. The vertical axis is labeled 'True label' with 'False' at the top and 'True' at the bottom. The horizontal axis is labeled 'Predicted label' with 'False' on the left and 'True' on the right. The top-left cell (True False, Predicted False) is dark blue and contains the value 22165. The top-right cell (True False, Predicted True) is light blue and contains the value 760. The bottom-left cell (True True, Predicted False) is light blue and contains the value 1015. The bottom-right cell (True True, Predicted True) is light blue and contains the value 1816.

	False	True
False	22165	760
True	1015	1816

Source: Own elaboration



Figure 5.21: ROC Curve of COPOD - Scenario 1

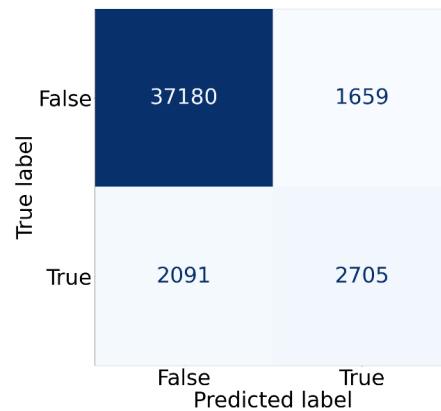


*Source:* Own elaboration

In the two-dimensional scenario, COPOD's results remain consistent with its performance in the one-dimensional analysis. The model correctly identified 37,180 genuine transactions, with 1,659 false positives, resulting in a specificity of 95.7%. However, sensitivity dropped slightly to 56.4%, with 2,705 correctly identified fraudulent transactions and 2,091 undetected. Despite this slight decline, COPOD still outperforms its peers overall in this scenario.

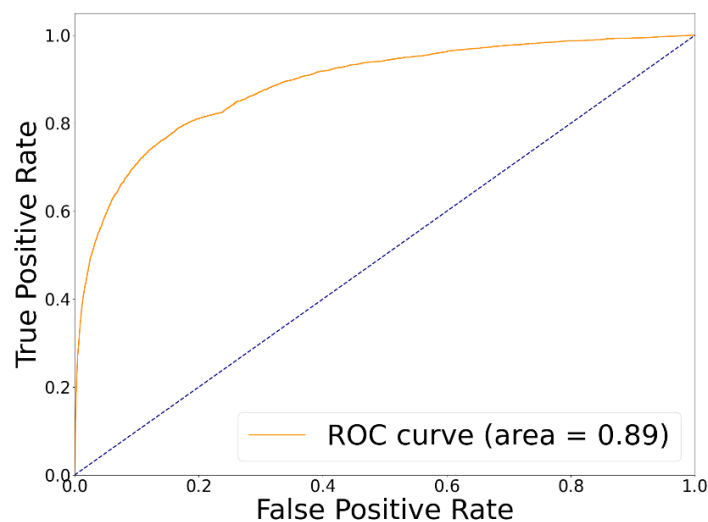
Figure 5.23 presents the ROC curve for COPOD in Scenario 2, which has an AUC of 0.89 — slightly better than K-Means and on par with Isolation Forest. Notably, COPOD achieves this performance while maintaining a processing time of just 0.1 seconds, in contrast to the 6.3 seconds required by the Isolation Forest. This highlights COPOD's superior efficiency and suitability for real-time applications, making it the preferred choice in two-dimensional scenarios.

Figure 5.22: Confusion Matrix of COPOD - Scenario 2



Source: Own elaboration

Figure 5.23: ROC Curve of COPOD - Scenario 2



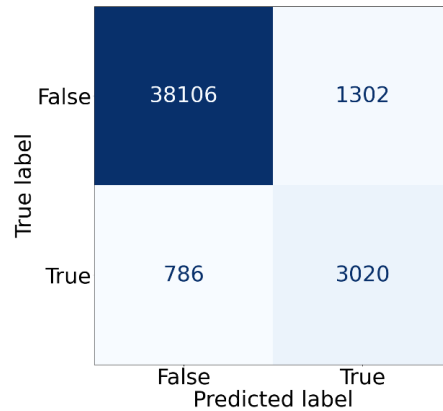
Source: Own elaboration

In the four-dimensional scenario, COPOD's performance improves further. The model achieved a specificity of 96.9%, accurately classifying 38,106 transactions as genuine, with only 1,302 false positives. Sensitivity also increased to 79.3%, correctly identifying 3,020 fraudulent transactions and leaving 786 undetected. These results highlight COPOD's enhanced capability to handle higher-dimensional data effectively.

The ROC curve for Scenario 3, shown in Figure 5.25, reveals an impressive AUC of 0.98, comparable to Isolation Forest's performance. Crucially, COPOD maintains a processing time of only 1.1 seconds, significantly faster than both K-Means and Isolation Forest. This rapid processing capability, combined with

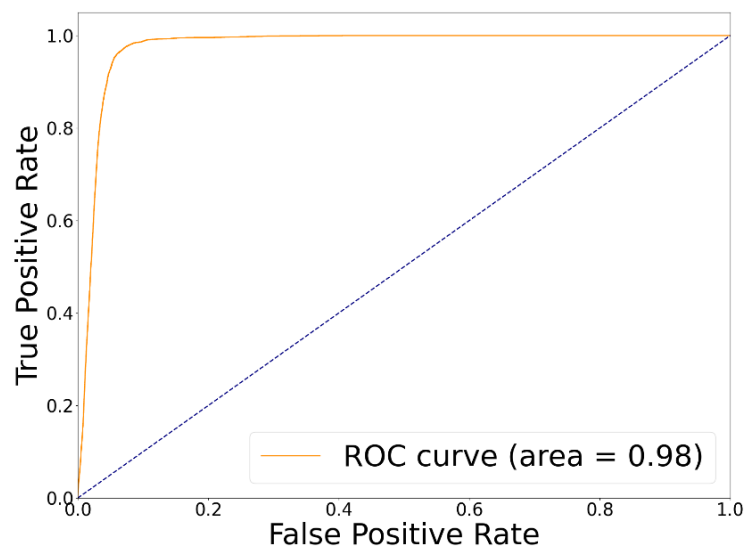
robust performance, positions COPOD as a highly efficient model for outlier detection in higher-dimensional datasets.

Figure 5.24: Confusion Matrix of COPOD - Scenario 3



Source: Own elaboration

Figure 5.25: ROC Curve of COPOD - Scenario 3



Source: Own elaboration

Moving to the real-world dataset, we applied the COPOD algorithm to each scenario. The results are shown in Table 5.5.

COPOD identified 21,150 outliers out of 211,499 total features, constituting 10.00% of the dataset. Given the high volume of potential alerts in this scenario, careful consideration is required to select a threshold that corresponds to a more manageable number of alerts.

With a larger dataset of 379,969 features in Scenario 2, COPOD identified 37,996 outliers. After preprocessing, the number of outliers was reduced to

Scenario	Total features	Outliers	Outliers (%)	Time (s)
1	211,499	21,150	10.00%	1.5906
2	379,969	37,996	10.00%	1.0054
3	328,674	32,868	10.00%	3.1174

Table 5.5: Performance metrics of COPOD on real-world data

6,636, which is 1.75% of the total features. This reduction after preprocessing aligns closely with the results from Isolation Forest, demonstrating effective outlier identification suitable for threshold determination.

In post-processing of Scenario 3, as detailed in Table 5.6, the number of outliers decreased from 32,868 to 10,613, representing 3.23% of the total. Given the double-counting issue in this scenario, this level of outliers remains appropriate and provides a balanced approach for transaction monitoring and threshold determination.

Scenario	Total transactions	Outliers after applying the threshold	Outliers (%)
2	379,969	6,636	1.75%
3	328,674	10,613	3.23%

Table 5.6: Outliers detected by COPOD after applying the thresholds to scenarios

Overall, COPOD exhibits consistent performance across different scenarios in identifying outliers. This makes COPOD comparable to Isolation Forest in terms of performance, allowing institutions to choose between them based on specific considerations such as risk appetite and processing time. Notably, COPOD demonstrates significantly shorter processing times compared to Isolation Forest, which is advantageous when evaluating larger numbers of thresholds across diverse customer portfolios.

# Chapter 6

## Limitations

This thesis aims to explore various outlier detection models applicable to the optimization of transaction monitoring (TM) systems. In the realm of TM, a prevalent challenge lies in assessing the efficacy of adopted approaches. Typically, financial institutions employ outlier detection models on transaction datasets to determine optimal threshold values for different scenarios and customer segments, often based on targeted alert volumes aligned with the institution's risk appetite. Striking a balance between alert volumes is critical, as it mitigates the risk of underreporting suspicious behavior to regulators while avoiding excessive alert volumes that may strain investigative resources or necessitate additional staffing.

### *Contamination Assessment*

Determining the appropriate number of alerts identified by each scenario depends on the desired level of contamination our model should tolerate. It is challenging to definitively state the proportion of transactions that are inherently suspicious. This proportion is dynamic and can vary across different seasons, economic conditions, and other factors. Given that we aim to detect transactions primarily associated with fraudulent activities — such as human trafficking, terrorism financing, and tax fraud — it remains elusive to pinpoint the exact correct proportion. Consequently, when assessing the optimal alert thresholds, we must carefully consider the risk appetite specific to the institution.

To initiate the optimization process, we must define risk appetite parameters. These parameters consist of weighted factors that assign importance to false positives (non-suspicious alerts), false negatives (missed Suspicious Activ-

ity Reports or SARs), and true positives (investigated SARs). The risk appetite parameters reflect the institution's risk-based approach and can be expressed as the ratio of the cost of false negatives to the cost of false positives. These parameters will differ for each institution.

Moreover, it is essential to recognize that risk appetite may vary across different scenarios and contexts, necessitating a nuanced approach to optimizing transaction monitoring systems.

#### *Model Evaluation*

Evaluation of outlier detection models relies heavily on historical alert data labels, typically categorized as false positives (FP), worthy alerts (WA), and suspicious-activity reports (SAR). FP alerts are those deemed false upon initial investigation, while WA alerts undergo further scrutiny but are ultimately dismissed. SAR alerts are escalated for in-depth investigation and reporting to regulators. However, this approach presents several challenges.

Firstly, when setting thresholds for new scenarios, the absence of historical labels disables the possible evaluation. Secondly, with the scenarios previously used, reliance on above-the-line results may overlook anomalies falling below previous thresholds, exacerbating the lack of labeled data, as in the first case. Thirdly, even if a scenario addresses these issues, the accuracy of dataset labels remains uncertain. Label accuracy is contingent upon the effectiveness of the institution's investigation processes, susceptibility to human error, and the ability to identify suspicious activity accurately. Additionally, feedback from regulators on filed SARs is often unavailable, further clouding label validity. Consequently, reliance on dataset labels hinges on assumptions rather than confirmed accuracy.

Evaluating model functionality based on data labels alone is fraught with complexity and often yields inconclusive results. Instead, reliance on industry expertise and expert knowledge becomes paramount. Nonetheless, outlier detection within TM remains crucial, enabling institutions to monitor, report, and potentially mitigate criminal activities such as human trafficking, terrorist financing, and warfare, underscoring the significance of a well-structured monitoring process.

#### *Scenario Limitations*

Transaction monitoring systems are only as effective as the scenarios they employ. The selection of scenarios plays a pivotal role in identifying suspicious

activities. Financial institutions typically choose scenarios based on various criteria, including risk assessments, regulatory requirements, historical data, trends, typologies, and red flags. However, the evolving creativity of criminals, coupled with advancements in AI, poses a challenge. Scenarios may become overly robust, exceeding the transaction monitoring system's capacity to handle them effectively. For instance, consider Scenario 3, which we utilized during our analysis. This scenario focuses on monitoring the volume and frequency of deposits, particularly rapid increases. To cover all possible situations, it operates with an n-day lookback on a daily basis. Notably, the time windows for the 30-day period and the 6-month period are not mutually exclusive. Additionally, customer activity during the last 30 days contributes to monitoring low activity over the 12-month period. Unfortunately, this approach introduces an issue of double counting. The challenge is most pronounced during the first half of each month, potentially limiting the scenario's ability to generate alerts. As it stands, the scenario performs optimally toward the end of the month.

#### *Disclosure of Confidentiality*

It is important to note that the code and data used in this thesis are proprietary and confidential, as they are part of ongoing production processes at the authors organization. Due to the proprietary nature of these resources, they can not be shared or made publicly available. This limitation is necessary to protect sensitive information and intellectual property. While this restriction may affect the ability to fully reproduce the results, it is in line with standard industry practices for safeguarding confidential and proprietary information. The author appreciates the understanding of these constraints and their impact on the availability of detailed resources.

# Chapter 7

## Model Comparison

The performance evaluation of various models in detecting fraudulent transactions reveals noteworthy insights. Overall, the models demonstrated satisfactory performance, with the exception of K-Means, which exhibited significant underperformance in both artificial and real-world data scenarios, particularly in higher dimensions.

Table 7.1 presents the results of model application on artificial datasets, serving as an initial evaluation before deployment on real-world transactions. Across different dimensions, all models encountered challenges in achieving satisfactory sensitivity levels. Notably, Isolation Forest and Copulas showed varying performance across dimensions, whereas K-Means consistently lagged behind, especially in sensitivity metrics.

In one dimension, K-Means exhibited the highest specificity (99%), indicating its strength in identifying genuine transactions accurately but at the cost of lower sensitivity compared to Isolation Forest and Copulas (specifically, 89.1% for Isolation Forest). Isolation Forest and Copulas demonstrated competitive sensitivity (72.1% and 64.1%, respectively) and AUC (0.86 and 0.82, respectively), highlighting their effectiveness in detecting fraudulent transactions while minimizing false positives. Copulas notably distinguished itself with a processing time of 0.02 seconds, significantly outperforming the other models in efficiency.

In two dimensions, Isolation Forest and Copulas maintained competitive specificity (95.5% and 95.7%, respectively) but experienced decreased sensitivity (54.8% and 56.4%, respectively) and AUC scores (0.89 both). K-Means continued to excel in specificity (98.7%) but struggled with sensitivity (48%) and AUC (0.84), alongside higher processing times, indicating limitations in



complex datasets.

In higher dimensions, both Isolation Forest and Copulas exhibited enhanced performance, maintaining competitive metrics across all dimensions. The Isolation Forest demonstrated an impressive AUC (0.98), with robust sensitivity (81.5%) and specificity (98.7%), though it came with increased processing times. Copulas also achieved a high AUC (0.98), with competitive sensitivity (79.3%) and specificity (96.7%), while showcasing superior processing time efficiency. This underscores Copulas' capability in handling complex dependencies effectively. Conversely, K-Means achieved a decent AUC (0.93), but this result does not compensate for its severely poor specificity (0.001%), highlighting its inefficiency in higher-dimensional datasets.

Isolation Forest consistently demonstrated strong overall performance across dimensions, albeit with increasing processing times in higher dimensions. K-Means, effective in lower dimensions, struggled with accuracy and efficiency as dimensionality increased. Copulas consistently offered efficient processing and competitive performance metrics.

The performance of all models varied notably with dimensionality, with Isolation Forest and Copulas maintaining effectiveness, while K-Means exhibited differing levels of resilience. Copulas consistently demonstrated more stable performance across dimensions.

When selecting a model, it is crucial to consider performance metrics alongside practical constraints such as processing time, especially in scenarios requiring real-time responses for high-frequency transaction monitoring.

The results, from the application of these models to real-world data, are presented in Table 7.2. K-Means exhibited inconsistent performance across all dimensions, which raises significant concerns about its reliability for practical use. The results from K-Means did not align with industry standards, making it challenging to establish appropriate thresholds for fraud detection. This inconsistency across dimensions further undermines its suitability for robust and reliable transaction monitoring systems, where accuracy and consistency are paramount.

Conversely, both Isolation Forest and Copulas continued to demonstrate strong performance and practical applicability. Their results consistently met industry benchmarks across various dimensions, reinforcing their reliability in detecting fraudulent transactions effectively. Isolation Forest's ability to maintain competitive performance metrics across different dimensionalities, despite

Scenario 1				
Model	Sensitivity	Specificity	AUC	Time (s)
Isolation Forest	0.721	0.891	0.86	2.2900
K-Means	0.572	0.99	0.72	2.9443
Copulas	0.641	0.967	0.82	0.0194
Scenario 2				
Model	Sensitivity	Specificity	AUC	Time (s)
Isolation Forest	0.548	0.955	0.89	6.318
K-Means	0.48	0.987	0.84	7.122
Copulas	0.564	0.957	0.89	0.1086
Scenario 3				
Model	Sensitivity	Specificity	AUC	Time (s)
Isolation Forest	0.815	0.969	0.98	4.161
K-Means	0.311	0.001	0.93	4.34
Copulas	0.793	0.967	0.98	0.1097

Table 7.1: Quantitative performance metrics of models across all scenarios

increasing processing times in higher dimensions, underscores its suitability for a wide range of applications.

Copulas, on the other hand, not only delivered competitive performance metrics but also exhibited exceptional efficiency with processing times, particularly noteworthy in higher dimensions where computational efficiency becomes critical. This efficiency makes Copulas particularly advantageous.

The robust performance of Isolation Forest and Copulas in real-world scenarios underscores their viability as leading models for fraud detection systems. Their ability to consistently deliver results that align with industry standards across dimensions reinforces their practical utility and reliability in detecting fraudulent activities while minimizing false positives.

Scenario 1				
Model	Potential Number of Alerts	Outliers	Outlier Percentage	Time (s)
Isolation Forest	211,499	29,000	13.71%	24.1952
K-Means	211,499	3,143	1.49%	1.5906
Copulas	211,499	21,150	10.00%	0.7054
Scenario 2				
Model	Potential Number of Alerts	Outliers	Outlier Percentage	Time (s)
Isolation Forest	379,969	5,227	1.38%	31.3524
K-Means	379,969	162,384	42.74%	1.0054
Copulas	379,969	6,636	1.75%	0.8298
Scenario 3				
Model	Potential Number of Alerts	Outliers	Outlier Percentage	Time (s)
Isolation Forest	328,674	14,530	4.42%	28.0342
K-Means	328,674	151	0.05%	3.1174
Copulas	328,674	10,613	3.23%	0.9054

Table 7.2: Qualitative performance metrics of models across all scenarios

# Chapter 8

## Conclusion

This thesis conducts a thorough evaluation of three outlier detection methods — Isolation Forest, K-Means, and a Copula-based approach — specifically in the realm of fraudulent transaction detection. We detailed the rationale behind selecting these models and explained how their results can be optimized for transactional monitoring. The discussion also covered the limitations inherent to these methods and the complex nature of transactional monitoring, emphasizing the critical role of expert knowledge in leveraging these models effectively. One of the takeaways is that an effective customer segmentation model is crucial for maximizing the performance of outlier detection methods.

Our experimental results and comparative analysis provide significant insights into each method's strengths and weaknesses across various scenarios.

Isolation Forest consistently delivered strong performance across different metrics, including sensitivity, specificity, and AUC, in both artificial and real-world datasets. Although processing times increased with higher dimensionality, its effectiveness in identifying fraudulent transactions makes it a valuable tool for applications with moderate to high-dimensional data.

The Copula-based approach excelled in efficiency and processing time, particularly in higher-dimensional contexts. Its capability to model complex multivariate dependencies and achieve competitive sensitivity and specificity highlights its suitability for environments where both computational efficiency and accuracy are essential. COPOD demonstrated reliable performance and fast processing, making it particularly advantageous for fraud detection systems.

Conversely, K-Means faced significant challenges in higher-dimensional settings, with reduced sensitivity compromising its reliability as an outlier detection method for complex datasets, despite maintaining a strong AUC. Al-

though it performed well in terms of specificity with lower-dimensional data, its effectiveness diminished significantly as dimensionality increased. This decline in performance makes K-Means less suitable for robust fraud detection compared to Isolation Forest and Copulas, which both excel in handling higher-dimensional data.

In summary, Isolation Forest and Copulas emerged as the most effective outlier detection methods in the scenarios tested. Isolation Forest offers a balanced approach between accuracy and computational demands, while Copulas provide exceptional efficiency and performance, especially in high-dimensional contexts where modeling complex dependencies is crucial.

When selecting an outlier detection model for practical use, factors such as dimensionality, processing time, and accuracy must be carefully evaluated. For low-dimensional data, simpler models may suffice, as the added complexity of advanced methods might not yield better results. Isolation Forest and Copulas stand out as leading methods for fraud detection systems, aligning well with industry standards and practical needs.

This thesis contributes to the field by comparing these models within the specific context of fraud detection and providing insights into real-world applicability and threshold settings.

Future research could investigate integrating these models or developing hybrid approaches to harness their individual strengths. Additionally, exploring other methods and analyzing larger, more diverse datasets could yield deeper insights and advance anomaly detection techniques.

# Bibliography

- AGGARWAL, C. C. & P. S. YU (2001): “Outlier detection for high dimensional data.” *ACM SIGMOD Record* **30(2)**: pp. 37–46.
- AHMED, M., J. HU, X. HU, X. LIU, J. KIM, D. DEY, & K. POCHIRAJU (2016): “Anomaly detection for temporal data: A survey.” *ACM Computing Surveys (CSUR)* **50(5)**: pp. 1–38.
- ALPAYDIN, E. (2020): *Introduction to Machine Learning*. Cambridge, MA: MIT Press.
- BELLMAN, R. E. (1961): *Adaptive Control Processes*. Princeton, NJ: Princeton University Press.
- BISHOP, C. M. (2006): *Pattern Recognition and Machine Learning*. New York: Springer.
- BORA, D. J. & D. A. K. GUPTA (2014): “Effect of Different Distance Measures on the Performance of K Means Algorithm: An Experimental Study in Matlab.” *International Journal of Computer Science and Information Technologies* **6**.
- CHANDOLA, V., A. BANERJEE, & V. KUMAR (2009): “Anomaly detection: A survey.” *ACM Computing Surveys (CSUR)* **41(3)**: pp. 1–58.
- CHEN, Y. & W. WU (2018): “Isolation Forest as an Alternative Data-Driven Mineral Prospectivity Mapping Method with a Higher Data-Processing Efficiency.” *Natural Resources Research* **28**: pp. 1–16.
- ELKAN, C. (2001): “The foundations of cost-sensitive learning.” *Department of Computer Science and Engineering 0114, University of California, San Diego, La Jolla, California* .
- FAESEL, K. (2022): *Finding Ghosts in Your Data*. Apress Berkeley, CA.

- FAYZRAKHMANOV, R., A. KULIKOV, & P. REPP (2020): “The difference between precision-recall and roc curves for evaluating the performance of credit card fraud detection models.” *Information Technologies and Computer-Based System Department, Perm National Research Polytechnic University* .
- FUJIMAKI, R., T. YAIRI, & K. MACHIDA (2005): “An Approach to Spacecraft Anomaly Detection Problem Using Kernel Feature Space.” *Research Center for Advanced Science and Technology* pp. 401–410.
- GENEST, C. & A. FAVRE (2007): “Everything you always wanted to know about copula modeling but were afraid to ask.” *Journal of Hydrologic Engineering* **12(4)**: pp. 347–368.
- GOODFELLOW, I., Y. BENGIO, & A. COURVILLE (2016): *Deep Learning*. Cambridge, MA: MIT Press.
- GUSTAVSON, H. (2019): *Clustering Based Outlier Detection for Improved Situation Awareness within Air Trac Control*. Ph.D. thesis, KTH ROYAL INSTITUTE OF TECHNOLOGY, SCHOOL OF ENGINEERING SCIENCES.
- HARIRI, S., M. C. KIND, & R. J. BRUNNER (2018): “Extended Isolation Forest.” *Instrumentation and Methods for Astrophysics* **6**.
- HAWKINS, D. M. (1980): *Identification of Outliers*. Springer.
- HENNIG, C., T. F. LIAO, & M. MEILA (2015): “Cluster analysis and data mining: An introduction.” *Wiley Encyclopedia of Operations Research and Management Science* pp. 1–15.
- HODGE, V. J. & J. AUSTIN (2004): “A survey of outlier detection methodologies.” *Artificial Intelligence Review* **22(2)**: pp. 85–126.
- IGLEWICZ, B. & D. C. HOAGLIN (1993): *Robust Statistics: Theory and Methods*. New York: John Wiley & Sons.
- JAIN, A. K., J. I. GHOSH, & N. K. RATHA (2020): “Data clustering: A review.” *ACM Computing Surveys (CSUR)* **52(3)**: pp. 1–35.
- JHAVERI, R. H., A. REVATHI, K. RAMANA, R. RAUT, & R. K. DHANARAJ (2022): “Review on Machine Learning Strategies for Real-World Engineering Applications.” *Mobile Information Systems* **2022**: p. 26.

- JOE, H. (2014): *Dependence Modeling with Copulas*. CRC Press, 1st edition.
- JORDAN, M. I. & T. M. MITCHELL (2015): “Machine learning: Trends, perspectives, and prospects.” *Science, New Series* **349**: pp. 255–260.
- KAUFMAN, L. & P. J. ROUSSEEUW (2009): *Finding Groups in Data: An Introduction to Cluster Analysis*. Wiley, 2nd edition.
- KOHOUTOVÁ, P. (2023): “Detecting financial frauds: the role of anomaly detection in aml pipeline.”
- LIU, F. T., K. M. TING, & Z.-H. ZHOU (2008): “Isolation forest.” *IEEE Transactions on Knowledge and Data Engineering* **23(6)**: pp. 993–1006.
- LIU, F. T., K. M. TING, & Z. H. ZHOU (2012): “Isolation-Based Anomaly Detection.” *ACM Transactions on Knowledge Discovery from Data* **6**: pp. 1–39.
- LIU, F. T., K. M. TING, & Z. H. ZHOU (2015): “Isolation Forest.” *IEEE International Conference on Data Mining* pp. 413–422.
- MITCHELL, T. M. (1997): *Machine Learning*. McGraw-Hill: Science/Engineering/Math.
- MONSON, D. & S. VANDERMARK (2013): *Suspicious Activity Reports and analytics: Staying ahead of the compliance curve*. B.m.: Deloitte.
- MURPHY, K. P. (2012a): *Machine Learning: A Probabilistic Perspective*. Cambridge, MA: MIT Press.
- MURPHY, K. P. (2012b): *Machine Learning: A Probabilistic Perspective (Adaptive Computation and Machine Learning series)*. The MIT Press.
- NELSEN, R. B. (2006): *An Introduction to Copulas*. Springer, 2nd edition.
- NGAI, E. W., L. XIU, & K. CHAU (2011): “The application of data mining techniques in financial fraud detection: A review.” *Decision Support Systems* **50(3)**: pp. 559–569.
- PATTON, A. J. (2012): “A review of copulas and their applications in finance.” *Journal of Financial Econometrics* **10(4)**: pp. 621–657.



- PHAM, D. T. & G. A. RUZ (2009): “Unsupervised training of Bayesian networks for Data Clustering.” *Proceeding: Mathematical, Physical and Engineering Sciences* **465**: pp. 2927–2948.
- SHI, T. & S. HORVATH (2006): “Unsupervised Learning With Random Forest Predictors.” *Journal of Computational and Graphical Statistics* **15**: pp. 118–138.
- SKLAR, A. (1959): “Fonctions de répartition ‘a n dimensions et leurs marges.” *Publ. Inst. Statist. Univ. Paris* **8**: pp. 229–231.
- STOJANOVIC, J., M. IVANOVIC, & D. MILINKOVIC (2022): “Efficient k-means clustering with improved centroid initialization.” *Pattern Recognition Letters* **152**: pp. 192–200.
- WANG, L., X. WANG, & Z. ZHANG (2018): “A review on clustering algorithms.” *Knowledge-Based Systems* **151**: pp. 18–36.
- XIA, Y., Y. LI, & X. ZHENG (2020): “Real-time anomaly detection for safety-critical systems.” *Journal of Real-Time Systems* **56(2)**: pp. 245–267.
- XU, R. (2005): “Survey of Clustering Algorithms.” *IEEE Transactions on Neural Networks* **16**.
- ZHANG, H., Y. ZHAO, & W. DING (2015): “Isolation forest with improved isolation criteria.” *Journal of Computer Science and Technology* **30(2)**: pp. 269–285.
- ZHENG LI, Yue Zhao, N. B. C. I. (2020): “COPOD: Copula-Based Outlier Detection.” *Industrial Conference on Data Mining* .