universität wien

MASTERARBEIT | MASTER'S THESIS

Titel | Title

Anomaly Detection in Financial Services: A comparison study of different methods used for anomaly detection

verfasst von | submitted by

Enis Pinari

angestrebter akademischer Grad | in partial fulfilment of the requirements for the degree of

Master of Science (MSc)

Wien | Vienna, 2024

Studienkennzahl lt. Studienblatt | Degree programme code as it appears on the student record sheet:	UA 066 977
Studienrichtung lt. Studienblatt | Degree programme as it appears on the student record sheet:	Masterstudium Business Analytics
Betreut von | Supervisor:	Ass.-Prof. Dott.ssa Dott.ssa.mag. Yllka Velaj PhD

Acknowledgements

There are a lot of people I want thank for their support throughout the time it took to write this paper.

First and foremost, I extend my deepest gratitude to my supervisor Ass.-Prof. Yllka Velaj, for her guidance and patience over the past few months. You were always ready to help and carefully listen to my ideas. I am truly grateful that you agreed to be my supervisor. Your input and motivation pushed me to deliver a piece of work I am proud of.

An important support pillar has been my family. Always, willing to listen, often pretending as if what I was saying made any sense. I am fortunate to have them in my life. If there was a person who always believed and cheered for me since day one, it would be my grandma. All this work, and every other achievement, is and will always be dedicated to her loving memory. This past year, I was blessed to experience the joy of becoming an uncle, and without a doubt, I dedicate this work to my little nephew, as well. I hope one day, if he gets the chance to read this paper, he feels a little bit proud of his uncle.

I also want to thank my friends. You were the ones I would complain to after long day of work. I admit, not everyone fully understood what anomaly detection is and why it interested me, but some good jokes came out of it. To those who did understand, thank you for the ideas that helped me deliver the best work possible.

Last but not least, I want to thank the young man that, three years ago, decided to leave his home and move to a new country. It was terrifying, having no idea how things would have turned out, but I think they turned out great. These past three years have been an incredible journey, filled with ups and downs, amazing new people, new memories, and a lot of personal growth. Looking back, I believe that the terrified young man would be proud of where I got and the person I have become.

Abstract

Anomaly Detection, also known as outlier detection, is the process of identifying data points that do not follow establish patterns. It has a long history of application, primarily in statistics, nonetheless with passing time its application has expanded across various domains. An anomaly event could indicate a potential fraud, network issues, diseases or faulty equipment. One of the biggest challenges, when dealing with anomalies, is defining what is a normal point and what characteristics it possess. Researchers have conducted numerous studies evaluating a vast number of anomaly detection models across various domains. These studies have highlighted the need for anomaly detection, especially, in the financial sector.

The developments in the financial industry have closely connected it to technology and data with the purpose of improving decision-making process and increasing efficiency. Though, the technological advancement have increased the volume of data the business is dealing with, it has created a growing need for anomaly detection. In finance, the role of anomaly detection is to identify unusual patters that could lead to fraud, identity theft, money laundering, etc.

The aim of this paper, is to compare the performance of unsupervised and supervised machine learning models in anomaly detection using financial data. It extends existing benchmark research by evaluating both classical and advanced models. Furthermore, it emphasizes the importance of anomaly detection in the financial industry, providing insights into model effectiveness and offering guidance for finance professionals in selecting appropriate models based the characteristics of their datasets or tasks at hand.

To achieve our goal, we conduct two main experiments. Experiment 1, uses four real world financial datasets to compare the performance of the machine learning algorithms, using ROC AUC, F1 Score and Runtime as the main performance metrics. For experiment 2, we use synthetic datasets to test the scalability of the chosen models. According to the findings of experiments, XGBoost, CatBoost and Random Forest Classifiers are the best model, in terms of providing good accuracy scores and reasonable computational runtime. Other good performing models are KNN, Isolation Forest and DBSCAN.

Keywords: anomaly detection, machine learning, models, finance, metrics, performance

Kurzfassung

Anomalieerkennung, auch als Ausreißererkennung bekannt, ist der Prozess der Identifizierung von Datenpunkten, die keinem festgelegten Muster folgen. Sie wird seit langem angewendet, vor allem in der Statistik, doch im Laufe der Zeit hat sich ihre Anwendung auf verschiedene Bereiche ausgeweitet. Ein Anomalieereignis könnte auf einen möglichen Betrug, Netzwerkprobleme, Krankheiten oder fehlerhafte Geräte hinweisen. Eine der größten Herausforderungen beim Umgang mit Anomalien besteht darin, zu definieren, was ein normaler Punkt ist und welche Eigenschaften er besitzt. Forscher haben zahlreiche Studien durchgeführt, in denen eine große Anzahl von Anomalieerkennungsmodellen in verschiedenen Bereichen bewertet wurden. Diese Studien haben die Notwendigkeit der Anomalieerkennung insbesondere im Finanzsektor hervorgehoben.

Die Entwicklungen in der Finanzbranche haben sie eng mit Technologie und Daten verknüpft, um den Entscheidungsprozess zu verbessern und die Effizienz zu steigern. Obwohl der technologische Fortschritt das Datenvolumen erhöht hat, mit dem das Unternehmen zu tun hat, ist ein wachsender Bedarf an Anomalieerkennung entstanden. Im Finanzwesen besteht die Aufgabe der Anomalieerkennung darin, ungewöhnliche Muster zu identifizieren, die zu Betrug, Identitätsdiebstahl, Geldwäsche usw. führen könnten.

Ziel dieses Dokuments ist es, die Leistung von unbeaufsichtigten und überwachten Modellen des maschinellen Lernens bei der Anomalieerkennung anhand von Finanzdaten zu vergleichen. Es erweitert die bestehende Benchmarkforschung, indem sowohl klassische als auch fortgeschrittene Modelle bewertet werden. Darüber hinaus betont es die Bedeutung der Anomalieerkennung in der Finanzbranche, bietet Einblicke in die Modellwirksamkeit und bietet Finanzfachleuten Anleitung bei der Auswahl geeigneter Modelle basierend auf den Eigenschaften ihrer Datensätze oder anstehenden Aufgaben.

Um unser Ziel zu erreichen, führen wir zwei Hauptexperimente durch. Experiment 1 verwendet vier reale Finanzdatensätze, um die Leistung der Algorithmen des maschinellen Lernens zu vergleichen, wobei ROC AUC, F1-Score und Laufzeit als Leistungsmetriken verwendet werden. Für Experiment 2 verwenden wir synthetische Datensätze, um die Skalierbarkeit des ausgewählten Modells zu testen. Den Ergebnissen der Experimente zufolge sind XGBoost, CatBoost und Random Forest Classifiers das beste Modell, was gute Genauigkeitswerte und eine angemessene Rechenlaufzeit betrifft. Andere Modelle mit guter Leistung sind KNN, Isolation Forest und DBSCAN.

Schlüsselwörter: Anomalieerkennung, Maschinelles Lernen, Modelle, Finanzen, Leistung, Metriken

Contents

List of Tables

List of Figures

1. Introduction

Anomaly detection has been a long-standing research topic. There are several approaches developed and used to detect anomalies across various domains, including statistical methods and machine learning algorithms depending on the tasks.

An anomaly is defined as an observation which deviates significantly from the majority of observations, implying that they were generated by a separate generating process [20].

Anomaly detection, also referred to as outlier detection, was often seen as a step in data processing phase, which often meant eliminating the outliers from the dataset to improve the performance of the models [8]. However, new studies consider anomaly detection crucial since it could provide valuable insights. Thus, the subject of anomalies has generated interest among researches in different disciplines. They often identify critical events, for example an increase in demand or a possible failure, which can trigger response actions [3].

Anomaly detection is used on various fields, such as in military surveillance, cybersecurity, fraud detection, healthcare, etc., [13]. An anomaly in traffic patterns of a computer network might indicate an unauthorized transferring of sensitive data to a target destination. In military monitoring, anomalies in satellite images can reveal elements such as enemy troop movements [24]. On the other hand, in healthcare it can check patient records to detect new symptoms or diseases [40]. Likewise, the analysis of credit card usage is another area of application for anomaly detection that can assist consumers and financial institutions to avoid cases of fraud [55].

Researchers argue that there are several reasons that make it difficult to deal with anomaly detection. One key challenge is the issue of how to define what is *normal* observation, as the definition is dependent on the dataset and can vary from one dataset to another. Additionally, the normal observations could change over time. As an example, the purchasing patterns of a customer evolves, which leads to it mistakenly taken as an anomalous behavior [3]. Furthermore, there is also the problem with the lack of labeled anomaly data to train the models. The majority of existing datasets, have an unbalanced spread, containing a large number of normal data points and just a few anomalies or labeled as such [8].

1.1. Problem Statement

Nowadays, the financial services industry is strongly connected to technology and information to improve efficiency and the decision-making process. Hence, the importance of methods helping in anomaly detection is significantly increasing.

The digitization of financial transaction has grown exponentially the volume of data

generated. Anomaly alerts are needed because of the availability of this amount of data. The anomalies can result in illegal activities like fraud, identity theft, risk, account takeover, money laundering, etc., that leads to serious issues [3].

Thus, the utilization of technology in financial services has improve efficiency, but at the same time introduced threats. Therefore, it is necessary to have anomaly detection models to guarantee compliance, integrity as well as prevent losses [32].

Due to the increase in demand, more and more supervised and unsupervised models are being adopted for anomaly detection in financial use cases. The characteristics of the financial industry, namely high fluctuation rates along with various types of financial instruments and transactions, require specific models ranging from simple to highly complex and advanced ones. However, with many models available, the problem of selecting the most suitable one arises, particularly in terms of efficiency and runtime.

1.2. Significance of the Study

In this study, we focus on comparing multiple machine learning models for anomaly detection using financial datasets. This study uses established benchmark research as foundation for our experiments and extends further by conducting a comparative analysis that covers classical and newer machine learning models. While previous work has primarily concentrated on traditional methods, our work also explores the use of more advanced models for anomaly detection.

The study emphasize the importance of anomaly detection in the financial industry, by using financial datasets to assess the performance of the models. Thus, it contributes to a better understanding of their effectiveness in a financial domain application. The findings can guide finance professionals to implement the most appropriate models for anomaly detection based on their dataset characteristics.

1.3. Research Questions

The objective of this study is to compare the performance of unsupervised and supervised machine learning models in anomaly detection using financial data. The following research questions were developed to support this objective and will be discussed in the following chapters.

- *What are the advantages and disadvantages of the anomaly detection models?*
- *How do these models perform in terms of efficiency and accuracy when applied to financial data?*

1.4. Thesis Structure

The thesis is structured as follows:

Chapter 1 provides a general overview of anomaly detection while describing the main problem at hand and the significance of the study.

Chapter 2 introduces related work in the field of anomaly detection, the results of which will serve as the basis for our experiments. Additionally, it provides definitions for certain ideas and concepts crucial for anomaly detection and financial services.

Chapter 3 details the methodology used for conducting the experiments, including an explanation of the models selected, datasets, data pre-processing steps and metrics used to evaluate the results.

Chapter 4 focuses on the experiments and provides a detail explanation of the findings.

Chapter 5 summarizes the findings of the experiments and provides insight into future work.

2. Literature Review

To investigate the various uses of anomaly detection, a number of studies have been carried out in different domains. Several studies aim to present the idea of anomaly detection in various domains, whereas others compare the different techniques that can be used for anomaly detection.

2.1. Anomaly Detection

It is important to note that the literature uses the terms "outlier" and "anomaly" interchangeably. Hawkins (1980), one of the first researchers to provide an extensive study on anomalies, describes an anomaly as an observation that deviates so much from other observations as to arouse suspicion that it was generated by a different mechanism [28]. This definition forms the basis of understanding anomalies on various applications [33]. Other notable definitions include, Johnson (1992), who interprets anomalies as observations in a dataset that appears to be inconsistent with the remainder of the dataset [36]. Likewise, Barnett and Lewis (1994), define an anomaly as an observation that appears to deviate markedly from the members of the sample in which it occurs [5]. These definitions are applied from a statistical point of view, where anomalies that vary from the normal mechanism of producing data, frequently provide information regarding abnormal characteristics affecting the generating mechanism [2].

Nonetheless, Boukerche et al. (2020) argue that whether anomalies are triggered by a different mechanism is not easily determined. In statistics and machine learning, anomalies are data points that cause problems when fitting the desired model. In these circumstances, the anomalies are either eliminated or other robust models are employed to reduce their impact [9].

Researchers have conducted numerous studies evaluating a vast number of models, creating comparison benchmarks for determining the best models for anomaly detection. Hodge and Austin (2004) were among the first who presented an extensive study on anomaly detection models. The research analyzes statistical methods, machine learning and ensemble methods used for anomaly detection. They classified the models according to their application and characteristics. Furthermore, the researchers emphasizes the challenges of anomaly detection in various types of data [34]. Chandola et al. (2009), in their survey, extends the work done by Hodge and Austin (2004). They concentrated on anomaly detection models that were not previously considered. By focusing on the real-life applications and advancements in the field, they aimed to expand the understanding of anomaly detection beyond the existing work [14].

In their survey on unsupervised anomaly detection models for high-dimensional nu-

merical data, Zimek et al. (2012) addresses the challenges posed by the "curse of dimensionality". Additionally, they discuss algorithms specifically designed to handle irrelevant features or attributes for high-dimensional data. The paper aims to improve the efficiency of anomaly detection for these data types by comparing different models [64]. Gupta et al. (2014) develop a survey for anomaly detection models using time-series data. The aim is to investigate the challenges brought on by temporal data in anomaly detection, as well as to explore the developments in computational capabilities to facilitate the analysis of temporal data in anomaly detection scenarios [26].

Han et al. (2022) conducted an evaluation of unsupervised, semi-supervised and supervised anomaly detection models using various benchmark datasets. The study addresses the need for a standardized benchmark in anomaly detection research. The main finding of the paper emphasizes the lack of a statistically superior unsupervised learning model. Additionally, it highlights the importance of selecting a model depending on the specific requirements of the application [27].

2.1.1. Types of Anomalies

To improve the selection of models suitable for the task of anomaly detection, there is a need classify the anomalies. Chandola et al. (2009) in [14] offered a classification of anomalies into three distinct categories based on the nature, behavior, and context.

1. *Point Anomalies:* If a single data point is considered anomalous in reference to the rest of the dataset, then it is a point anomaly. This is referred to as the simplest and most common form.

2. *Contextual Anomalies:* If a data point behaves anomalously in a specific context, but not in other cases, it is considered contextual anomaly. The context is dependent on the dataset and must be specified during the problem statement.

3. *Collective Anomalies:* If a collection of data points are anomalous with respect to the rest of the dataset, then they are collective anomalies. Individually the data points might not be anomalies, however their behavior as a collection is anomalous.

2.1.2. Anomaly Detection Applications

As previously mentioned, anomaly detection has a variety of applications across different domain. Hodge and Austin (2009) in [34] provided an extensive list of several domains where anomaly detection can be employed, as seen in Table 2.1.

Table 2.1.: Anomaly Detection applications in different domains

Domain	**Anomaly Detection application**
Fraud detection	Detect fraudulent transactions or fraudulent applications for credit cards.
Loan applications	Detect fraudulent applications or problematic customers
Network performance	Detect network issues or potential cyberattacks.
Medical Research	Detect cancerous cells, monitor heart rate or identify new structures in molecules.
Industrial Systems	Detect fault in machinery or products during manufacturing.
Text Novelties	Detect news articles, topic detection and identifying under/over-performing commodities for traders.
Data Mining	Detect unexpected entries, mislabelled data or errors in the dataset.

Anomalies can occur as a results of human or mechanism errors, and natural deviations in the population, among other factors. The approach in dealing with the anomalies is dependent on the application task [34].

In their study, He et al. (2016) investigated the log-based anomaly detection, by evaluating the performance of six models. The focus was the precision and effectiveness analysis of the models using two production logs dataset [29]. Another researcher, Zuo (2017) examined anomaly detection in geo-chemical data processing. The study compared three popular models used in this domain: fractal/multi-fractal models, compositional data analysis and machine learning models, focusing primarily on the later [65].

Sodermal et al. (2012) discussed anomaly detection in automated surveillance. The authors explored models used across several previous research studies, in order to provide insights into their effectiveness and applicability in automated surveillance application [56]. Furthermore, Ibindunmyoe et al. (2015) focused on the application of anomaly detection and bottleneck identification for computing systems. The study assessed the core elements of the task and classified existing applications [35]. On the other hand, Tsai et al. (2009) studied anomaly detection on intrusion detection, specifically machine learning models. The paper offered an overview of models used for solving intrusion detection challenges designed between 2000 and 2007 [60]. Buczak and Guven (2016) described machine learning and data mining models for anomaly detection in cyber-intrusion detection, emphasizing the challenges in cybersecurity [12]. Moreover, Bhuyan et al. (2014) presented a survey on anomaly detection for network monitoring. The research revolved around identifying the most common types of network attacks. They made comparisons of different models with regard to their effectiveness in responding to network intrusion and the weaknesses surrounding the approaches [6].

2.2. Financial Sector

For every country, the economy's ability to grow and develop depends on how well different sectors operate. The financial sector is a significant player in the economic development. Its impact on the economy is subtle yet very complex. A developed financial sector leads to an efficient use of financial products and services. It enhances the movement of funds, which enables investment and consumption to take place [59].

The financial sector includes the design and implementation of financial decisions and relationships. In most countries, these financial decisions and relationships between an individual and a firm involve financial institutions such as banks, insurance companies, capital markets, and any other similar institution that provides financial services or products.

One of the basic functions of the financial sector is encouraging savings and distributing loans throughout different times. It provides payment services, products, like hedging, to help individuals and services protect themselves against uncertainties. A well-functioning financial sector lowers the costs and risks of producing and exchanging goods and service, which significantly increases living standards [30].

The advancement on the sector have affected the financial markets to move at a high speed, where a lot of things are happening simultaneously and several roles are in charge of monitoring these activities. Since every financial institution handles large volumes of data, data is an important asset for these organizations. As financial transactions become more digitized and complex, the volume of data grows exponentially [37].

2.2.1. Anomaly Detection in Finance

Numerous studies have highlighted the need for anomaly detection models in the financial sector. The survey developed by Delmaire et al. (2009) focuses on the anomaly detection models, specifically on credit card fraud detection models. The research delivered an understanding of different types of credit card fraud and the most efficient models employed to detect them [18]. Furthermore, Ngai et al. (2011) in their research reviewed literature on financial fraud detection using data mining techniques. The researchers concentrated on the application of these techniques in credit card fraud, money laundering and security fraud [44]. Furthermore, Ryman-Tubb and Krause (2011) proposed a neural network model for extracting rules to detect credit card fraud. The research aimed to develop a system that by using neural networks, it would automatically extract rules to recognize patterns of fraud. Thus, improving the accuracy of the system [49].

Using various models, Zareapoor et al. (2012) was able to deliver an updated survey on detecting fraudulent activities in credit card transactions [62]. Likewise, Richhariya and Singh (2012) conducted a similar survey comparing various models for detecting fraudulent activities. The goal was to offer an understanding of anomaly detection techniques in the financial sector, emphasizing the challenges, advancements, and best practices [48]. On the contrary, Falaki et al.(2015) suggested a probabilistic model to predict the likelihood of a transaction being fraudulent or not based on several features and patterns [51].

Moreover, Adewumi and Akinyelu (2018) applied machine learning and nature-inspired

models for credit card fraud detection. Something their paper sought to achieve was to assess the efficiency in improving the accuracy and reliability of the models [1]. Pourhabibi et al.(2020) systematically reviewed graph-based anomaly detection models for fraud detection. The research examined various methods to analyze the connection patterns in communication networks to detect suspicious behavior in regard to bank fraud, insurance fraud and anti-money laundering [45]. Additionally, Hilal and Yawney (2022) published a comprehensive survey on financial fraud fraud detection, emphasizing novel models and recent advances on the field [31].

3. Methodology

In this chapter, we provide a description of our experiments. It includes a detailed explanation of the models employed, datasets used to perform the experiments, the evaluation metrics, as well as the methodology adopted for the experiments.

3.1. Machine Learning Models

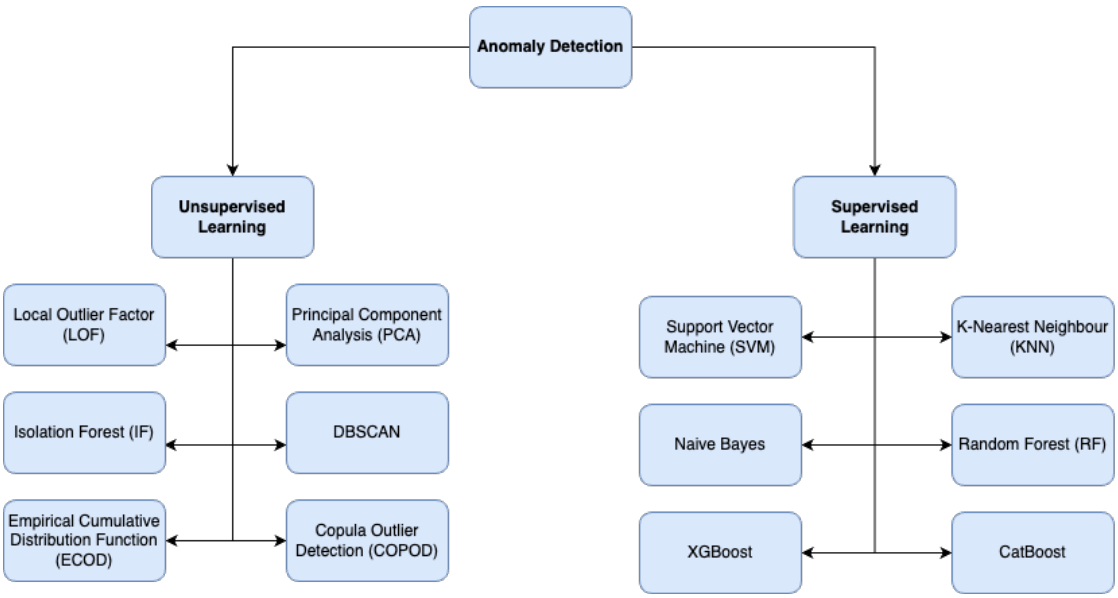


Figure 3.1.: Machine Learning Models

3.1.1. Unsupervised Learning Models

Unsupervised learning models analyze and identify patterns in unlabeled data, usually needed for tasks like clustering or anomaly detection. These models are feed only by unlabeled training data, thus making it difficult to quantitatively evaluate their performance [43]. Unsupervised learning model are generally used when you need to cluster a group of elements based on their similarity. Another area of application for unsupervised models, is the automatic labeling of data [7]. For anomaly detection task, unsupervised learning models try to identify anomalies without prior knowledge, as the labeled anomaly points are missing. The models look for data points that deviate

significantly from the majority of the data otherwise considered as normal points. A detailed analysis of the unsupervised models used for anomaly detection, shown in Figure 3.1, is presented on the following section.

Local Outlier Factor (LOF)

Breunig et al.(2000) developed Local Outlier Factor (LOF) as an unsupervised machine learning model designed to find outliers on a multidimensional dataset. LOF evaluates how much a data point is isolated from its neighbors by assessing the *density* of other points in the neighborhood, which is simply a count of other neighbor points and their closeness.

For every individual data point, the model determines the local density, if this density is lower compared to the densities of neighbor points, then the point is an outlier. A score close to 1, indicates that the data point is similar to its neighbors, hence is not an outlier, whereas a score higher than 1 suggests the point is an outlier.

There is one parameter in the model named *MinPts*, which represents how many neighbors must be included in the calculation. The choice of this parameter can significantly influence the results of the model [11].

Principal Component Analysis (PCA)

Shyu et al.(2003) introduced an unsupervised machine learning model called Principal Component Analysis (PCA), which aims to reduce the dimensionality of datasets while keeping as much information as possible. PCA transforms a large set of correlated variables into a smaller set of uncorrelated variables, making it easier to visualize data and helps with data analysis. Such transformation is carried out with the help of *principal components*, which represent linear combinations of the original variables in the dataset.

They are orders so that the first principal component captures the highest variance, the second one captures the second highest variance and so on. The principal components are derived from the eigenvalues and eigenvectors of the datasets' covariance matrix. Eigenvalues represent the amount of the variance captured by each principal component, whereas eigenvectors give the direction of the components.

For anomaly detection tasks, it uses a subset of principal components to reconstruct the original dataset. Anomalies are identified based on the differences between the reconstructed dataset and the original one [54].

Isolation Forest

Isolation Forest is part of unsupervised machine learning that focuses on identifying anomalies in a dataset by isolating them. In their paper, Liu et al.(2008) describe the idea that anomalies are rare and distinct from the majority of the data, hence it is easier to isolate them from the rest of the dataset. Based on this idea, the Isolation Forest model is developed. The model builds a group of binary trees, known as *Isolation Trees*. To build each tree, a feature is chosen at random, followed by a random split value for

the feature, which is used to split the data. This process continues iteratively until each data point is isolated.

An important parameter of the model is the *path length* from the root of the tree to a certain data point. Each data point has an anomaly score assigned based on the average length across all trees. For anomaly points the path length is shorter since these points are isolated faster, hence they receive higher score [41].

DBSCAN

Ester et al. (1996) introduced Density-Based Spatial Clustering of Application with Noise, or commonly known as DBSCAN, a clustering algorithm that groups points based on their density. In DBSCAN clusters are regions in the data space where the density of points is higher than the surrounding areas. This indicates that a cluster is formed when there are sufficient points near one another. The model has two main parameters that need to be taken into consideration. *MinPts* which define the minimum number of points a cluster must have. *Epsilon (ϵ)* is the radius or how large a neighborhood around a point.

According to the mentioned parameters, there are three types of points we can locate on a cluster. *Core Points* are points that have a minimum number of neighboring points within a specific radius. *Border Points* are points that is not a core point, however it is within the neighborhood of a core point. *Noise Points* are neither core or border points. They are consider outliers or anomalies.

The model can identify cluster of arbitrary shapes. It starts with an arbitrary point and checks all the points in the neighborhood of epsilon. If the point is a core point, a cluster is formed. The model iteratively goes through all the points within neighborhood reach, and expands the clusters, until all points are handled [21].

Empirical Cumulative Distribution Function (ECOD)

Empirical Cumulative Distribution based Outlier Detection model, or ECOD, is an unsupervised machine learning model designed to detect outliers. According to Li et al. (2023), the model computes the empirical cumulative distribution functions or ECDFs, for each dimension of the dataset. The ECDFs provide an understanding of how data points are distributed without making strong assumptions regarding the overall dataset distribution.

Then, for each dataset, the ECOD model calculates the tail probabilities across each dimension to assess how extreme the data point is. In the end, the model aggregates the tail probabilities to generate a single outlier score for each data point. The higher the score the greater the likelihood the data point is an outlier [39].

Copula Outlier Detection (COPOD)

Li et al. (2020) proposed the Copula Based Outlier Detection (COPOD) as an outlier detection model that operates by using copulas. Copulas are mathematical functions that

allow the modelling of joint distributions for multiple variables while separating their individual marginal distributions. This makes it possible for the model to understand the dependencies between various data dimensions.

COPOD builds an empirical copula from the dataset, which helps on determining how the data points are related to each other. The model, then, calculates the tail probabilities of each data point. This indicates how extreme is the data point in regard to the rest of the dataset. By evaluating both the left and right tail probabilities, it can identify outliers that might be unusually high or low. The process is robust to the distribution of data, since it incorporates a correction for skewness [38].

3.1.2. Supervised Learning Models

Supervised learning models make predictions on labeled data. The models are feed by a set of training data, which are labeled points, and makes prediction on the test data, which are unseen points. Supervised learning can be used for classification, regression or ranking tasks [43]. Supervised learning models are trained on labeled datasets, where each data point is clearly labeled as anomaly or not. The models learns the characteristics of the normal data points and based on the knowledge predicts the labels on unseen data points. The supervised learning models, presented in Figure 3.1, are discussed below.

Support Vector Machine (SVM)

Cortes and Vapnik (1995) presented the Support Vector Machine (SVM) model as a supervised machine learning model used primarily for classification tasks. The model aims to determine the optimal boundary or *hyperplane*, that separates classes in a dataset. A hyperplane is an *n-1 dimensional* flat affine subspace that acts as a boundary, distinguishing between classes with the largest margin.

The margin is defined as the distance of the hyperplane from the nearest point belonging to any of the two classes. These points are also called support vectors, and only they affect the location of the hyperplane and its movement. The basic idea of the SVM model is to find the hyperplane that results to maximum margin so as to increase the likelihood of correct classification of the unseen data. For anomaly detection, SVM finds a decision boundary that captures the normal points, while the data points that lie far from this boundary are labeled as anomalies.

For real world scenarios, where the datasets are not perfectly separable, SVM model introduces *soft margins*. Soft margin permits some misclassifications, in order to strike a balance between maximizing the margin and minimizing the classification error [16].

K-Nearest Neighbors (KNN)

K-Nearest Neighbors is a non-parametric model that operates on the basis that similar data points are likely to belong on the same class. As reported by Cover and Hart (1967), the k-NN model classifies new data points based on the class of their closest *'k'* neighbors.

The distance between the data point and its neighbors is calculated using metrics, with Euclidean distance as one of the most popular.

The model is not built traditionally during the training phase, but instead stores all the training dataset used to make classification of new data points. When a new data point is to be classified as an anomaly or not, the model searches the 'k' nearest neighbors and then makes the classification by the majority of 'k' neighbors. If the distance to the k-nearest neighbors is significantly larger, then the data point is labeled as anomaly, since it is presumed to belong in an isolated region.

It is crucial to set an optimal value for 'k', as it significantly effects the model's output. Small value of k makes the model prone to the noise present in the data, but a large value of k will lead to overlooking the local patterns and smooths the decision boundaries [17].

Naive Bayes

The Naive Bayes model predicts the class of a given data point based on the Bayes' theorem. Domingos and Pazzani (1997) describe the model as a probabilistic classification model, as it calculates the probability that a data point belongs to each class and assigns the point to the class with the highest probability.

The model relies on the *"naive"* assumption that given the class label, all features are independent of each other. Thus, the presence or the lack of one feature has no effect on the presence or the lack of other features. The Bayes' theorem as mathematical formula used for calculation is:

$$P(C_i \mid A) = \frac{P(C_i) \cdot P(A \mid C_i)}{P(A)} \tag{3.1}$$

where:

- $P(C_i \mid A)$ is the probability of class C_i given the feature A
- $P(C_i)$ is the prior probability of class C_i
- $P(A \mid C_i)$ is the probability of the feature A given the class C_i
- $P(A)$ is the total probability of the feature A

For anomaly detection tasks, the model is trained on labeled data. The prior probability $P(C_i)$ is calculated for each class along with probability $P(A \mid C_i)$. When a new data point has to be classified, Bayes' Theorem is used to calculates the posterior probabilities for the normal and anomaly class. If a data point has a low probability of belonging to the normal class, then it is labeled as anomaly. [19].

Random Forest Classifier

Breiman (2001) developed an ensemble learning model used for classification tasks called Random Forest Classifier. The model goes through various steps that include creating multiple trees and aggregating their predictions.

To create the decision trees, a random sample of the training data is chosen with replacement for each tree. This follows the *bootstrapping* technique, where some of the data points might be repeated while other are not included. Additionally, for the construction of the trees a random sample of features is selected at each node to establish the best split. The randomness allows for the trees to be diverse and have no correlation among them.

The random features and the bootstrapped samples are used to construct each decision tree. The tree grows up to its maximum depth without pruning, enabling it to recognize complex patterns in the data. After the trees are built, the Random Forest Classifier casts votes for the class label on each tree in order to make a prediction. The class with the majority of votes is chosen as the predicted class [10]. A data point that constantly receives minority votes or is flagged as an anomaly across multiple trees is labeled as an anomaly point.

XGBoost

XGBoost, or eXtreme Gradient Boosting, is a machine learning technique optimized for efficiency and performance in predicting tasks. In their paper, Chen and Guestrin (2016) establish XGBoost as an application of *gradient boosting*, a method that combines the predictions of several weak learners, usually decision trees, to produce a robust predictive model.

First, an initial prediction is made, which based on the task could be the target feature's average value (for regression tasks) or a constant values (for classification tasks). Then, XGBoost builds a sequence of decision trees, each designed to improve on the errors of the previous one. This is done sequentially. For each iteration, the model calculates the difference between the actual values and the predicted values. The new decision tree is trained based on these differences, which essentially learns to improve the errors.

The *loss function* is minimized using the *gradient descent*. The model computes the gradient of the loss function in regards to the predictions to determine the direction in which the predictions need to be adjusted to minimize errors. After all the trees are built, the final prediction is made by aggregating all the predictions from each tree. The model assigns anomaly scores based on how well a data point fits the trained decision boundaries. Anomalies have low probability or high loss of being classified as normal points.

Furthermore, the model prevents overfitting through the regularization terms in the objective function that penalize overly complex trees in favor of simpler ones [15].

CatBoost

Prokhorenkove et al.(2018) presented CatBoost, a gradient boosting algorithm specialized to better handle categorical features and minimize prediction bias. The foundation of the model are the principals of boosting, where multiple weak learners are combined to create a robust predictive model.

CatBoost develops models in a sequential manner, where each new model improves the errors of the previous one. The base learners are called *oblivious decision trees*. By

applying the same splitting criteria on each level, these trees minimize overfitting and speed up the prediction process. In anomaly detection, CatBoost builds a series of decision tress that separate normal point from anomalies. The data points that are far from the decision boundaries are classified as anomalies.

The most notable property of CatBoost, is its ability to handle categorical features without requiring a lot of preprocessing. It uses a method known as *ordered boosting*, which includes permutating the data to prevent target leakage. This way the model learn form the data in a way that accurately represents real-world scenarios. In addition, the model keeps a set of supporting models that predict using random permutation of the training data. This method mitigates prediction shifts caused by structure and the processing of the data. On top of that, CatBoost incorporates a Bayesian bootstrap process for subsampling the dataset at each iteration. Therefore, it further reduces prediction bias [46].

3.2. Datasets

3.2.1. Dataset 1: Credit Card Transaction

This dataset contains information about the credit transactions of customers. It provides records of transactions including transaction times, amounts, and details of the merchant. The dataset has *over 1.290.000 records and 24 features*. The dataset contains the following features:

- Unnamed:0 : row ID for each data record
- trans_date_trans_time: the timestamp of the transaction
- cc_num: number of the credit card used for the transaction
- merchant: the name of the merchant where the transaction took place
- category: category of the transacation (food, clothes, etc.)
- amt: amount of the transaction
- first: cardholder's first name
- last: cardholder's last name
- gender: cardholder's gender
- street: cardholder's street adress
- city: cardholder's city
- zip: cardholder's zip code
- lat: geographical coordination of the transaction (latitude)

- long: geographical coordination of the transaction (longitude)
- city_pop: population of the city where the transaction happened
- job: cardholder's occupation
- dob: cardholder's date of birth
- trans_num: unique transaction number
- unix_time: unix timestamp of the transaction
- merch_lat: geographical coordination of the merchant (latitude)
- merch_long: geographical coordination of the merchant (longitude)
- merch_zip: zip code of the merchant
- is_fraud: class label

The dataset can be found in Kaggle, under `https://www.kaggle.com/datasets/priyamchoksi/credit-card-transactions-dataset/data`

3.2.2. Dataset 2: Fraudulent Transaction Data

The dataset describes fraudulent transactions of a financial company with the goal of developing an action plan. The personal information of the customers is hidden here as well. The dataset has *over 6.300.000 records and 10 features*. A short description of the features, is as follows:

- step: unit of time
- type: transaction category
- amount: amount of transcation
- nameOrig: ID of the customer who initiated the transaction
- oldbalanceOrg: initial balance of the customer before the transaction
- newbalanceOrg: new balance of the customer after the transaction
- nameDest: ID of the transaction's recipient
- oldbalanceDest: initial balance of the recipient before the transaction
- newbalanceDest: new balance of the recipient after the transaction
- isFraud: class label

The dataset can be found in Kaggle, under `https://www.kaggle.com/datasets/chitwanmanchanda/fraudulent-transactions-data`

3.2.3. Dataset 3: Bank Payments

The dataset mimics the payments information of a banking institution in Spain. It contains no personal information [42]. Additionally, details regarding fraudulent transactions are present in the dataset. The dataset has *over 590.000 records and 10 features.* The feature description is:

- step: unit of time
- customer: customer ID
- age: the age of the customer
- gender: the gender of the customer
- zipcodeOri: customer's zip code
- merchant: the merchant's ID
- zipMerchant: merchant's zip code
- category: type of transaction
- amount: transaction's amount
- fraud: class label

The dataset can be found in Kaggle as *'bs140513_032310.csv'*, `https://www.kaggle.com/datasets/ealaxi/banksim1`

3.2.4. Dataset 4: Metaverse Financial Transaction Dataset

This dataset provides information about blockchain financial transactions and is aimed at developing anomaly detection models. It includes different types of transactions, user behaviors and risk profiles. The dataset has *over 78.000 records and 14 features.* A description of the features is presented below:

- timestamp: date and time of the transaction
- hour_of_day: hour when the transaction happened
- sending_address: sender's blockchain address
- receiving_address: recepient's blockchain address
- amount: transacation's amount
- transaction_type: category of transaction
- location_region: continent where the transaction happened

- ip_prefix: the prefix of the IP address
- login_frequency: user's login count
- session_duration: time spent per session
- purchase_behavior: buyer's behavior pattern
- age_group: user's activity level
- risk_score: transaction risk rating
- anomaly: class label

The dataset can be found in Kaggle, under `https://www.kaggle.com/datasets/faizaniftikharjanjua/metaverse-financial-transactions-dataset`

3.3. Hardware & Software Specifications

3.3.1. Hardware

The experiments are performed on a MacBook Air laptop with the following specifications 3.1:

Table 3.1.: Hardware Specifications

Artifact	**Specs**
CPU Model name	Apple M1 with 8-core CPU
RAM	8 GB
GPU	Integrated 7-core GPU

3.3.2. Software

Python 3.10 was used as the main programming language to run the models and perform the evaluations. For data processing and preparation, we used **Pandas** library, while **Matplotlib** was employed for data visualization. The machine learning models were built using the **Scikit-learn** and **PyOD** [63] libraries. Additionally, the parameters for each model were optimized with the **Hyperopt** library.

3.4. Experiment Setup

There are two main experiments that we execute to evaluate the performance of the machine learning models for anomaly detection. First, we evaluate the same models using four real world financial datasets. Next, we test the scalability of the models using synthetic datasets of various sizes.

3.4.1. Experiment 1: Real-world Datasets

The first experiment was conducted using four real-world datasets from the financial sector. All the financial datasets were sourced from Kaggle website. During the data preparation phase, the datasets went through common steps of data cleaning, checking for missing values, performing the necessary data transformations, standardization, and feature engineering.

3.4.2. Experiment 2: Scalability Test

Scalability testing is a type of non-functional testing that assess the ability of an application, network, or system to adapt to changes caused by increased user demand without violating predetermined objectives [23]. The goal is to validate that resources are being balanced appropriately by adding new resources and performing tests to determine the effect on response time and the application itself [25].

For our experiment, we conducted scalability testing by evaluating the performance of the selected machine learning models on synthetic datasets. The synthetic datasets were generated using the *make_classification* function of the Scikit-learn library. Each dataset has *15 features and 2 possible classes* for classification. The datasets vary on size starting with *50.000, 150.000, 350.000, 550.000, 750.000,* and *1.000.000* data records.

3.4.3. Evaluation Metrics

The performance evaluation for each model is done using the following metrics: *ROC AUS Score, F1 Score* and *Runtime*. Generally, these metrics are used to evaluate the performance of supervised machine learning models, where the *true labels* need to be feed to the model. However, to have a fair comparison between all models we have adopted the same metrics for the unsupervised learning models, as well. Thus, all the datasets chosen have a label column, which is omitted when running the unsupervised models and used only to calculate the performance metrics, namely *ROC AUC Score* and *F1 Score*. To further strengthen the evaluation results of the unsupervised models, we use metrics such as *Precision, Recall* and *Accuracy*.

ROC-AUC Score

The ROC AUC Score, which stands for *Area Under the Receiver Operating Characteristic Curve*, represents a single value that summarizes the performance of a binary classifier across all classification thresholds. Fawcett (2006) demonstrated that the score measures the capacity of a classifier to distinguish between the positive and negative instances. Generally, the ROC AUC score ranges from 0 to 1. A value of 0.5 signifies that the model has no discriminative power, while a value of 1.0 means a perfect discrimination between the positive and negative instances.

After the model generates the prediction scores, the data points are sorted in a descending order based on the scores. The sorting gives the opportunity to assess the

model performance at various threshold levels. Next, for each threshold the *true positive rate* (TPR) and *false positive rate* (FPR) are calculated.

$$\text{TPR} = \frac{\text{TP}}{\text{TP} + \text{FN}} \quad \text{and} \quad \text{FPR} = \frac{\text{FP}}{\text{FP} + \text{TN}} \tag{3.2}$$

where:

- TP are the true positives
- TN are the true negatives
- FP are the false positives
- FN are the false negatives

Furthermore, the ROC curve is constructed with FPR on the x-axis and TPR on the y-axis. Each point on the curve represents a different threshold. The AUC score is calculated using different methods, the most popular being the *trapezoidal rule*. The method gives the score by summing up the areas of the trapezoids created between consecutive points on the curve [22].

Precision

Precision, as defined in previous work by Saito & Rehsmeier (2015), is the ratio of positive prediction that are in fact correct. That is the assessment of how many of the positive predictions are actually true. A higher value suggests that the model has a low rate of false positive, thus a predicted positive instance is likely to be correct [50].

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{3.3}$$

Recall

Similarly, recall refers to the ratio of positive instances identified successfully by the model. It measures how well the model predicts the positive instances. As highlighted by Saito & Rehmsmeier (2015), recall is an important metric in evaluating the performance of binary classifier. A higher values implies that the model is identifying effectively the positive instances [50].

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{3.4}$$

F1 Score

F1 Score is a metric that evaluates the performance of a classification model, especially when dealing with imbalance in the distribution of classes in the dataset. Sokolova et al. (2006), in their paper suggest that the metric finds a balance between recall and precision scores, as it is calculated as the harmonic mean between the two.

$$\text{F1} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \quad \text{or} \quad \text{F1} = \frac{2 \cdot \text{TP}}{2 \cdot \text{TP} + \text{FP} + \text{FN}} \tag{3.5}$$

where:

- Precision indicates how many of the predicted positive cases are actually positive.
- Recall measures how well the model identifies all positive cases.

The fact that F1 Score gives a single value that captures the false positives and false negative values is one of its advantages [57].

Accuracy

Accuracy is another common metric used to evaluate the performance of a model. According to Tom Fawcett (2006), accuracy is defined as the percentage of correct predictions from all the predictions made by the model. In other words, it measures how well the model is doing overall. The formula for calculating accuracy is as follows:

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}} \tag{3.6}$$

Nonetheless, the accuracy metrics does not perform well and its results could be misleading when dealing with imbalanced datasets [22].

Runtime

Similarly, *runtime* metric is used to evaluate the performance of the models. Runtime is calculated as the difference between the time the model starts the training phase and the time when it provides the classification labels for each data point. The performance is measured in *seconds (s)*.

In our experiments, the runtime evaluation does not include the process of finding the optimal hyperparameters. As mentioned earlier, the Hyperopt library runs 50 iterations looking to optimize the performance of the models, significantly increasing the runtime for each model. Therefore, the process is not taken into consideration.

3.4.4. Hyperparameter Tuning

Hyperparameter tuning is the process of finding the right hyperparameters for a given model to achieve optimal performance. These parameters are determined before the learning process and are not learned during the training phase. Generally, the performance of the model is greatly influenced by its parameters [61].

In our experiments, we use the *Hyperopt* library in Python, which aims to minimize a *loss function*. An advantage of this library is its high level of customization. It requires the definition of an *objective function* that is feed with hyperparameters from a predetermined space, and an *optimize function* that runs trails [4]. To optimize the parameters of our models, we run Hyperopt with 50 iterations.

3.4.5. Model Validation

Cross Validation for Supervised Learning

As mentioned before, evaluation phase plays a key role on assessing the performance of our models. According to Refaeilzadeh et al.(2009) cross-validation is one of the most popular methods generally used to evaluate various machine learning models. It is a statistical method that splits the dataset into two groups, one group is used for training the models, whereas the other for testing or validating the model. The two groups must cross over in subsequent rounds so that each data point has equal probability of being validated.

K-fold is one the simplest forms of cross-validation. During k-fold cross evaluation the data is partitioned into k folds of equal size. Consequently, k iterations of training and testing are performed subsequently. Each iteration uses k-1 folds of data to train the models, while one fold is used for testing it. The model performance can be evaluated using various predetermined metrics [47].

Validation for Unsupervised Learning

Cross validation is not a common process used for performance evaluation of unsupervised learning models. Due to the nature of unsupervised learning model, evaluating their performance is nto always straightforward. Tan et al. (2018), in their book, discuss two methods of evaluating unsupervised learning models, namely internal and external validation. *Internal or unsupervised validation*, refers to evaluating the model's performance by focusing only on the input data used for running the model. It does not require external information, as it assess whether the clusters are meaningful and distinct based on the data characteristics.

The other method is called *external or supervised validation*. External validation evaluates the performance by using external information, such as class labels. It compares the model's result with an established ground truth to measure their effectiveness. Commonly used metrics for external validation include, Precision, Recall, F-measure, etc., [58]. In our datasets, class labels are present but are omitted during the model training. As a result, we have adopted the external validation. To have a better understanding of the unsupervised models' performance we will adopt three other metrics in the evaluation, namely *precision*, *recall* and *accuracy*.

3.4.6. Adopted Methodology

In the research, we have adopted the *Cross-industry standard process for data mining*, also known as **CRISP-DM process**, for our anomaly detection task. The process goes through 6 phases, as described in [53] and [52].

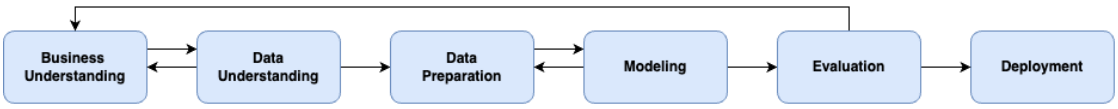


Figure 3.2.: CRISP-DM Process

Business Understanding

Business Understanding focuses on understanding the objectives of anomaly detection from a business point of view. The understanding is translated into a data mining problem related to identifying unusual patterns or outliers. Next, a plan is designed to accomplish the objectives.

Data Understanding

Data Understanding phase starts with data collection and getting familiar with the dataset. Then, proceed with checking the data quality for anomaly detection purposes. This includes further exploring the structure, identifying trends, data distributions and deviations that could be potential challenges for anomaly detection.

Data Preparation

Data Preparation phase covers all the necessary steps to create a dataset that is ready to be fed into the anomaly detection models. Such steps include, handling missing values, normalizing features, transforming variables, and feature engineering that help distinguish anomalies from normal points.

Modeling

Modeling phase involves selecting and implementing various models suitable for anomaly detection. Since there exist different types of models, it is important to select the best models for identifying outliers.

Evaluation

Evaluation phase is an important step where the performance of a model is evaluated. For the evaluation, metrics such as ROC AUC and F1 Score are used to verify the models performance on anomaly detection.

Deployment

Deployment phase is the last step of the process, where the results and the knowledge gained from the anomaly detection task are presented so that the non-technical users can benefit from it. This includes reporting to decision makers to act on the findings.

4. Findings & Results

In this chapter, we present and discuss in detail the results of our experiments. Our discussion includes a thorough investigation on the datasets, then moving into the performance results of the selected models.

4.1. Experiment 1: Real-world Datasets - Data Understanding

4.1.1. Overview of Class Distribution in All Datasets

Table 4.1.: Class Distribution Summary

Dataset	Non-Anomaly Transactions	Non-Anomaly Transactions (Percentage)	Anomaly Transactions	Anomaly Transactions (Percentage)
Dataset 1	1.289.169	**99.4%**	7.506	**0.6%**
Dataset 2	6.354.407	**99.8%**	8.213	**0.2%**
Dataset 3	587.443	**98.7%**	7.200	**1.3%**
Dataset 4	63.494	**80%**	15.106	**20%**

Table 4.1 summarizes the class distribution across all datasets. As presented in the table there is a considerable imbalance between the distribution of anomaly and non-anomaly transactions across all datasets. For Dataset 1 the majority of the transactions are labeled as non anomalies. There are *1.289.169 non anomaly transactions* labeled as *0*, and *7.506 anomaly transactions* labeled as *1*.

Similarly, in Dataset 2 there are *6.354.407 non-fraudulent transactions*, labeled as *0* and *8.213 fraudulent transactions*, labeled as *1*.

Dataset 3 has *587.443 non-anomaly transactions*, labeled as *0*, and *7200 anomaly transactions*, labeled as *1*.

Furthermore, Dataset 4 has *63.494 non-anomaly transactions*, labeled *0*, which make the majority of the labels in the dataset. By contrast, there are only *15.106 anomaly transactions*, labeled *1*.

The imbalance in the class labels can have a significant impact on the results of the experiments, hence it will be addressed in details when evaluating the performance of the models.

4.1.2. Dataset 1

Dataset 1 contains information about credit card transactions. The dataset has *1.296.675 records* and *24 features*. The dataset contains a feature labeled as *Unanmed: 0* which will be dropped from the analysis as it does not posses valuable information.

Table 4.2.: Dataset 1 Feature Details

Feature	**Data Type**
Unnamed: 0	int64
trans_date_trans_time	object
cc_num	int64
merchant	object
category	object
amt	float64
first	object
last	object
gender	object
street	object
city	object
state	object
zip	int64
lat	float64
long	float64
city_pop	int64
job	object
dob	object
trans_num	object
unix_time	int64
merch_lat	float64
merch_long	float64
is_fraud	int64
merch_zipcode	float64

Table 4.2 gives an overview of the features present in the dataset, along with their datatype. The majority of the features are categorical variables, which need to be transformed into numerical variables for the anomaly detection task. In addition, all the features have no missing values, except for *merch_zipcode*, which represents the zip code of the merchant where the transaction occurred. Due to the high number of missing values, this feature will be dropped and not taken into consideration during the experiments.

Figure 4.1 is the correlation matrix, revealing the correlation coefficients between the features of Dataset 1. The matrix suggests that the features *long* and *merch_long* have a perfect positive correlation.

long represents the geographical coordinates of the transaction, whereas *merch_long* represents the geographical coordinates of the merchant. This strong correlation suggests that the transactions are occurring at the exact same location as the merchant physical location. Thus, there is no geographical discrepancy between the locations, which implies that the transactions are happening in-person.

On the other hand, the feature *zip* is negatively correlated to the features *merch_long* and *long*. The *zip* feature contains the address details of the cardholder. The negative correlation could indicate that the merchant's location is often far from the cardholder's

registered address details. The cardholder could be traveling and the transaction are occurring in geographically distant areas from where the person resides, including different cities or countries.

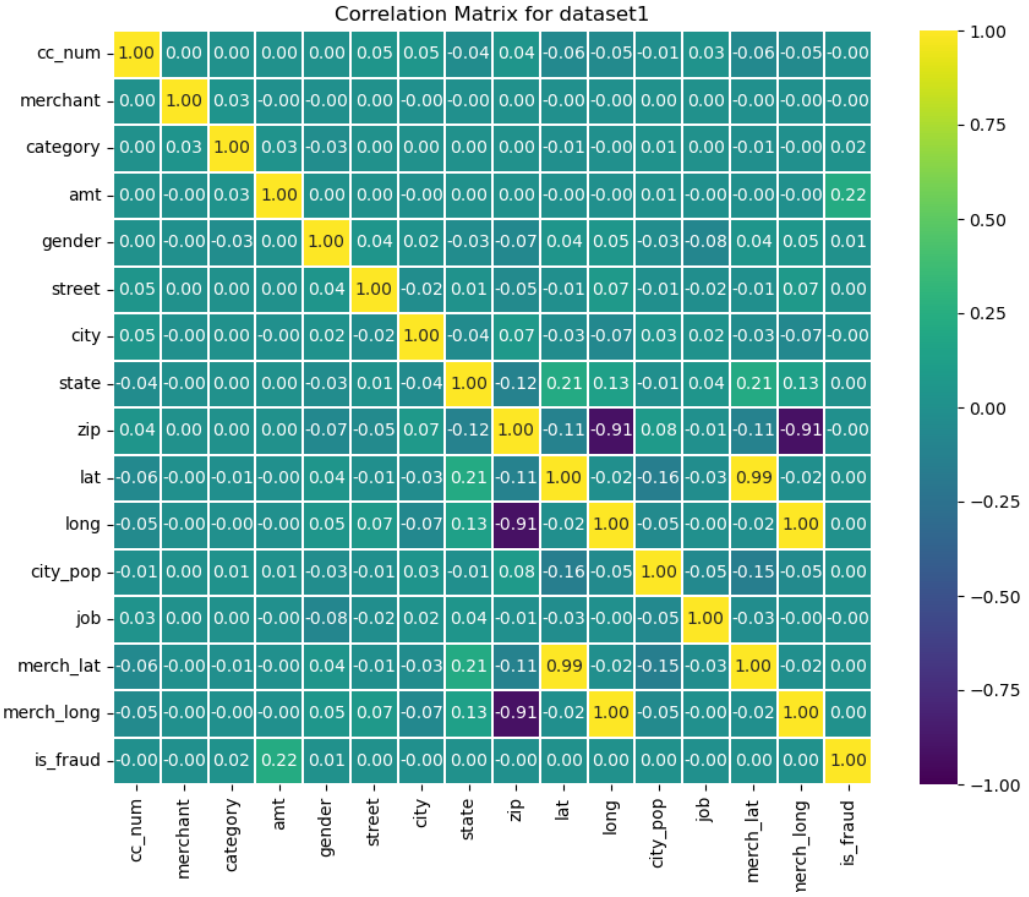


Figure 4.1.: Dataset 1 Correlation Matrix

Figure 4.2 displays a two-dimensional visualization of the data from Dataset 1. The blue points represent normal data point, whereas red points are anomalies. The plot reveals that the anomaly points and normal points are overlapping. This indicates that there are no distinguishable patterns between anomaly and non-anomaly points, and there is a small deviation from each other.

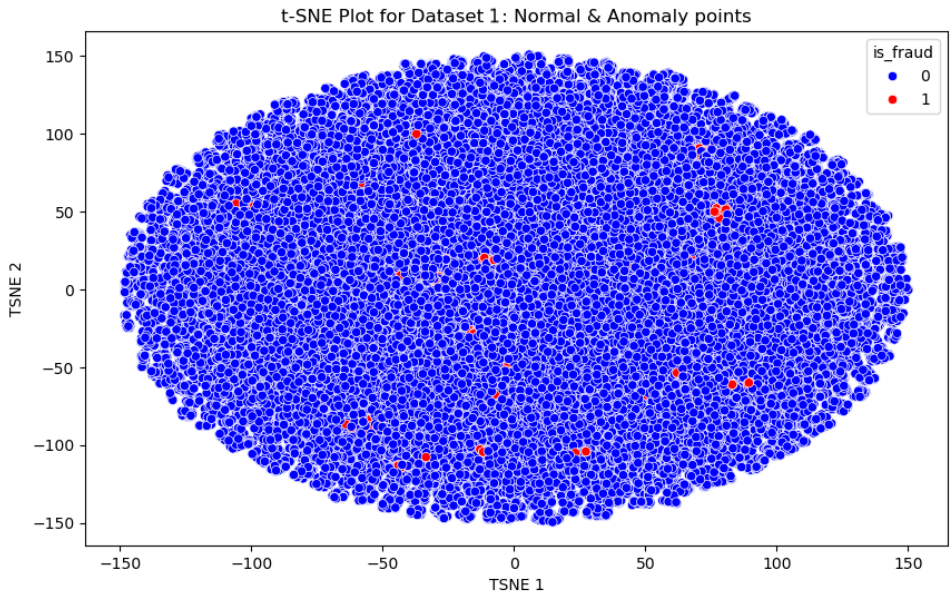


Figure 4.2.: Dataset 1: Anomaly & Non-Anomaly Points

As mentioned beforehand, the literature categorizes the anomalies into point anomalies, collective anomalies, and contextual anomalies. If we attempt to group the anomalies of Dataset 1, then all transactions labeled as anomalies are considered as point anomalies, since they represent a single anomalous transaction at a point in time. The red group of anomalies in the top right of the Figure 4.2 are a representation of collective anomalies. By further investigating the dataset, we notice that transactions with the largest amount are all taking place in the middle of the night. This transactions represent a form of contextual anomalies, Table 4.3.

Table 4.3.: Dataset 1: Contextual Anomalies

Transaction date	**Amount**	**Class Label**
2.1.19 1:06	1376.04	1
2.1.19 1:47	1371.81	1
2.1.19 3:05	1334.07	1
2.1.19 3:38	1324.80	1
2.1.19 3:55	1313.18	1
2.1.19 13:38	1312.98	1
2.1.19 23:52	1294.83	1
3.1.19 1:05	1292.21	1
3.1.19 1:35	1289.89	1
3.1.19 3:17	1289.07	1

4.1.3. Dataset 2

Dataset 2 possess details about fraudulent transactions of a financial company. The dataset has *6.362.620 records* and *10 features*.

Table 4.4.: Dataset 2 Feature Details

Feature	**Data Type**
step	int64
type	object
amount	float64
nameOrig	object
oldbalanceOrg	float64
newbalanceOrig	float64
nameDest	object
oldbalanceDest	float64
newbalanceDest	float64
isFraud	int64
isFlaggedFraud	int64

Table 4.4 displays all the features of Dataset 2. In this dataset, most of the features are numerical variables, while only three being categorical variables. As with the previous dataset, these categorical features will be transformed into numerical variables. All the features are complete, meaning there are no missing values on the dataset.

Figure 4.3 shows the correlation matrix, along with the correlation coefficients for Dataset 2. The matrix reveals a perfect correlation between the *oldbalanceOrg* feature and *newbalanceOrg* feature. The *oldbalanceOrg* represents the initial balance before the transaction, while the *newbalanceOrg* is the balance after the transaction. The correlation between the balances it is expected to be perfect, because the transaction sum is the key factor that influences a change in the balances. Moreover, it could indicate that the transaction amounts are relatively small in comparison to the overall account balances. Therefore, the difference between both balances is minimal. Additionally, this can happen when most transactions have a consistent pattern. The finding is strongly supported as well by the near perfect correlation between the *oldbalanceDest* feature and *newbalancedest* feature. The *oldbalanceDest* feature is the initial balance of the recipient before the transaction, and *newbalancedest* feature is the balance of the recipient after the transaction.

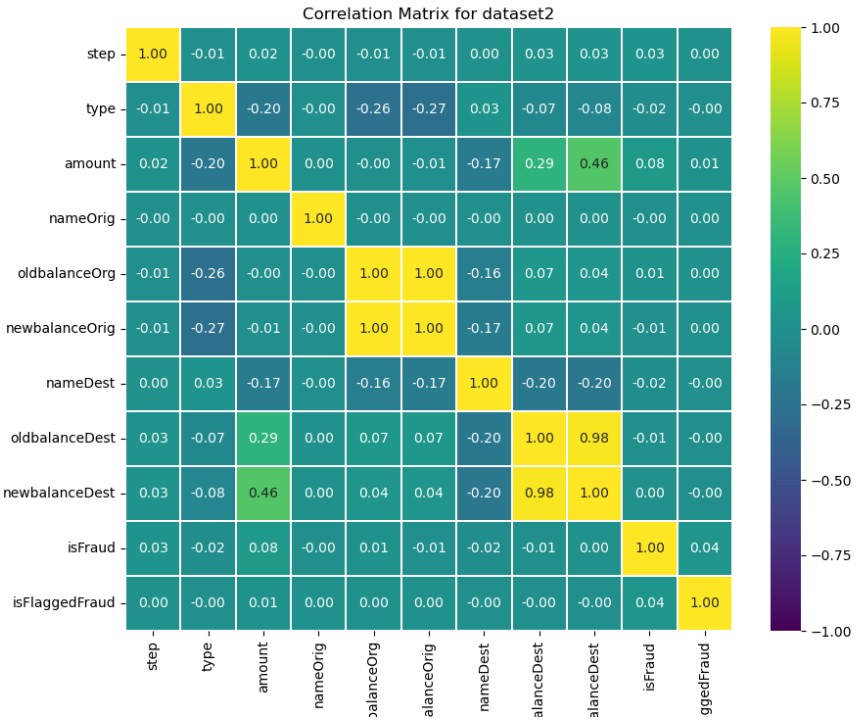


Figure 4.3.: Dataset 2 Correlation Matrix

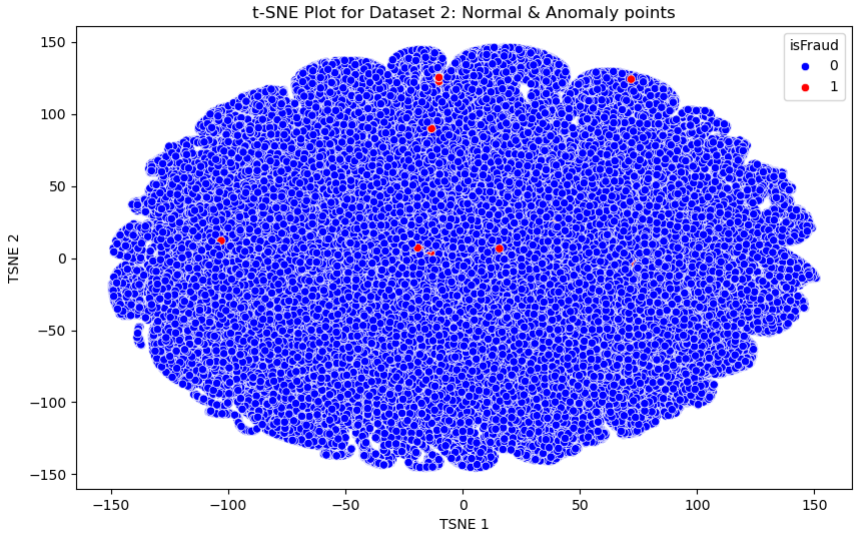


Figure 4.4.: Dataset 2: Anomaly & Non-Anomaly Points

Figure 4.4 offers a view of the distribution of anomaly transactions in Dataset 2. As before, the normal transactions (data points) are represented by the blue point, whereas the red points are anomaly transactions. There is an overwhelming amount of normal data points, that cover all the plot, with some anomalies appearing throughout. Once again, this means that there is a minimal difference between the classes. Thus, we can safely assume that anomaly transactions do not follow some trends.

Once again, for point anomalies, each anomalous transactions in the dataset is considered as point anomaly. Based on Figure 4.4, there is no visible group of anomalies that together could be considered as collective anomalies. Table 4.5 represent an example of contextual anomalies. From the sample of the dataset, it is noticeable that transfers between accounts have usually large amounts of money.

Table 4.5.: Dataset 2: Contextual Anomalies

Transaction Type	**Amount**	**Class Label**
TRANSFER	10000000.00	1
TRANSFER	9996886.64	1
TRANSFER	9977761.05	1
TRANSFER	9960382.40	1
TRANSFER	9887819.06	1
TRANSFER	9725837.08	1
TRANSFER	9639524.70	1
TRANSFER	9585040.37	1

4.1.4. Dataset 3

Dataset 3 provides data related to a financial payment system. The dataset is created to replicate the information provided by a bank in Spain. Due to data protection and privacy concerns, the real information cannot be provided. The Dataset 3 contains *594.643 records* and *10 features*. Table 4.6 below outlines the data types of the present in the dataset. The greater part of the features are categorical variables, which, as previously mentioned, will be further transformed into numerical variables to facilitate the experiments. Notably, the dataset contains no missing values.

Table 4.6.: Dataset 3 Feature Details

Feature	**Data Type**
step	int64
customer	object
age	object
gender	object
zipcodeOri	object
merchant	object
zipMerchant	object
category	object
amount	float64
fraud	int64

The correlation matrix shown in Figure 4.5 presents relatively low correlation coefficients between the features in the dataset. The *amount* feature has a correlation coefficient of 0.49 with the class label feature *fraud*. The *amount* feature describes the total value of the transaction. A correlation coefficient of 0.49 is not high, however it can imply that higher transaction value have a slightly higher likelihood of increasing the class meaning, hence being labeled as fraud.

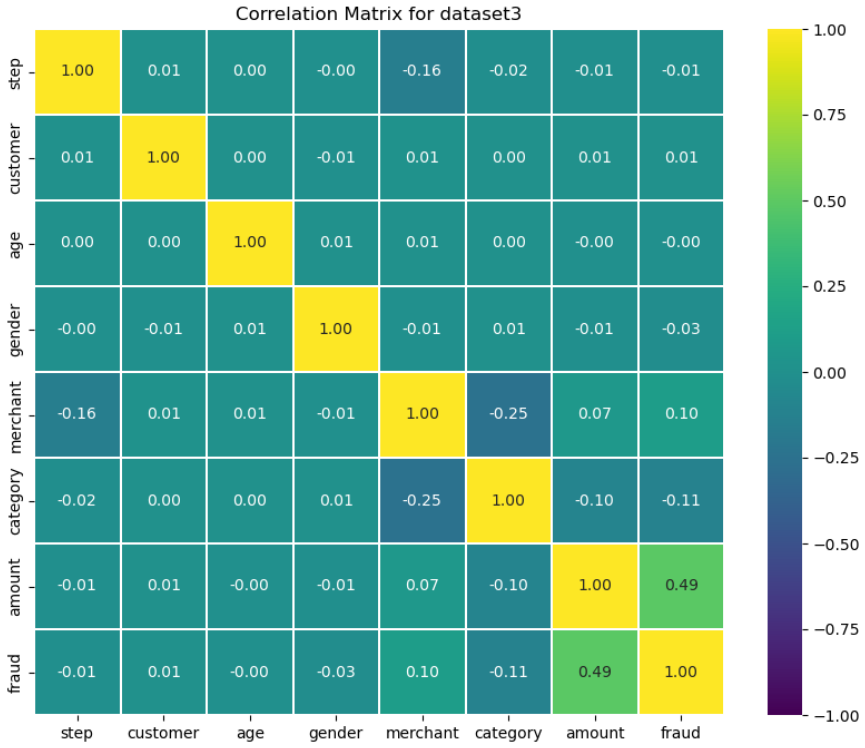


Figure 4.5.: Dataset 3 Correlation Matrix

Figure 4.6 showcases a visualization of the normal data (blue points) and anomalies (red points). The plot presents significant overlap between both classes. In addition, there is a massive overlap between the normal data points, as indicated by the light blue regions in the plot. Nonetheless, there is small region of anomaly points, that is clearly separated from the rest of the data points. This implies that anomalies belonging to that region share similar characteristics to each other that are easily distinguishable from the other group.

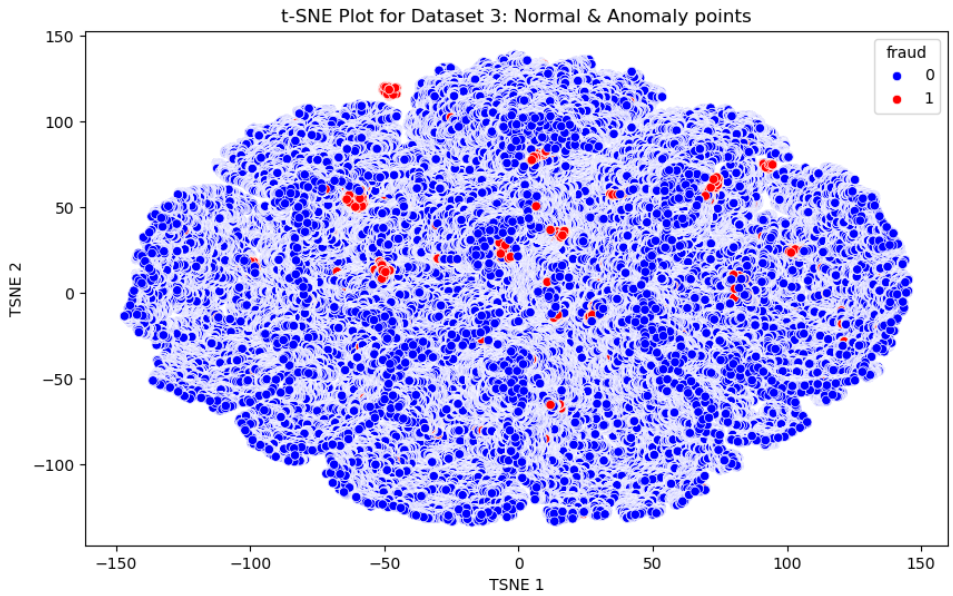


Figure 4.6.: Dataset 3: Anomaly & Non-Anomaly Points

For Dataset 3, taking into account each transaction labeled as anomaly point, individually, then we would have examples of point anomalies in the dataset. As illustrated in Figure 4.6, there are several red points, or anomaly points, that form groups and are representations of collective anomalies. Furthermore, Table 4.7 presents an instance of contextual anomalies. In the dataset, certain anomaly transactions fall into the category of 'sport and toys'. Without additional details, these transactions can be classified as anomaly points, since generally transactions in the category of 'sports and toys' do not have large amounts.

Table 4.7.: Dataset 3: Contextual Anomalies

Transaction category	**Amount**	**Class Label**
'es_sportsandtoys'	1258.330	1
'es_sportsandtoys'	1201.130	1
'es_sportsandtoys'	1167.960	1
'es_sportsandtoys'	1157.070	1
'es_sportsandtoys'	1142.230	1
'es_sportsandtoys'	1140.840	1
'es_sportsandtoys'	1122.350	1
'es_sportsandtoys'	1097.590	1
'es_sportsandtoys'	1079.270	1
'es_sportsandtoys'	1078.750	1

4.1.5. Dataset 4

Dataset 4 contains information on financial transactions in the blockchain. In comparison to the other datasets, this one is smaller in size, with *78.600 records* and *14 features*.

Table 4.8.: Dataset 4 Feature Details

Feature	**Data Type**
timestamp	object
hour_of_day	int64
sending_address	object
receiving_address	object
amount	float64
transaction_type	object
location_region	object
ip_prefix	float64
login_frequency	int64
session_duration	int64
purchase_pattern	object
age_group	object
risk_score	float64
anomaly	object

As presented in Table 4.8, the dataset has an equal split between the categorical and numerical variables. For better comprehension and the opportunity to include the features in our experiments, the categorical variables will be transformed into numerical variables. On top of that, there are no missing values on the dataset.

Figure 4.7 presents the correlation matrix for Dataset 4. There is a high correlation coefficient of 0.87 between the *login_frequency* feature, the user's login count, and the *session_duration* feature, time spent in session. This strong correlation hints that users who are more active, by logging in more frequently, tend to have longer session duration. On one hand, frequent users could engage on more complex transactions that require more time to carry out. On the other hand, it could indicate that we are dealing with *power users or automated systems*.

Furthermore, *login_frequency* has a moderate negative correlation with *purchase_pattern*. It suggests that users who log in more frequently are not necessarily purchasing more. Additionally, *login_frequency* has a moderate positive correlation with *age_group*. It revels that older users are more likely to frequently log in.

Nevertheless, the *risk_score* feature, which depicts the transaction risk rating, has a correlation coefficient of 0.52 with both *purchase_pattern* and *age_group*. The correlation with *purchase_pattern* might indicate that frequent or high-value purchasing behaviors are associated to a higher risk score.

Likewise, a correlation of 0.52 among *purchase_pattern* and *age_group*, might reflect that older users have a higher risk score, a result of engaging into riskier behavior. Nonetheless, it is worth mentioning that *purchase_pattern* is an influential feature on the dataset. Similarly, *risk_score* and the class label feature *anomaly* have a strong positive correlation of 0.82. It implies that *risk_score* is effectively predicting the likelihood a transaction is flagged as anomaly or not.

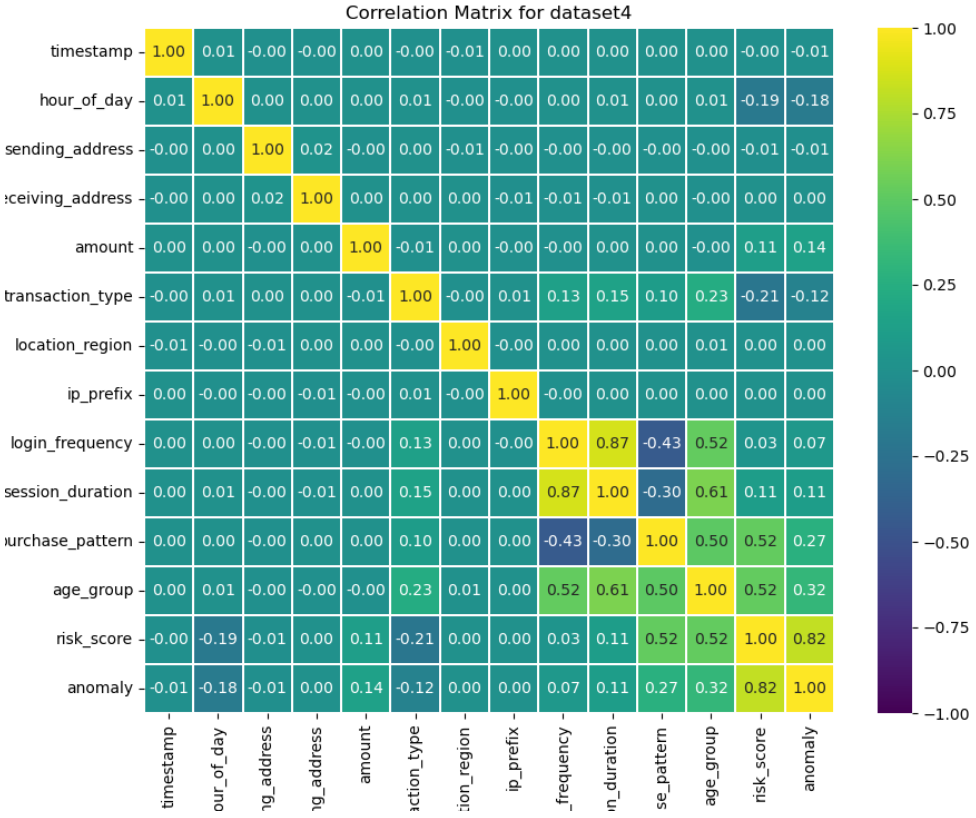


Figure 4.7.: Dataset 4 Correlation Matrix

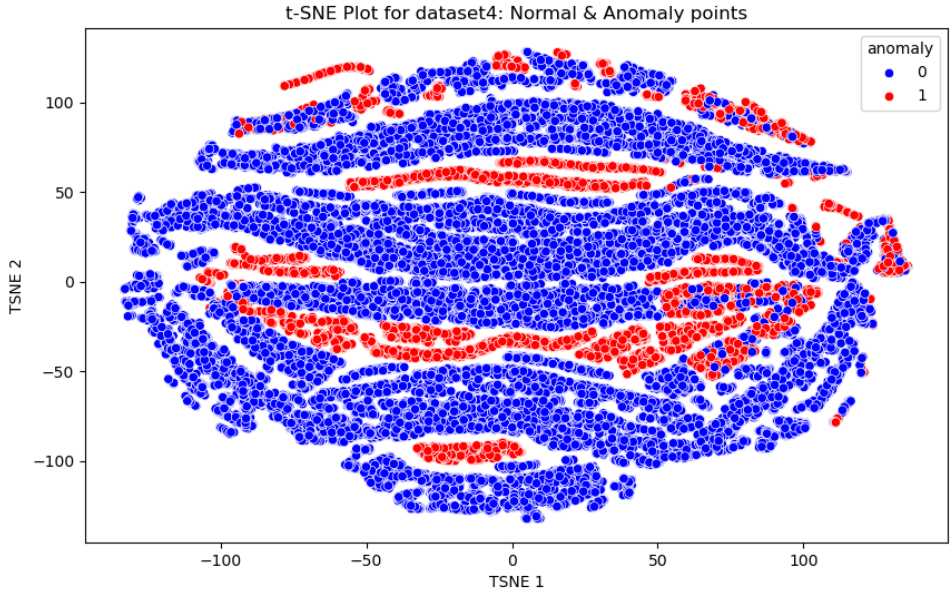


Figure 4.8.: Dataset 4: Anomaly & Non-Anomaly Points

Figure 4.8 illustrates the distribution of anomaly points (red points) and normal points (blue points). The blue data points tend to form more cohesive clusters, implying that normal data share similar characteristics. On the other hand, anomaly data have a more diverse behavior. There are certain regions where the red points are grouped together, however there are few overlapping regions with the normal points. This suggests that most anomalies have distinct patterns, while the rest share similarities with the normal data points.

When examining the transactions from Dataset 4 labeled as anomaly points individually, we observe examples of point anomalies. Based on Figure 4.8, we notice that the majority of the anomalies form distinctive groups, where each group is thought to be a form of collective anomalies. Table 4.9, demonstrates contextual anomalies in the dataset. Transactions that have random purchase patterns are categorized as phishing or scams. At the same time, these transaction have large amounts, therefore they are all labeled as anomaly points for further investigations.

Table 4.9.: Dataset 4: Contextual Anomalies

Transaction Category	**Purchase Pattern**	**Amount**	**Class Label**
phishing	random	1393.030	1
scam	random	1337.710	1
phishing	random	1320.390	1
phishing	random	1320.370	1
phishing	random	1273.700	1
scam	random	1261.330	1
scam	random	1252.760	1
phishing	random	1249.040	1
scam	random	1229.500	1
scam	random	1229.180	1
phishing	random	1207.120	1
phishing	random	1206.540	1

4.2. Experiment 1: Real-world Datasets - Model Evaluation

As previously stated, Experiment 1 was conducted on four real world dataset across different supervised and unsupervised machine learning models. The selected models include distance-based models such as *K-Neearest Neighbors (KNM), Support Vector Machine(SVM), Local Outlier Factor (LOF), Principal Component Analysis(PCA), DBSCAN.* For distance-based models is important to standardize the data to ensure that all the features have an equal contribution on the model. This has an impact when calculating distance measures, namely Euclidean distance. In our experiments we have used the *StandardScaler* from *sklearn* library for data standardization. The rest of the models are able to handle non-standardized data, thus data standardization is not required.

Due to the highly imbalance datasets, in regards to the distribution of anomaly and non-anomaly data points, the performance metrics, namely ROC AUC, F1 Score, Precision, Recall and Accuracy are calculated using the weighted parameter to balance the class

distribution.

Another point of discussion is the number of features in each dataset. Every dataset contains certain features that do not provide valuable information, hence are excluded from the analysis. This aims to enhance the performance and improve computational efficiency.

- **Dataset 1:** Unnamed: 0', 'trans_date_trans_time', 'trans_num', 'unix_time', 'dob', 'first', 'last', 'merch_zipcode'.
- **Dataset 2:** 'nameOrig', 'nameDest'.
- **Dataset 3:** no features dropped.
- **Dataset 4:** 'timestamp', 'sending_address', 'receiving_address'.

4.2.1. Evaluation of Supervised Learning Models

Table 4.10.: Supervised Learning Models Performance

Dataset	Model	Cross Validation			Holdout Validation		
		ROC AUC	**F1 Score**	**Runtime (s)**	**ROC AUC**	**F1 Score**	**Runtime (s)**
Dataset 1	KNN	0.688	0.995	45.894	0.891	0.993	16.894
	Random Forest	0.808	**0.997**	2591.672	0.988	**0.997**	62.63
	XGBoost	**0.906**	0.994	32.68	**0.996**	**0.997**	0.901
	SVM	0.621	0.994	7901.67	0.884	0.989	9.378
	Naive Bayes	0.5	0.991	**1.876**	0.843	**0.997**	**0.074**
	CatBoost	0.816	**0.997**	194.367	0.995	**0.997**	5.362
Dataset 2	KNN	0.839	0.999	154.141	0.809	0.999	138.133
	Random Forest	0.893	**1.0**	2289.313	0.888	**1.0**	355.152
	XGBoost	0.702	0.999	133.301	**0.936**	**1.0**	84.971
	SVM	0.731	0.999	11838.396	0.725	0.999	3158.488
	Naive Bayes	0.582	0.996	**8.495**	0.551	0.997	**1.278**
	CatBoost	**0.943**	**1.0**	933.603	0.935	**1.0**	139.687
Dataset 3	KNN	0.851	0.995	17.882	0.798	0.994	3.324
	Random Forest	0.875	**0.996**	195.782	0.879	**0.996**	58.382
	XGBoost	**0.9**	0.989	13.098	**0.896**	0.996	60.491
	SVM	0.834	0.995	632.823	0.744	0.993	128.561
	Naive Bayes	0.884	0.987	**0.573**	0.886	0.987	**0.12**
	CatBoost	0.892	**0.996**	133.709	0.894	0.996	14.108
Dataset 4	KNN	**1.0**	**1.0**	1.68	**1.0**	**1.0**	0.676
	Random Forest	**1.0**	**1.0**	7.639	**1.0**	**1.0**	1.967
	XGBoost	**1.0**	**1.0**	1.444	**1.0**	**1.0**	2.613
	SVM	**1.0**	**1.0**	5.372	**1.0**	**1.0**	1.106
	Naive Bayes	0.989	0.982	**0.096**	0.989	0.982	**0.019**
	CatBoost	**1.0**	**1.0**	15.781	**1.0**	**1.0**	3.145

Table 4.10 presents the performance results for the supervised machine learning models based on ROC AUC, F1 Scores, and Runtime, evaluated using cross validation and holdout validation.

For **Dataset 1**, XGBoost achieves the highest ROC AUC Scores for cross validation and holdout validation. The ROC AUC Score of 0.906 indicates that XGBoost correctly distinguishes between anomaly and normal points in 90.6% of the cases. However, there is a slight decrease in the cross validation score. Typically, it is expected that models perform better and have higher scores with cross validation. When a decrease happens, as seen here, it suggests the models have over-fitted during the holdout validation. Generally, all the models have suffered from over-fitting. Other well performing models, in terms of ROC AUC Scores, are Random Forest Classifier and CatBoost.

Regarding the F1 Score, Random Forest Classifier and CatBoost have the highest scores. The scores remain unchanged for both model for cross validation as well as holdout validation. The F1 Scores of 0.997, hints that the model are highly reliable for anomaly detection. Naive Bayes is the fastest performing model when comparing the models runtime for cross validation and holdout validation.

For **Dataset 2**, CatBoost has the highest ROC AUC Score in cross validation. Nevertheless, during holdout validation, XGBoost slightly outperformed it, with a marginally better score of 0.001. It is worth noting, with an exception of XGBoost, the rest of the model have achieved better scores with cross validation. The decrease in scores for XGBoost, indicates a potential over-fitting issue during holdout validation. In terms of F1 Score, Random Forest Classifer and CatBoost have perfect scores, while the rest of the models follow closely with near-perfect scores. One more time, Naive Bayes is the fastest model across cross validation and holdout validation.

For **Dataset 3**, once again XGBoost stands out as the best performing model in cross validation and holdout validation. All models demonstrated good ROC AUC Scores, with most either improving or maintaining their scores in cross validation compared to holdout validation. For F1 Scores, once again, Random Forest Classifier and CatBoost have the highest scores, followed closely by the rest of the models. Similarly, Naive Bayes continues to be the fastest performing model, significantly outpacing the other models in terms of runtime.

For **Dataset 4**, nearly all the models achieved perfect ROC AUC Scores and F1 Scores for cross validation and holdout validation. This is because of the dataset structure, which differs from the other datasets. Dataset 4, as seen in Figure 4.8, has distinct patterns between the normal and anomaly points. Since anomalies have easily identifiable characteristics, it is easier for models to achieve near perfect scores. Furthermore, with approximately 75.000 data points, this dataset is relatively small, allowing powerful models to excel. As with the other datasets, Naive Bayes continues to be the fastest model.

Across all datasets, **XGBoost**, **CatBoost** and **Random Forest Classifier** are the best performing model, with Naive Bayes consistently demonstrating the fastest runtime. The high ROC AUC scores imply that the models are highly effective in separating between the classes. Similarly, the high F1 Scores says the models can accurately identify true positive, or anomaly points in our case, while minimizing false positives, labeling normal data points as anomalies, and false negative, missed anomaly points. This is highly valuable in a financial context, where detecting fraud or anomalies is crucial for

safeguarding assets and maintaining trust in financial system.

4.2.2. Evaluation of Unsupervised Learning Models

Table 4.11 presents the performance results of the unsupervised machine learning models based on ROC AUC, F1 Scores, Precision, Recall, Accuracy and Runtime.

Table 4.11.: Unsupervised Learning Models Performance

Dataset	Model	ROC AUC	F1 Score	Precision	Recall	Accuracy	Runtime (s)
Dataset 1	LOF	0.746	**0.986**	0.991	**0.981**	0.639	231.333
	PCA	0.749	0.942	0.991	0.900	0.709	**2.335**
	Isolation Forest	0.780	0.854	0.992	0.754	0.725	13.012
	DBSCAN	**0.867**	0.974	**0.993**	0.958	**0.867**	628.642
	COPOD	0.682	0.940	0.989	0.897	0.567	27.019
	ECOD	0.646	0.940	0.989	0.897	0.551	25.610
Dataset 2	LOF	**0.842**	**0.985**	**0.998**	0.973	0.641	1412.685
	PCA	0.820	0.946	**0.998**	0.900	0.648	**8.538**
	Isolation Forest	0.832	0.938	**0.998**	0.885	0.688	58.045
	DBSCAN	0.702	0.993	0.995	**0.998**	**0.992**	3087.481
	COPOD	0.758	0.946	**0.998**	0.900	0.609	133.726
	ECOD	0.756	0.946	**0.998**	0.900	0.627	99.564
Dataset 3	LOF	0.763	**0.976**	**0.997**	**0.975**	0.526	81.790
	PCA	0.932	0.940	0.985	0.905	0.814	**1.023**
	Isolation Forest	**0.971**	0.898	0.989	0.832	0.910	36.828
	DBSCAN	0.416	0.958	0.957	0.990	**0.935**	27.565
	COPOD	0.968	0.943	0.988	0.910	0.908	4.577
	ECOD	0.960	0.943	0.988	0.909	0.894	4.225
Dataset 4	LOF	0.549	0.722	0.712	0.808	0.500	5.963
	PCA	0.760	**0.813**	0.814	**0.834**	**0.642**	**0.115**
	Isolation Forest	0.717	0.402	**0.820**	0.393	0.607	4.047
	DBSCAN	0.785	0.675	0.645	0.800	0.604	10.174
	COPOD	**0.846**	0.801	0.800	0.824	0.626	1.147
	ECOD	0.783	0.798	0.795	0.821	0.621	0.583

For **Dataset 1**, DBSCAN excels at distinguishing anomalies from normal data points, achieving an ROC AUC Score of 0.867. Additionally, this models has a great Precision Score, suggesting that DBSCAN makes very few false positive errors. Precision Scores across all models are strong, with minimal differences between them.

Moreover, LOF stands out with the highest F1 Score of 0.986, which means it maintains a strong balance between detecting the anomaly points and minimizing false positives. This is supported by the LOF's high Precision and Recall values. The high Recall score indicates that the model can detect the true anomaly points with minimal false negatives.

DBSCAN scores highly in terms of Accuracy with a score of 0.867. Thus, making it the best model at correctly classifying normal and anomaly data points. Furthermore, PCA demonstrates the fastest runtime with 2.335 seconds.

For **Dataset 2**, LOF outperforms the rest of the models, obtaining the highest ROC AUC and F1 Score, which indicates the strong ability to identify anomalies while maintaing

a balance between precision and recall. In terms of Precision, all models achieve near-perfect scores. The difference between models scores is insignificant.

DBSCAN stands out as having the best Recall and Accuracy scores. There is a considerable jump in the DBSCAN's Accuracy compared to the other models, with a notable 0.31 difference with the next best model, Isolation Forest. Once again, PCA remains the fastest performing model with 8.538 seconds.

For **Dataset 3**, Isolation Forest leads with the highest ROC AUC score of 0.971. Most models have high ROC AUC scores, with the exception of DBSCAN, which lags behind with a poor score of 0.416. In terms of F1 Scores, LOF performs the best with a score of 0.976. Generally, all the models have consistently good F1 Scores, implying strong performance in precision and recall. This can be seen in the table where LOF is the best performing model for Precision and Recall.

DBSCAN, despite the low ROC AUC score, is the best model in terms of accuracy, with a score of 0.935, whereas LOF has the lowest scores. The contrast between the ROC AUC and Accuracy, indicates that DBSCAN is correctly clustering the majority or the normal data points but but fails to distinguish the anomaly or minority points. PCA continues to be the fastest model, completing the execution time in 1.023 seconds.

For **Dataset 4**, COPOD achieves the highest ROC AUC score of 0.846. In contrast, LOF scores poorly with 0.546, suggesting the model is performing similarly to random guessing. PCA excels with the best F1 Score. However, Isolation Forest has the lowest F1 Score, despite having high in Precision. This discrepancy implies that while it makes accurately positive predictions, it fails in terms of predicting true positives leading to a low Recall. Regarding Accuracy, all the models performance is suboptimal, with PCA having the highest score of 0.642. In Dataset 4, PCA is the best performing model in terms of recall and runtime, as well, due to the relatively small size of this dataset.

Overall, there is no single model that performs well across all datasets and evaluation metrics. Isolation Forest demonstrates relatively good performance in the majority of cases, however it tends to struggle with smaller datasets. DBSCAN and PCA also shows decent performance, especially in terms of F1 Score, Precision, Recall and Runtime.

4.2.3. Comparative Analysis of Models Performance

Figure 4.9 presents a comparative analysis of models used for anomaly detection in terms of ROC AUC scores and Runtime across the four datasets. Each subgraph (a, b, c, d) represents one dataset. On the x-axis the Runtime performance is plotted measured in seconds, whereas on the y-axis are the corresponding ROC AUC scores. The models are evaluated based on their trade-off between classification accuracy and computational efficiency.

Dataset 1 (a): XGBoost and DBSCAN are the most effective models for anomaly detection. The models offer a balance between performance and relatively low runtime. On the contrary, models such as Random Forest Classifier have longer runtime, despite achieving a satisfactory ROC AUC score. Similarly, SVM not only has one of the lowest ROC AUC scores but also proves to be the slowest mode, requiring more computational

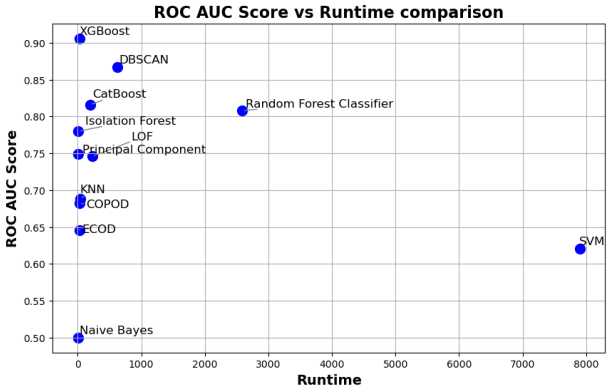
resources, and has one the lowest ROC AUC scores. Naive Bayes despite being the fastest model, obtains the lowest ROC AUC score among all models.

Dataset 2 (b): CatBoost and Random Forest Classifiers are the best performing models, with CatBoost being faster and achieving higher ROC AUC scores. As before, SVM falls short in terms of ROC AUC and Runtime trade-off. Despite being slightly behind with ROC AUC scores, KNN, Isolation Forest, and PCA, are faster than CatBoost, which makes them a strong options for when runtime is a priority.

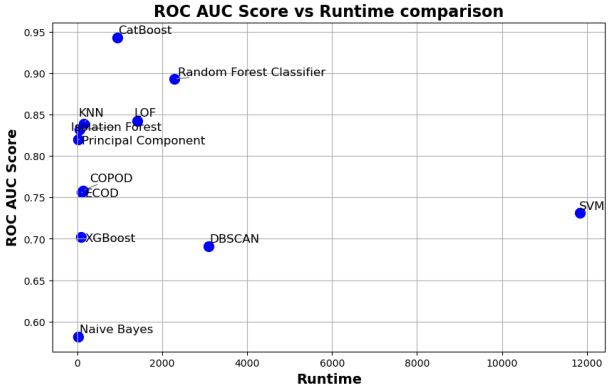
Dataset 3 (c): With near perfect ROC AUC scores and excellent runtime performance, newer models like COPOD and ECOD perform the best. Isolation Forest also demonstrates strong performance, though with a slightly higher runtime. Nonetheless, the majority of the models show great results, both in terms of ROC AUC and Runtime. SVM continues to be a computationally expensive model, however its ROC AUC score has improved.

Dataset 4 (c): The majority of the models achieve perfect ROC AUC scores, namely XGBoost, KNN, SVM, Random Forest Classifier and CatBoost. Due to the smaller dataset size, the runtime is lower compared to the previous datasets. The only models that do not perform well in terms of ROC AUC scores are LOF and DBSCAN.

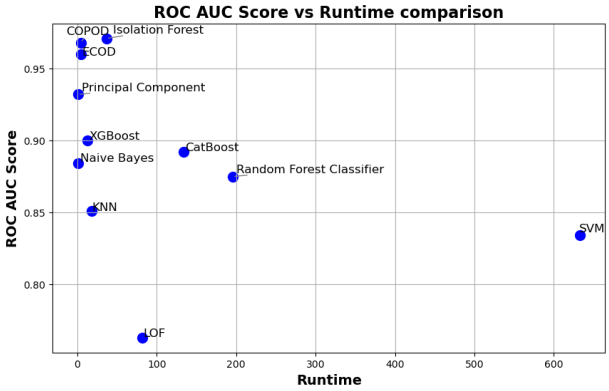
To sum up, if the primary focus is on the models' effectiveness in distinguishing between anomaly and normal points, Random Forest Classifier and CatBoost achieve the highest ROC AUC score across all datasets. This matters when the applications are highly sensitive, as for fraud detection or regulatory compliance. Additionally, if runtime is a priority, then XGBoost, ECOD, COPOD, PCA, KNN and Isolation Forest are the fastest running models. Speed will remain critical for systems needing to detect anomalies in real-time, such as trading systems, and risk monitoring, among others. If a balance is required, then XGBoost, CatBoost and Random Forest Classifier provide a reasonable trade-off between ROC AUC score and Runtime.



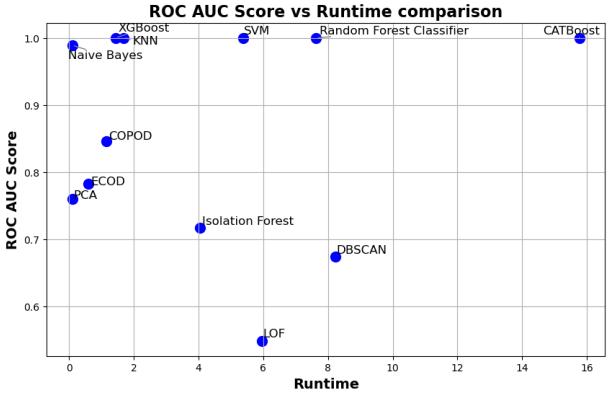
(a) Dataset 1



(b) Dataset 2



(c) Dataset 3



(d) Dataset 4

Figure 4.9.: ROC AUC & Runtime Comparison

Figure 4.10 presents a comparative analysis between F1 Score and Runtime for all datasets. For each graph, the x-axis represents the runtime measured in seconds, and the y-axis the F1 Scores.

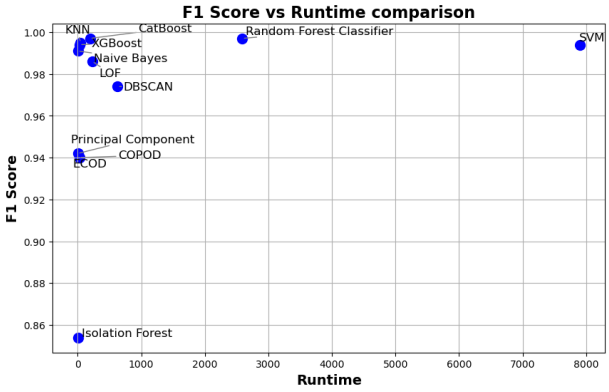
Dataset 1 (a): KNN, XGBoost, CatBoost, Naive Bayes and LOF demonstrate the best F1 Score with the fastest runtime. However, although Random Forest Classifier has almost perfect F1 Score, is also computationally expensive. Likewise, SVM is the slowest model despite the great F1 Score.

Dataset 2 (b): KNN, Naive Bayes, and XGBoost achieve the highest F1 scores in the fastest time. ECOD, COPOD, PCA, and Isolation Forest show excellent runtime performance, even though their F1 scores slightly decrease. SVM is the most computationally expensive model, with a runtime of 12.000 seconds.

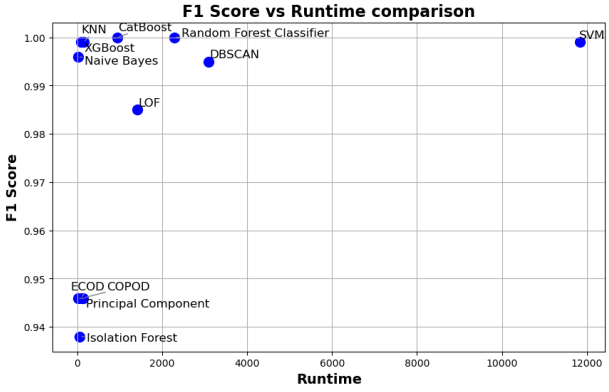
Dataset 3 (c): Similarly, KNN, Naive Bayes, and XGBoost once again are the best performing models overall. CatBoost and Random Forest Classifier, also have great results but their runtime is notably higher. From all the models, Isolation Forest achieves the lowest F1 score, while SVM remains the slowest model.

Dataset 4 (d): The vast majority of the models achieve perfect F1 scores, and the runtime is almost inconsiderable, with CatBoost being the slowest at just 16 seconds. Nonetheless, some models, despite having great runtime, show lower F1 Scores, such as Isolation forest (0.4) and DBSCAN (0.6).

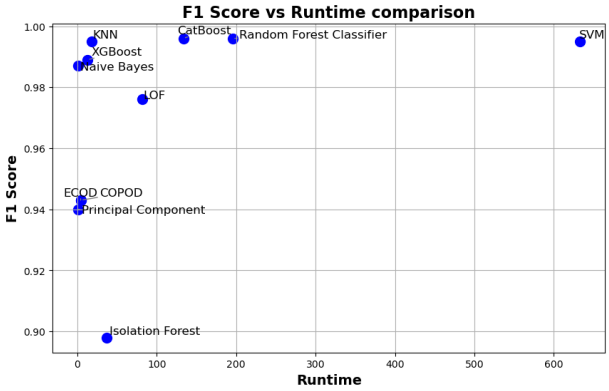
To summarize, if accuracy is considered as the primary goal of the anomaly detection task, KNN, XGBoost, CatBoost, and Random Forest classifiers achieve near-perfect F1 Scores across all datasets. If priority is runtime, Naive Bayes and PCA are the fastest models, while KNN, XGBoost and CatBoost offer the best trade-off between accuracy and runtime for a balance between accuracy and runtime.



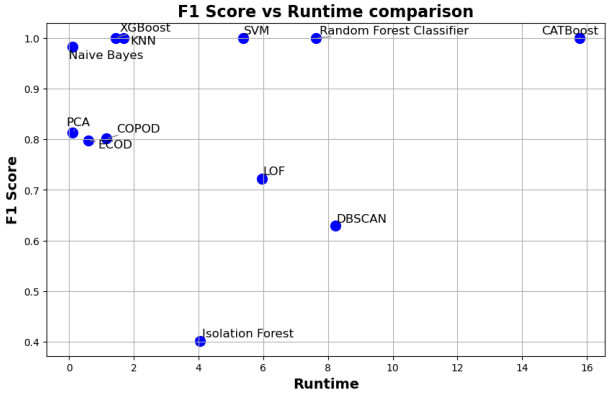
(a) Dataset 1



(b) Dataset 2



(c) Dataset 3



(d) Dataset 4

Figure 4.10.: F1 Score & Runtime Comparison

4.2.4. Distance Based Validation for Cluster Anomaly Detection

Distance based validation for cluster models, such as DBSCAN, offers a better understanding as to why certain data points are labeled as anomalies. The distance calculation can demonstrate that anomaly points are statistically distinct from regular data points. Furthermore, it can further verify the performance of the metrics. If the distance of anomalies to the cluster centroids is constantly larger than the average distance between the cluster points and the centroids, it can indicate that the model has performed well. Thus, performing as a robustness check. In addition, by using the distance based validation along with performance metrics, it helps create a more comprehensive overview of the anomaly detection process.

To perform the distance based validation for anomaly detection, we will focus solely on DBSCAN, as a classic clustering model. The model does not require a predefined number of clusters but groups points based on their density. For our analysis, we use the optimal parameters determined during the hyperparameter tuning process. For each dataset, DBSCAN generates large numbers of clusters:

- Dataset 1: 1129 clusters
- Dataset 2: 28 clusters
- Dataset 3: 193 clusters
- Dataset 4: 45 clusters

Given the large number of clusters in the results, this section will provide a summary of descriptive statistics for the distance validation of the first 10 clusters in each dataset. A detailed view of each cluster's result across all datasets will be included in the Appendix A.1. The *Cluster Index* represents the reference number for the cluster. The *Anomaly Distance (Mean)* shows the average distance of an anomaly point to the cluster centroid. The *Point Distance (Mean)* is the average distance of a cluster point to the centroid. The *Ratio* is calculate as the ratio of *Anomaly Distance (Mean)* and *Point Distance (Mean)*. The *Anomaly Distance (Min)* and *Anomaly Distance (Max)* offers the minimum and maximum distance of anomaly points from the cluster.

Table 4.12 presents an overview of distance validation for 10 clusters in Dataset 1. In average, the distance of anomaly point to the centroid of these clusters is between 5 and 7, whereas the average distance of a cluster point to the centroid is around 1.3. This results in a high ratio signifying that anomalies are generally disperse and far from the cluster centroid, compared to the cluster points.

However, as displayed on the last two columns, there are anomalies that have a lower distance than the average point distance. Despite that, these data points are labeled as anomalies by DBSCAN. From our previous evaluation, DBSCAN for Dataset 1 achieved a high ROC AUC score of 0.867, highlighting the effectiveness of the model, yet the model is not accurate in all its predictions.

Table 4.12.: Distance Validation Performance Summary - Dataset 1

Cluster Index	Anomaly Distance (Mean)	Point Distance (Mean)	Ratio	Anomaly Distance (Min)	Anomaly Distance (Max)
0	5.046	1.393	3.621	1.141	180.185
1	7.010	1.353	5.182	1.259	180.366
2	6.325	1.335	4.737	0.666	180.373
3	5.383	1.302	4.135	1.206	180.167
4	5.606	1.399	4.009	0.419	180.362
5	5.520	1.395	3.957	1.001	180.208
6	5.519	1.376	4.010	1.170	180.348
7	5.320	1.356	3.924	0.963	180.335
8	5.240	1.360	3.851	0.922	180.205
9	5.389	1.350	3.992	1.128	180.371
10	5.639	1.352	4.171	0.999	180.320

A sample of the results for the distance validation of Dataset 2 can is shown in Table 4.13. In our Experiment 1, DBSCAN for Dataset 2 reached an ROC AUC score of 0.702, relatively a moderate score. In the table, the ratios between the distances are significant, meaning the anomaly points are far from the cluster centroids. Similarly, since the average point distance is small, across all clusters, it can be implied that the clusters are more tight. For cluster 0 and 10, DBSCAN was able to perfectly separate the anomaly points, since their minimum distances are higher than the average point distance.

Table 4.13.: Distance Validation Performance Summary - Dataset 2

Cluster Index	Anomaly Distance (Mean)	Point Distance (Mean)	Ratio	Anomaly Distance (Min)	Anomaly Distance (Max)
0	8.871	0.888	9.994	1.004	316.734
1	8.938	0.817	10.947	0.756	316.757
2	8.891	0.976	9.105	0.489	316.775
3	8.514	1.359	6.263	1.077	316.677
4	8.855	0.737	12.013	0.715	316.773
5	9.124	0.815	11.189	0.586	316.917
6	8.804	0.885	9.945	0.578	316.521
7	8.651	1.037	8.339	0.741	316.431
8	8.415	1.077	7.816	0.741	316.255
9	8.772	0.668	13.137	0.604	316.260
10	8.630	0.724	11.916	0.743	316.268

Based on the results of Experiment 1, dataset 3 had the poorest performance in terms of DBSCAN, with an low ROC AUC score of 0.416. The average distance of anomaly points to centroids, displayed on table 4.14, are smaller compared to the other dataset. The ratios between the distances is significant, however for all the clusters the minimum distance of the anomaly points to centroids is less than the average point distance. The same trends follows the majority fo the results, which can be seen in Appendix A.6.

Table 4.14.: Distance Validation Performance Summary - Dataset 3

Cluster Index	Anomaly Distance (Mean)	Point Distance (Mean)	Ratio	Anomaly Distance (Min)	Anomaly Distance (Max)
0	4.362	1.356	3.218	0.811	74.584
1	4.392	1.305	3.367	0.961	74.566
2	4.301	1.233	3.489	0.826	74.636
3	4.309	1.359	3.172	0.806	74.571
4	4.545	1.351	3.364	0.864	74.606
5	4.253	1.343	3.167	0.954	74.599
6	4.553	1.338	3.403	0.808	74.598
7	4.299	1.292	3.328	0.883	74.606
8	4.483	1.293	3.468	0.837	74.625
9	4.599	1.267	3.629	0.830	74.602
10	4.536	1.278	3.550	0.812	74.629

Table 4.15 summarizes the results of the distance validation for dataset 4. DBSCAN for this dataset achieved an ROC AUC score of 0.785 during Experiment 1. The average distance between the anomaly points and the cluster centroids varies close to and around 4. Furthermore, some the average point anomalies are low due to compact clusters. The ratios are relatively high, and is worth noting that the majority of the minimum anomaly distances are higher than the average point distance.

Table 4.15.: Distance Validation Performance Summary - Dataset 4

Cluster Index	Anomaly Distance (Mean)	Point Distance (Mean)	Ratio	Anomaly Distance (Min)	Anomaly Distance (Max)
0	4.323	1.330	3.251	1.653	6.916
1	3.968	1.073	3.699	1.144	6.503
2	4.170	1.057	3.944	1.135	6.634
3	4.327	1.328	3.258	1.772	6.616
4	4.239	0.900	4.710	1.173	7.124
5	3.898	1.057	3.690	1.156	6.305
6	4.249	0.956	4.445	1.210	6.923
7	4.498	1.317	3.416	1.797	6.587
8	4.487	1.317	3.408	1.653	7.195
9	4.426	0.962	4.600	0.782	7.199
10	3.996	0.886	4.510	1.173	6.888

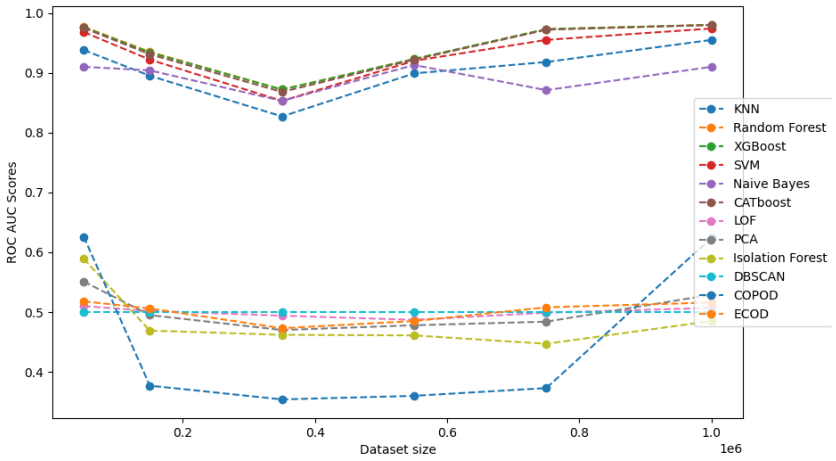
4.3. Experiment 2: Scalability Testing

Figure 4.11 describe the results of the scalability testing for the three evaluation metrics, ROC AUC, F1 Score and Runtime. The scalability test is run on synthetic dataset of various sizes, 50.000, 150.000, 350.000, 550.000, 750.000, and 1.000.000 data points. Each plot has the dataset sizes plotted on the x-axis with the corresponding metric on the y-axis. Every line represents one of the machine learning models.

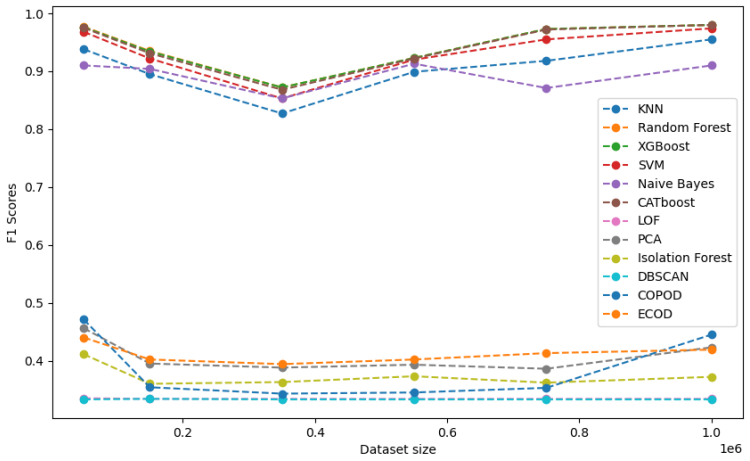
ROC AUC (a): Across all models, ROC AUC scores are generally stable. Some models such as KNN, SVM, CatBoost, XGBoost and Naive Bayes tend to have a small fluctuation, however their scores tend to be more consistent as the dataset increases. COPOD, model depicted with the light blue, has more variance in its performance. The model starts strong but the performance decreases with larger datasets and it increases again in the end. COPOD is a model that is highly sensitive to the underlying data distribution. In synthetic datasets, with an increase in dataset size the distribution could shift or data patterns become more complex. This can cause the shift in the performance of COPOD.

F1 Score (b): The performance of the models tends to mimic the performance of ROC AUC. Generally, the model have stable scores as the dataset size progresses. COPOD's performance appears to be more stabilized in comparison to the ROC AUC.

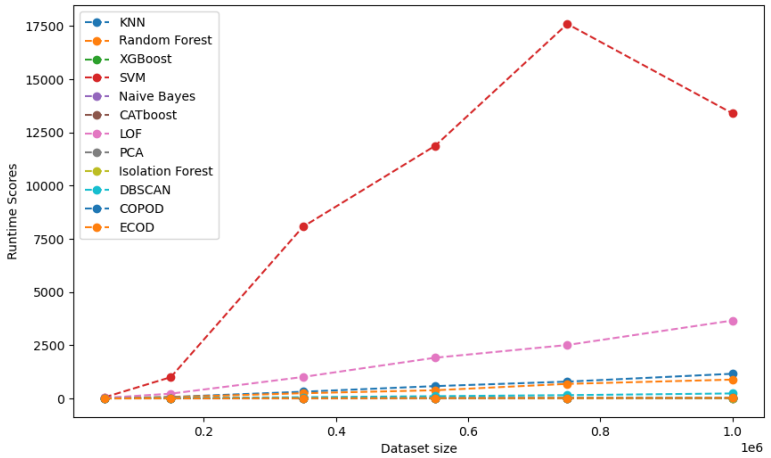
Runtime (c): The majority of the models tend to have a steady runtime performance as the dataset grows. There is a small increase in runtime for some of the model, as it is expected. Nonetheless, in the case of SVM and LOF the runtime performance drastically deviates. LOF has a linear raise in runtime, which is notable compared to the rest. However, is not as considerable compared to the runtime of SVM. The model is computationally expensive, with a time complexity between $O(n^2)$ and $O(n^3)$. This means as the dataset size increase, the runtime exhibits quadratic growth, that is attributed to the pairwise comparisons SVM has to make between data points.



(a) ROC AUC Score



(b) F1 Score



(c) Runtime (s)

Figure 4.11.: Scalability Testing

5. Conclusions

Anomaly Detection, also known as outlier detection, is the process of identifying data points that do not follow establish patterns. It has a long history of application, primarily in statistics, nonetheless with passing time it has been used across various domains. An anomaly event could indicate fraud, network issues, diseases or faulty equipment. One of the biggest challenges, when dealing with anomalies, is defining what is a normal point.

The developments in the financial industry have closely connected it to technology and data with the purpose of improving decision-making process and increasing efficiency. Though, the technological advancement have increased the volume of data the business is dealing with, creating a growing need for anomaly detection. In finance, an anomaly could lead to fraudulent transaction, identity theft, money laundering, etc.

In this paper, we compared twelve machine learning models across four real-world financial datasets to determine the most suitable models for anomaly detection task in finance. Among the supervised learning models chosen is K-Nearest Neighbors (KNN), a simple model to implement, easy to interpret, and flexible, as it makes no assumptions about the underlying data distribution. However, the performance is sensitive to the choice of 'k', it can become computationally expensive with large datasets, and is vulnerable to the curse of dimensionality.

Random Forest Classifier efficiently handles imbalanced and high-dimensional datasets, is robust, and scales well to large datasets. Nonetheless, it is sensitive to noisy data and can be both memory and computationally expensive.

Support Vector Machine (SVM) performs well with high-dimensional data and is less sensitive to outliers (anomalies), allowing for non-linear decision boundaries. However, it faces challenges with scalability, performing poorly with large dataset, being memory intensive and struggling with imbalanced datasets.

Naive Bayes is a simple model that requires minimal training time and provides probabilistic predictions, making it more robust to noise. Though, it has limited interpretability and relies on the assumption that all features are independent, which is rarely the case.

XGBoost has strong predictive performance. Due to its distributed computation and parallel processing capabilities, is highly scalable. The model handles well imbalanced datasets and captures complex relationships in the data. Nevertheless, is sensitive to noise. In terms of training time and memory usage, XGBoost can be computationally expensive and requires careful tuning of its parameters.

CatBoost is specifically designed to handle categorical variables without needing data pre-processing. It requires less hyperparameter tuning, is fast, and scalable. Similarly, it prevents overfitting and supports non-linear relationships in data. However, it has limited interpretability, as understanding individual predictions can be challenging. Additionally, CatBoost can have high memory consumption for large datasets.

For unsupervised learning models we had, Local Outlier Factor (LOF) model, which is highly effective at anomaly detection, since it considers the local density of data points. It adapts well to different types of datasets and handles high dimensional data. Disadvantages of the model are the sensitivity to the number of neighbors, difficulty tuning the model and computational complexity.

Isolation Forest is scalable, making it efficient for working with large, high dimensional data without needing significant pre-processing. Furthermore, is robust to noise, as it isolates anomalies by splitting data on random features. Regardless, it requires careful hyperparameter tuning and uses a randomness factor during tree-building, which can lead to inconsistent results.

Principal Component Analysis (PCA) is straightforward and fast model. It focuses on capturing the most variance, thus filtering out the noise. However, while reducing dimensionality, PCA might lose valuable information and assumes linearity in the data, which does not hold true for complex datasets.

DBSCAN is a clustering model where there is no need to predefine the number of clusters. It can detect clusters of different shapes and densities. However, parameter sensitivity is a disadvantage. DBSCAN needs careful parameter tuning for eps and minPts. Furthermore, the model is slow with large datasets due to being computationally expensive.

Copula Based Outlier Detection (COPOD) is a flexible model that adapts well to different data types and distributions, making it highly effective for high-dimensional data. On the other hand, since the model is a combination of methods is more vulnerable to overfitting.

Empirical Cumulative Distribution Function (ECOD) combines multiple anomaly detection techniques, leverages their strengths. It is robust to noise and has good generalization on unseen data, reducing the risk of overfitting. Nevertheless, the performance is heavily dependent on the performance of the base model.

Following the results of our experiments, professionals dealing with anomaly detection in financial data should prioritize models like XGBoost, CatBoost and Random Forest Classifiers. These models offer a balance between accuracy and computational time, making them suitable for time sensitive and high stakes results, often needed in the financial industry. In addition, the models are powerful enough to handle the high demands of the financial sector, where the accurate and fast anomaly detection can prevent potential losses.

Nevertheless, other models, such as DSBCAN and KNN, can be helpful for specific tasks in the financial context. DBSCAN as a clustering algorithm can be valuable for identifying outliers in unstructured datasets, whereas KNN has a more intuitive approach that can be used when dealing with smaller datasets or when high intepretability is required. Both models provide satisfactory performance and, based on the task, can be applied for anomaly detection in finance. Financial professionals must keep in mind the specific characteristics of each anomaly detection task, including dataset size, dimensionality, intepretability, etc., before choosing the right model for their task.

Anomaly detection is especially important in finance due to the potential for direct

financial losses and reputational damage to a business as a result of undetected anomalies. Compared to other sectors, where anomalies indicate minor system issues or process inefficiency, financial anomalies are often indicators of financial fraud, risks, market instability or compliance failures. An effective anomaly detection strategy in finance protects the clients and supports the companies on their objectives for fraud prevention and risk management. Thus, making it an invaluable component for maintaining stability of the financial sector.

5.1. Limitations & Future work

The study provides valuable insights into anomaly detection detection for financial services, however there are limitations to the work, that can be challenged in the future to better understand of the topic. The study takes into consideration only twelve machine learning models, ranging from typical classification model, clustering algorithms and newer models. There are still quite a few number of models that could be tested and compared. Future work, can delve deeper into those models, as well as test more advanced deep learning algorithms. Additionally, newer studies can test the models on larger and diverse financial datasets, as well as on powerful machines that better handle the computational power.

Bibliography

[1] Aderemi Adewumi and Ayo Akinyelu. A survey of machine-learning and nature-inspired based credit card fraud detection techniques. *International Journal of System Assurance Engineering and Management*, 8, 12 2016.

[2] Charu C. Aggarwal. *An Introduction to Outlier Analysis*. Springler, 2017.

[3] Archana Anandakrishnan, Senthil Kumar, Alexander Statnikov, Tanveer Faruquie, and Di Xu. *Anomaly Detection in Finance: Editors' Introduction*, volume 71 of *Proceedings of Machine Learning Research*. PMLR, 14 Aug 2018.

[4] Alexander Bakumenko and Ahmed Elragal. Detecting anomalies in financial data using machine learning algorithms. *Systems*, 10:1–29, 08 2022.

[5] Vic Barnett and Toby Lewis. *Outliers in Statistical Data, 3rd Edition*. John Wiley and Sons, 1994.

[6] Monowar H. Bhuyan, D. K. Bhattacharyya, and J. K. Kalita. Network anomaly detection: Methods, systems and tools. *IEEE Communications Surveys & Tutorials*, 16(1):303–336, 2014.

[7] Giuseppe Bonaccorso. *Machine Learning Algorithms: A reference guide to popular algorithms for data science and machine learning*. Packt Publishing, 2017.

[8] Azzedine Boukerche, Lining Zheng, and Omar Alfandi. Outlier detection: Methods, models, and classification. *ACM Comput. Surv.*, 53(3), jun 2020.

[9] Azzedine Boukerche, Lining Zheng, and Omar Alfandi. Outlier detection: Methods, models, and classification. *ACM Comput. Surv.*, 53(3), jun 2020.

[10] Leo Breiman. Random forests. *Mach. Learn.*, 45(1):5–32, oct 2001.

[11] Markus M. Breunig, Hans-Peter Kriegel, Raymond T. Ng, and Jörg Sander. Lof: identifying density-based local outliers. *SIGMOD Rec.*, 29(2):93–104, may 2000.

[12] Anna L. Buczak and Erhan Guven. A survey of data mining and machine learning methods for cyber security intrusion detection. *IEEE Communications Surveys & Tutorials*, 18(2):1153–1176, 2016.

[13] Varun Chandola, Arindam Banerjee, and Vipin Kumar. Anomaly detection: A survey. *ACM Comput. Surv.*, 41(3), jul 2009.

[14] Varun Chandola, Arindam Banerjee, and Vipin Kumar. Anomaly detection: A survey. *ACM Comput. Surv.*, 41(3), jul 2009.

[15] Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, page 785–794, New York, NY, USA, 2016. Association for Computing Machinery.

[16] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Mach. Learn.*, 20(3):273–297, sep 1995.

[17] T. Cover and P. Hart. Nearest neighbor pattern classification. *IEEE Transactions on Information Theory*, 13(1):21–27, 1967.

[18] Linda Delamaire, Hussein Abdou, and John Pointon. Credit card fraud and detection techniques: A review. *Banks and Bank Systems*, 4, 01 2009.

[19] Pedro M. Domingos and Michael J. Pazzani. On the optimality of the simple bayesian classifier under zero-one loss. *Machine Learning*, 29:103–130, 1997.

[20] Remi Domingues, Maurizio Filippone, and Jihane Zouaoui. A comparative evaluation of outlier detection algorithms: Experiments and analyses. *Pattern Recognition*, 74, 09 2017.

[21] Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, KDD'96, page 226–231. AAAI Press, 1996.

[22] Tom Fawcett. An introduction to roc analysis. *Pattern Recogn. Lett.*, 27(8):861–874, jun 2006.

[23] Zao-Bin Gan, Deng-Wen Wei, and V. Varadharajan. Evaluating the performance and scalability of web application systems. In *Third International Conference on Information Technology and Applications (ICITA'05)*, volume 1, pages 111–114 vol.1, 2005.

[24] Prasanta Gogoi, D.K. Bhattacharyya, B. Borah, and Jugal K. Kalita. A survey of outlier detection methods in network anomaly identification. *The Computer Journal*, 54(4):570–588, 2011.

[25] Jordi Guitart, V. Beltran, David Carrera, Jeison Torres, and E. Ayguade. Characterizing secure dynamic web applications scalability. volume 2005, pages 108a– 108a, 04 2005.

[26] Manish Gupta, Jing Gao, Charu C. Aggarwal, and Jiawei Han. Outlier detection for temporal data: A survey. *IEEE Transactions on Knowledge and Data Engineering*, 26(9):2250–2267, 2014.

[27] Songqiao Han, Xiyang Hu, Hailiang Huang, Mingqi Jiang, and Yue Zhao. Adbench: Anomaly detection benchmark, 2022.

[28] D. Hawkins. *Identification of Outliers*. Springer, 1980.

[29] Shilin He, Jieming Zhu, Pinjia He, and Michael R. Lyu. Experience report: System log analysis for anomaly detection. *2016 IEEE 27th International Symposium on Software Reliability Engineering (ISSRE)*, pages 207–218, 2016.

[30] Richard Herring and Anthony Santomero. The role of the financial sector in economic performance. 02 1970.

[31] Waleed Hilal, S. Gadsden, and John Yawney. Financial fraud: A review of anomaly detection techniques and recent advances. *Expert Systems with Applications*, 193:116429, 12 2021.

[32] Waleed Hilal, S. Andrew Gadsden, and John Yawney. Financial fraud: A review of anomaly detection techniques and recent advances. *Expert Systems with Applications*, 193:116429, 2022.

[33] Waleed Hilal, S. Andrew Gadsden, and John Yawney. Financial fraud: A review of anomaly detection techniques and recent advances. *Expert Systems with Applications*, 193:116429, 2022.

[34] Victoria Hodge and Jim Austin. A survey of outlier detection methodologies. *Artificial intelligence review*, 22(2):85–126, 2004.

[35] Olumuyiwa Ibidunmoye, Francisco Hernández-Rodriguez, and Erik Elmroth. Performance anomaly detection and bottleneck identification. *ACM Comput. Surv.*, 48(1), jul 2015.

[36] Richard W. Johnson. *Applied Multivariate Statistical Analysis*. Prentice Hall, Englewood Cliffs, 1992.

[37] Yetong Li. *Anomaly detection in Financial Data*, pages 419–426. 10 2023.

[38] Zheng Li, Yue Zhao, Nicola Botta, Cezar Ionescu, and Xiyang Hu. Copod: Copula-based outlier detection. In *2020 IEEE International Conference on Data Mining (ICDM)*, pages 1118–1123, 2020.

[39] Zheng Li, Yue Zhao, Xiyang Hu, Nicola Botta, Cezar Ionescu, and George H. Chen. Ecod: Unsupervised outlier detection using empirical cumulative distribution functions. *IEEE Transactions on Knowledge and Data Engineering*, 35(12):12181–12193, 2023.

[40] Jessica Lin, Eamonn Keogh, Ada Fu, and Helga Van Herle. Approximations to magic: Finding unusual medical time series. In *Proceedings of the 18th IEEE Symposium on Computer-Based Medical Systems*, CBMS '05, page 329–334, USA, 2005. IEEE Computer Society.

[41] Fei Tony Liu, Kai Ming Ting, and Zhi-Hua Zhou. Isolation forest. In *2008 Eighth IEEE International Conference on Data Mining*, pages 413–422, 2008.

[42] Edgar Alonso Lopez-Rojas and Stefan Axelsson. Banksim: A bank payment simulation for fraud detection research. 09 2014.

[43] Mehryar Mohri, Afshin Rostamizadeh, and Ameet Talwalkar. *Foundations of Machine Learning*. The MIT Press, 2012.

[44] E.W.T. Ngai, Yong Hu, Y.H. Wong, Yijun Chen, and Xin Sun. The application of data mining techniques in financial fraud detection: A classification framework and an academic review of literature. *Decision Support Systems*, 50:559–569, 02 2011.

[45] Tahereh Pourhabibi, Kok-Leong Ong, Booi H. Kam, and Yee Ling Boo. Fraud detection: A systematic literature review of graph-based anomaly detection approaches. *Decis. Support Syst.*, 133(C), jun 2020.

[46] Liudmila Prokhorenkova, Gleb Gusev, Aleksandr Vorobev, Anna Veronika Dorogush, and Andrey Gulin. Catboost: unbiased boosting with categorical features. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, NIPS'18, page 6639–6649, Red Hook, NY, USA, 2018. Curran Associates Inc.

[47] Payam Refaeilzadeh and Huan Tang, Leiand Liu. *Cross-Validation*, pages 532–538. Springer US, Boston, MA, 2009.

[48] Pankaj Richhariya and Prashant K Singh. A survey on financial fraud detection methodologies. *International Journal of Computer Applications*, 45(22):15–22, May 2012.

[49] Nick F. Ryman-Tubb and Paul Krause. Neural network rule extraction to detect credit card fraud. In Lazaros Iliadis and Chrisina Jayne, editors, *Engineering Applications of Neural Networks*, pages 101–110, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg.

[50] Takaya Saito and Marc Rehmsmeier. The precision-recall plot is more informative than the roc plot when evaluating binary classifiers on imbalanced datasets. *PloS one*, 10(3):e0118432, 2015.

[51] Falaki Samuel oluwole, Boniface Alese, Olumide Adewale, J Ayeni, Ganiyu Aderounmu, and Ismaila W.O. Probabilistic credit card fraud detection system in online transactions. *International Journal of Software Engineering and its Applications*, 6, 01 2015.

[52] Nils Schwenzfeier and Volker Gruhn. Towards a practical process model for anomaly detection systems. In *Proceedings of the 1st International Workshop on Software Engineering for Cognitive Services*, SE4COG '18, page 41–44, New York, NY, USA, 2018. Association for Computing Machinery.

[53] Colin Shearer. The crisp-dm model: the new blueprint for data mining. *Journal of data warehousing*, 5(4):13–22, 2000.

[54] Mei-Ling Shyu, Shu-Ching Chen, Kanoksri Sarinnapakorn, and Liwu Chang. A novel anomaly detection scheme based on principal component classifier. 01 2003.

[55] Karanjit Singh and Shuchita Upadhyaya. Outlier detection: Applications and techniques. *International Journal of Computer Science Issues*, 9, 01 2012.

[56] Angela Sodemann, Matthew Ross, and Brett Borghetti. A review of anomaly detection in automated surveillance. *IEEE transactions on systems, man, and cybernetics, part C*, 42, 01 2011.

[57] Marina Sokolova, Nathalie Japkowicz, and Stan Szpakowicz. Beyond accuracy, f-score and roc: a family of discriminant measures for performance evaluation. In *Proceedings of the 19th Australian Joint Conference on Artificial Intelligence: Advances in Artificial Intelligence*, AI'06, page 1015–1021, Berlin, Heidelberg, 2006. Springer-Verlag.

[58] Pang-Ning Tan, Michael Steinbach, Anuj Karpatne, and Vipin Kumar. *Introduction to Data Mining (2nd Edition)*. Pearson, 2nd edition, 2018.

[59] Vanessa Tchamyou and Simplice Asongu. Information sharing and financial sector development in africa. *Journal of African Business*, 18(1):24–49, 2017.

[60] Chih-Fong Tsai, Yu-Feng Hsu, Chia-Ying Lin, and Wei-Yang Lin. Intrusion detection by machine learning: A review. *Expert Systems with Applications*, 36(10):11994–12000, 2009.

[61] Li Yang and Abdallah Shami. On hyperparameter optimization of machine learning algorithms: Theory and practice. *Neurocomputing*, 415:295–316, 2020.

[62] Masoumeh Zareapoor, Seeja K.R., and Afshar Alam. Analysis on credit card fraud detection techniques: Based on certain design criteria. *International Journal of Computer Applications*, 52:35–42, 08 2012.

[63] Yue Zhao, Zain Nasrullah, and Zheng Li. Pyod: A python toolbox for scalable outlier detection. *Journal of Machine Learning Research*, 20(96):1–7, 2019.

[64] Arthur Zimek, Erich Schubert, and Hans-Peter Kriegel. A survey on unsupervised outlier detection in high-dimensional numerical data. *Statistical Analysis and Data Mining: The ASA Data Science Journal*, 5(5):363–387, 2012.

[65] Renguang Zuo. Machine learning of mineralization-related geochemical anomalies: A review of potential methods. *Natural Resources Research*, 26:1–8, 05 2017.

A. Appendix

Table A.1.: Distance Validation Performance Summary - Dataset 1 (Full Results)

Cluster Index	**Anomaly Distance (Mean)**	**Point Distance (Mean)**	**Ratio**	**Anomaly Distance (Min)**	**Anomaly Distance (Max)**
0	5.046	1.393	3.621	1.141	180.185
1	7.010	1.353	5.182	1.259	180.366
2	6.325	1.335	4.737	0.666	180.373
3	5.383	1.302	4.135	1.206	180.167
4	5.606	1.399	4.009	0.419	180.362
5	5.520	1.395	3.957	1.001	180.208
6	5.519	1.376	4.010	1.170	180.348
7	5.320	1.356	3.924	0.963	180.335
8	5.240	1.360	3.851	0.922	180.205
9	5.389	1.350	3.992	1.128	180.371
10	5.639	1.352	4.171	1.000	180.320
11	6.190	1.361	4.548	1.126	180.409
12	6.705	1.349	4.970	0.867	180.345
13	5.810	1.371	4.237	0.924	180.325
14	5.701	1.380	4.130	0.978	180.333
15	5.405	1.358	3.979	1.227	180.310
16	6.651	1.335	4.984	1.094	180.425
17	6.584	1.358	4.846	1.184	180.420
18	5.257	1.354	3.882	1.107	180.316
19	5.419	1.404	3.859	1.100	180.170
20	5.719	1.364	4.192	1.111	180.329
21	6.673	1.349	4.946	0.783	180.348
22	5.523	1.362	4.054	1.132	180.330
23	5.269	1.346	3.915	1.224	180.347
24	6.125	1.343	4.562	1.254	180.331
25	10.547	1.392	7.576	1.008	180.538
26	5.764	1.350	4.270	0.667	180.309
27	6.675	1.361	4.904	0.618	180.326
28	6.203	1.358	4.568	1.095	180.324
29	5.780	1.406	4.112	1.041	180.216
30	5.981	1.393	4.292	1.111	180.330
31	5.270	1.309	4.025	1.302	180.169
32	5.722	1.409	4.062	1.014	180.217
33	5.719	1.410	4.056	0.870	180.442
34	5.592	1.383	4.042	0.784	180.222
35	5.411	1.364	3.967	1.166	180.347
36	5.647	1.112	5.079	0.828	180.404
37	5.983	1.341	4.461	1.136	180.327
38	5.555	1.357	4.093	1.010	180.307
39	6.461	1.328	4.866	0.818	180.399
40	5.726	1.365	4.195	0.924	180.412
41	7.186	1.348	5.331	0.931	180.388
42	5.072	1.361	3.727	0.802	180.317
43	6.090	1.361	4.473	1.082	180.404
44	5.474	1.339	4.088	1.125	180.328
45	5.610	1.352	4.149	0.951	180.356
46	5.501	1.310	4.199	1.280	180.144
47	6.296	0.364	17.279	0.906	179.372
48	5.555	1.410	3.941	0.585	180.233
49	6.694	1.312	5.104	1.941	180.164
50	5.609	1.340	4.186	0.983	180.403

Table A.2.: Distance Validation Performance Summary - Dataset 1 (Full Results)

Cluster Index	**Anomaly Distance (Mean)**	**Point Distance (Mean)**	**Ratio**	**Anomaly Distance (Min)**	**Anomaly Distance (Max)**
51	5.487	1.301	4.218	1.549	180.140
52	5.763	1.347	4.279	0.911	180.333
53	5.668	1.353	4.187	1.028	180.343
54	5.312	1.368	3.882	0.697	180.343
55	6.171	1.397	4.416	1.131	180.464
56	6.540	1.358	4.814	1.389	180.356
57	7.203	1.343	5.363	1.087	180.361
58	5.720	1.330	4.302	1.415	180.126
59	6.300	1.301	4.843	1.496	180.164
60	5.476	1.351	4.054	1.017	180.308
61	6.670	1.358	4.911	1.236	180.383
62	6.536	1.349	4.845	1.113	180.423
63	5.895	1.403	4.203	1.445	180.207
64	5.359	1.361	3.938	0.824	180.342
65	5.869	1.310	4.479	1.479	180.147
66	5.438	1.331	4.086	0.892	180.162
67	5.751	1.354	4.247	1.081	180.391
68	5.841	1.355	4.310	0.969	180.396
69	5.472	1.325	4.129	1.078	180.360
70	5.297	1.357	3.905	0.931	180.317
71	5.341	1.374	3.888	0.832	180.363
72	5.998	1.404	4.272	0.881	180.433
73	5.244	1.358	3.862	1.069	180.208
74	5.567	1.353	4.114	0.958	180.342
75	5.866	1.345	4.362	1.028	180.385
76	5.383	1.378	3.907	1.119	180.196
77	5.512	1.360	4.054	1.077	180.377
78	5.558	1.382	4.023	0.513	180.353
79	6.478	1.337	4.846	1.044	180.384
80	5.280	1.355	3.897	0.869	180.374
81	5.994	1.350	4.439	0.703	180.320
82	5.383	1.360	3.957	1.039	180.372
83	5.478	1.366	4.010	0.929	180.332
84	5.618	1.407	3.994	1.109	180.169
85	6.139	1.383	4.439	0.727	180.247
86	5.709	1.327	4.302	1.008	180.378
87	5.372	1.321	4.065	1.242	180.163
88	5.557	1.304	4.261	1.242	180.163
89	5.833	1.362	4.283	1.134	180.343
90	5.631	1.261	4.464	0.837	180.243
91	7.731	1.371	5.641	1.027	180.418
92	5.722	1.354	4.227	1.097	180.345
93	6.592	1.331	4.952	0.868	180.410
94	5.300	1.383	3.831	1.022	180.318
95	6.015	1.411	4.261	0.249	180.374
96	5.541	1.356	4.087	0.989	180.279
97	5.870	1.377	4.263	0.893	180.313
98	5.524	1.357	4.071	1.112	180.348
99	5.753	1.364	4.218	0.662	180.384
100	5.364	1.370	3.916	1.033	180.383
101	5.588	1.355	4.125	0.819	180.330
102	5.503	1.416	3.887	1.034	180.352
103	5.283	1.369	3.859	0.958	180.362
104	5.942	1.342	4.429	0.989	180.291
105	5.685	1.360	4.181	1.027	180.347
106	5.910	0.994	5.947	0.925	180.243
107	5.393	1.348	4.001	0.769	180.376
108	5.544	1.390	3.988	0.853	180.430
109	6.677	1.359	4.915	0.963	180.324
110	6.479	1.359	4.767	0.911	180.351
111	6.000	1.309	4.584	0.580	180.361
112	5.761	1.014	5.683	0.822	180.226
113	5.498	1.350	4.071	1.194	180.367
114	5.569	1.371	4.063	0.481	180.304
115	5.433	1.352	4.020	0.900	180.409

Table A.3.: Distance Validation Performance Summary - Dataset 1 (Full Results)

Cluster Index	Anomaly Distance (Mean)	Point Distance (Mean)	Ratio	Anomaly Distance (Min)	Anomaly Distance (Max)
116	5.713	1.196	4.775	0.677	180.389
117	5.658	1.361	4.159	0.863	180.325
118	5.736	1.347	4.259	0.916	180.356
119	5.663	1.359	4.168	1.187	180.389
120	5.101	1.397	3.652	0.882	180.156
121	6.009	1.322	4.547	1.144	180.193
122	5.857	1.380	4.245	0.805	180.356
123	5.407	1.415	3.821	1.063	180.382
124	7.457	1.405	5.308	1.032	180.257
125	5.162	1.441	3.582	0.879	180.216
126	6.100	1.371	4.448	1.190	180.371
127	5.483	1.332	4.116	1.162	180.340
128	5.545	1.376	4.029	1.064	180.219
129	5.767	1.340	4.305	0.883	180.358
130	6.532	1.386	4.714	1.091	180.206
131	5.305	1.372	3.867	1.270	180.321
132	5.713	1.356	4.213	0.958	180.341
133	5.769	1.355	4.258	0.920	180.411
134	5.787	1.391	4.159	1.015	180.184
135	5.486	1.320	4.156	0.791	180.356
136	5.553	1.393	3.987	0.992	180.338
137	5.672	1.364	4.158	1.048	180.338
138	6.391	0.958	6.673	1.065	180.244
139	6.717	1.369	4.907	0.825	180.380
140	5.538	1.333	4.153	0.875	180.370
141	5.570	1.365	4.081	1.121	180.317
142	5.270	1.392	3.786	1.083	180.183
143	6.353	1.338	4.749	0.993	180.333
144	5.612	1.364	4.116	1.022	180.313
145	5.866	1.343	4.367	1.207	180.133
146	6.111	1.376	4.440	0.540	180.365
147	5.249	1.346	3.900	1.242	180.361
148	6.681	1.355	4.931	1.086	180.244
149	5.431	1.319	4.117	1.239	180.146
150	6.494	1.376	4.720	1.121	180.353
151	5.223	1.222	4.275	0.924	180.257
152	5.300	1.350	3.927	1.085	180.369
153	5.601	1.406	3.985	0.910	180.443
154	5.825	1.384	4.208	1.076	180.327
155	6.723	1.348	4.986	0.914	180.366
156	5.698	1.414	4.030	0.491	180.359
157	6.599	1.368	4.822	0.799	180.394
158	5.442	1.393	3.907	0.980	180.304
159	5.503	1.360	4.047	0.981	180.334
160	6.413	1.421	4.514	1.034	180.406
161	5.920	1.366	4.334	0.917	180.370
162	5.076	1.400	3.624	1.099	180.179
163	7.369	1.400	5.265	1.417	180.243
164	6.207	1.392	4.459	1.298	180.185
165	5.504	1.373	4.010	0.431	180.333
166	6.620	1.371	4.828	1.108	180.380
167	5.607	1.376	4.074	0.892	180.299
168	5.886	1.344	4.380	0.844	180.349
169	7.861	1.405	5.597	1.167	180.505

Table A.4.: Distance Validation Performance Summary - Dataset 1 (Full Results)

Cluster Index	**Anomaly Distance (Mean)**	**Point Distance (Mean)**	**Ratio**	**Anomaly Distance (Min)**	**Anomaly Distance (Max)**
170	6.000	1.312	4.573	1.777	180.166
171	6.711	1.350	4.970	0.941	180.326
172	5.323	1.350	3.945	0.914	180.346
173	5.491	1.360	4.038	1.098	180.304
174	7.329	1.346	5.446	0.922	180.371
175	5.459	1.372	3.979	0.865	180.319
176	5.195	1.348	3.852	0.459	180.321
177	5.536	1.354	4.088	0.699	180.326
178	5.468	1.371	3.988	0.560	180.354
179	6.671	1.374	4.855	1.302	180.395
180	7.334	1.371	5.348	1.029	180.346
181	5.127	1.375	3.729	0.781	180.317
182	5.200	1.356	3.834	1.205	180.362
183	5.451	1.351	4.035	1.143	180.356
184	5.497	1.395	3.942	1.211	180.194
185	5.528	1.251	4.420	0.882	180.209
186	5.118	1.303	3.928	1.076	180.141
187	6.567	1.357	4.840	1.271	180.367
188	7.190	1.383	5.200	1.500	180.364
189	6.309	1.394	4.526	1.299	180.214
190	5.923	1.377	4.303	1.267	180.312
191	6.477	1.371	4.724	0.960	180.310
192	6.161	1.366	4.514	1.058	180.329
193	5.757	1.359	4.239	1.142	180.354
194	5.855	1.382	4.241	0.990	180.381
195	6.603	1.318	4.924	1.257	180.266
196	6.617	1.368	4.839	1.072	180.383
197	6.255	1.358	4.625	1.025	180.409
198	6.403	1.357	4.718	1.038	180.313
199	6.011	1.386	4.335	0.965	180.326
200	6.725	1.360	4.943	0.924	180.287

Table A.5.: Distance Validation Performance Summary - Dataset 2 (Full Results)

Cluster Index	**Anomaly Distance (Mean)**	**Point Distance (Mean)**	**Ratio**	**Anomaly Distance (Min)**	**Anomaly Distance (Max)**
0	8.871	0.888	9.994	1.004	316.734
1	8.938	0.817	10.947	0.756	316.757
2	8.891	0.976	9.105	0.489	316.775
3	8.514	1.359	6.263	1.077	316.677
4	8.855	0.737	12.013	0.715	316.773
5	9.124	0.815	11.189	0.586	316.917
6	8.804	0.885	9.945	0.578	316.521
7	8.651	1.037	8.339	0.741	316.431
8	8.415	1.077	7.816	0.741	316.255
9	8.772	0.668	13.137	0.604	316.260
10	8.630	0.724	11.916	0.743	316.268
11	8.391	1.023	8.203	0.811	316.256
12	8.367	1.004	8.337	0.658	316.266
13	8.404	1.093	7.688	0.639	316.269
14	8.689	0.612	14.203	0.925	316.243
15	8.763	0.693	12.653	0.501	316.449
16	8.420	0.917	9.179	0.651	316.272
17	8.526	0.773	11.030	0.998	316.275
18	8.433	1.046	8.061	0.897	316.262
19	8.464	0.905	9.351	0.904	316.249
20	9.192	0.501	18.360	0.633	316.267
21	9.267	0.544	17.036	0.639	316.463
22	8.894	0.593	14.988	0.879	316.241
23	8.921	0.651	13.702	0.610	316.250
24	8.484	0.742	11.434	0.553	316.235
25	8.714	0.804	10.839	1.048	316.238
26	9.165	0.408	22.463	1.332	115.181
27	8.400	0.142	59.082	0.905	57.256
28	8.081	0.627	12.889	1.037	34.206

Table A.6.: Distance Validation Performance Summary - Dataset 3 (Full Results 1/4)

Cluster Index	**Anomaly Distance (Mean)**	**Point Distance (Mean)**	**Ratio**	**Anomaly Distance (Min)**	**Anomaly Distance (Max)**
0	4.362	1.356	3.218	0.811	74.584
1	4.392	1.304	3.367	0.961	74.566
2	4.301	1.233	3.489	0.826	74.636
3	4.309	1.359	3.172	0.806	74.571
4	4.545	1.351	3.364	0.864	74.606
5	4.253	1.343	3.167	0.954	74.599
6	4.553	1.338	3.403	0.808	74.598
7	4.299	1.292	3.328	0.883	74.606
8	4.483	1.293	3.468	0.837	74.625
9	4.599	1.267	3.629	0.830	74.602
10	4.536	1.278	3.550	0.812	74.629
11	4.355	1.265	3.443	0.823	74.608
12	4.336	1.344	3.227	0.941	74.595
13	4.332	1.279	3.387	0.722	74.618
14	4.246	1.280	3.317	0.775	74.622
15	4.782	1.267	3.776	0.774	74.676
16	4.832	1.308	3.694	0.796	74.627
17	4.771	1.296	3.682	0.808	74.568
18	4.481	1.261	3.554	0.793	74.653
19	4.785	1.351	3.541	0.901	74.653
20	4.387	1.286	3.413	0.795	74.593
21	4.838	1.277	3.787	0.946	74.655
22	4.604	1.333	3.454	0.780	74.573
23	4.304	1.305	3.299	0.823	74.593
24	4.801	1.301	3.690	0.822	74.564
25	4.693	1.290	3.638	0.805	74.570
26	4.499	1.344	3.348	0.768	74.484
27	4.748	1.241	3.826	0.861	74.590
28	4.826	1.303	3.703	0.834	74.547
29	4.794	1.367	3.507	0.787	74.696
30	4.811	1.354	3.553	0.695	74.697
31	4.548	1.265	3.595	0.791	74.622
32	4.895	0.746	6.567	0.748	74.605
33	4.754	1.308	3.635	1.002	74.554
34	5.536	0.483	11.466	1.379	74.670
35	4.447	1.248	3.562	0.646	74.512
36	4.440	1.361	3.263	0.689	74.501
37	4.420	1.270	3.481	0.403	74.540
38	5.043	0.440	11.473	0.931	74.612
39	4.391	1.249	3.515	0.522	74.541
40	6.259	0.229	27.311	1.003	74.823
41	4.871	1.262	3.860	0.761	74.694
42	6.153	0.213	28.942	1.163	74.894
43	4.936	0.870	5.677	1.073	74.648
44	4.759	1.197	3.976	0.725	74.725

Table A.7.: Distance Validation Performance Summary - Dataset 3 (Full Results 2/4)

Cluster Index	Anomaly Distance (Mean)	Point Distance (Mean)	Ratio	Anomaly Distance (Min)	Anomaly Distance (Max)
45	4.789	1.232	3.887	0.469	74.739
46	4.665	0.780	5.978	0.549	74.570
47	4.970	1.277	3.891	0.756	74.573
48	4.913	1.283	3.828	0.821	74.602
49	5.106	0.881	5.793	1.159	74.680
50	4.569	1.374	3.325	0.640	74.402
51	4.412	0.354	12.465	0.510	74.385
52	5.018	0.477	10.523	0.664	74.747
53	4.415	1.058	4.171	0.640	74.541
54	4.480	0.843	5.305	0.813	74.637
55	4.869	1.076	4.519	0.648	74.620
56	4.459	0.871	5.126	0.693	74.537
57	4.555	1.207	3.780	0.654	74.606
58	4.751	1.209	3.927	0.546	74.637
59	4.620	0.988	4.676	0.650	74.669
60	5.049	0.328	15.377	0.446	74.630
61	5.239	0.560	9.351	0.632	74.789
62	4.814	0.525	9.164	0.499	74.536
63	4.508	1.242	3.628	0.597	74.424
64	5.270	0.595	8.864	0.718	74.796
65	5.274	0.425	12.415	0.567	74.680
66	4.966	1.290	3.849	0.768	74.576
67	5.090	0.324	15.688	0.504	74.616
68	5.022	1.301	3.860	1.057	74.553
69	4.447	0.522	8.518	0.515	74.410
70	4.823	0.327	14.729	0.451	74.648
71	5.264	0.951	5.535	0.661	74.656
72	4.675	0.417	11.200	0.448	74.686
73	5.671	0.845	6.710	1.271	74.615
74	5.046	0.482	10.467	0.760	74.597
75	4.921	0.397	12.388	0.501	74.607
76	5.884	0.751	7.837	1.215	74.782
77	4.926	0.330	14.914	0.580	74.569
78	4.800	0.362	13.245	0.452	74.516
79	6.059	0.778	7.783	1.102	74.792
80	4.852	0.345	14.070	0.354	74.484
81	4.568	0.368	12.421	0.444	74.635
82	6.014	0.726	8.282	0.980	74.799
83	4.940	0.395	12.500	0.525	74.566
84	5.119	0.391	13.088	0.582	74.713
85	5.212	0.853	6.111	1.012	74.628
86	4.267	0.310	13.766	0.373	73.975
87	4.772	1.237	3.859	0.825	74.373
88	4.399	0.306	14.368	0.423	74.647
89	4.565	0.994	4.594	0.359	74.490
90	5.479	0.594	9.221	0.856	74.691
91	5.141	0.306	16.791	0.442	74.536
92	5.049	0.420	12.013	0.518	74.725
93	4.919	0.656	7.495	1.055	74.631
94	4.494	0.639	7.037	0.523	74.527
95	4.809	0.315	15.264	0.348	74.662
96	4.905	0.415	11.829	0.583	74.548
97	6.084	0.728	8.352	1.178	74.827
98	5.276	0.411	12.832	0.494	74.656
99	4.305	0.353	12.194	0.560	74.589
100	4.641	0.302	15.343	0.409	74.493

Table A.8.: Distance Validation Performance Summary - Dataset 3 (Full Results 3/4)

Cluster Index	Anomaly Distance (Mean)	Point Distance (Mean)	Ratio	Anomaly Distance (Min)	Anomaly Distance (Max)
101	5.058	0.358	14.115	0.502	74.490
102	4.885	0.307	15.932	0.446	74.383
103	4.886	0.340	14.365	0.507	73.740
104	4.850	0.311	15.606	0.408	74.280
105	5.021	0.211	23.793	0.413	74.200
106	4.740	0.910	5.209	0.833	74.468
107	4.640	0.254	18.305	0.376	74.507
108	5.044	0.279	18.074	0.371	74.690
109	4.911	0.315	15.596	0.428	74.893
110	4.840	0.311	15.545	0.312	74.727
111	5.931	0.600	9.882	1.394	74.791
112	4.796	0.299	16.028	0.469	74.540
113	4.961	0.294	16.893	0.379	74.806
114	5.148	0.548	9.397	0.933	74.579
115	5.343	0.966	5.534	0.748	74.603
116	4.806	0.357	13.480	0.485	74.784
117	4.790	0.330	14.524	0.419	74.316
118	4.715	0.332	14.201	0.465	74.511
119	5.518	0.525	10.513	1.262	74.605
120	5.152	0.339	15.182	1.000	74.579
121	4.685	0.341	13.732	0.579	74.630
122	4.684	0.294	15.935	0.487	73.594
123	4.627	0.296	15.644	0.448	73.870
124	5.112	0.370	13.826	0.651	74.743
125	4.828	0.377	12.806	0.486	73.696
126	4.659	0.276	16.868	0.455	73.494
127	4.482	0.405	11.054	0.486	73.476
128	4.908	0.320	15.332	0.436	74.224
129	5.110	0.408	12.505	0.676	74.628
130	5.294	0.311	17.007	0.586	74.706
131	5.035	0.308	16.304	0.444	74.586
132	5.107	0.480	10.630	0.746	74.557
133	5.307	0.336	15.743	0.524	74.792
134	4.864	0.262	18.564	0.423	73.884
135	4.610	0.344	13.402	0.519	73.442
136	5.495	0.496	11.061	1.053	74.698
137	4.678	0.264	17.712	0.493	73.650
138	4.635	0.327	14.194	0.469	73.450
139	5.051	0.369	13.646	0.469	74.313
140	5.353	0.369	14.482	0.516	74.791
141	4.464	0.348	12.804	0.485	73.472
142	4.480	0.257	17.440	0.486	73.412
143	5.225	0.296	17.662	0.495	74.780
144	4.996	0.354	14.115	0.514	74.488
145	5.334	0.393	13.556	0.608	74.750
146	5.307	0.409	12.967	0.648	74.732
147	5.434	0.326	16.640	0.614	74.781
148	4.840	0.278	17.378	0.449	74.694
149	4.753	0.255	18.618	0.408	73.607
150	4.649	0.249	18.678	0.431	73.433

Table A.9.: Distance Validation Performance Summary - Dataset 3 (Full Results 4/4)

Cluster Index	Anomaly Distance (Mean)	Point Distance (Mean)	Ratio	Anomaly Distance (Min)	Anomaly Distance (Max)
151	4.4534	0.4160	10.7042	0.6063	74.2956
152	4.9340	0.3761	13.1183	0.5029	74.6623
153	4.7752	0.3018	15.8223	0.4083	74.5780
154	5.5868	0.3453	16.1818	1.0138	74.5977
155	4.9190	0.5391	9.1244	0.6235	74.2692
156	4.8623	0.5633	8.6314	0.6479	74.5894
157	5.2522	0.2276	23.0751	0.8464	74.5560
158	4.9607	0.3065	16.1830	0.4080	74.3194
159	4.5884	0.2556	17.9490	0.4135	74.5835
160	4.7399	0.4044	11.7221	0.4820	74.5042
161	4.7091	0.2480	18.9855	0.3450	74.5887
162	4.3176	0.3592	12.0208	0.4286	74.6367
163	4.5502	0.2886	15.7680	0.4234	74.4557
164	4.9404	0.2824	17.4943	0.4128	74.4308
165	4.6014	0.4429	10.3896	0.4262	74.6483
166	4.1161	0.2780	14.8040	0.4294	74.2164
167	5.0392	0.2483	20.2980	0.3188	74.6707
168	5.0948	0.3989	12.7710	0.6691	74.7352
169	4.4009	0.4338	10.1448	0.5414	74.4467
170	4.1266	0.3956	10.4314	0.4627	74.4243
171	5.3174	0.2609	20.3804	1.0926	74.6117
172	5.0058	0.2942	17.0151	0.3891	74.5711
173	5.1202	0.5340	9.5882	0.5645	74.1626
174	5.0596	0.1824	27.7395	0.8344	74.5633
175	5.1129	0.2901	17.6236	0.3781	74.3292
176	4.7746	0.3810	12.5309	0.5073	74.5645
177	4.9408	0.3876	12.7472	0.5327	74.5104
178	4.9929	0.4315	11.5717	0.5088	74.4269
179	4.4427	0.3024	14.6935	0.4161	74.4349
180	5.0227	0.3252	15.4428	0.6140	74.7742
181	4.4893	0.3423	13.1133	0.5448	74.5667
182	5.0065	0.2684	18.6537	0.3659	74.6696
183	4.4796	0.3100	14.4521	0.3710	74.4205
184	4.6311	0.3159	14.6587	0.5417	74.6946
185	5.0251	0.2794	17.9836	0.5354	74.8140
186	4.6863	0.3001	15.6147	0.3814	74.2971
187	5.1674	0.1850	27.9341	1.1734	74.5469
188	5.2104	0.2536	20.5455	0.4504	74.7031
189	4.2367	0.2884	14.6918	0.3945	74.4753
190	4.7441	0.2853	16.6312	0.5338	74.5644
191	4.9268	0.2470	19.9436	0.3490	74.0366
192	4.2082	0.3126	13.4634	0.5120	73.9223
193	4.3296	0.3781	11.4506	0.4764	74.5535

Table A.10.: Distance Validation Performance Summary - Dataset 4 (Full Results)

Cluster Index	**Anomaly Distance (Mean)**	**Point Distance (Mean)**	**Ratio**	**Anomaly Distance (Min)**	**Anomaly Distance (Max)**
0	4.323	1.330	3.251	1.653	6.916
1	3.968	1.073	3.699	1.144	6.503
2	4.170	1.057	3.944	1.135	6.634
3	4.327	1.328	3.258	1.772	6.616
4	4.239	0.900	4.710	1.173	7.124
5	3.898	1.057	3.690	1.156	6.305
6	4.249	0.956	4.445	1.210	6.923
7	4.498	1.317	3.416	1.797	6.587
8	4.487	1.317	3.408	1.653	7.195
9	4.426	0.962	4.600	0.782	7.199
10	3.996	0.886	4.510	1.173	6.888
11	4.270	1.358	3.145	1.695	6.746
12	4.147	1.071	3.873	1.085	6.774
13	4.417	0.896	4.930	1.342	6.942
14	4.624	0.906	5.102	0.837	7.572
15	4.183	0.918	4.556	1.091	6.901
16	4.262	0.931	4.577	1.164	6.927
17	4.438	1.116	3.977	1.161	7.355
18	3.975	1.086	3.661	1.181	6.460
19	4.177	1.148	3.638	1.062	7.119
20	4.632	0.590	7.846	0.743	6.905
21	4.350	0.589	7.380	0.930	6.902
22	4.227	0.818	5.168	0.733	6.898
23	4.451	1.112	4.002	1.191	7.102
24	4.317	0.971	4.446	1.315	7.175
25	4.065	0.901	4.514	1.226	6.787
26	4.408	0.932	4.727	1.375	7.224
27	4.175	0.609	6.859	0.766	7.040
28	4.048	0.897	4.515	1.170	6.952
29	4.214	1.145	3.682	1.225	7.082
30	4.166	0.839	4.964	1.030	6.769
31	4.271	1.165	3.666	1.179	7.168
32	4.222	0.835	5.057	1.086	6.848
33	4.361	0.657	6.640	1.009	7.606
34	4.268	0.644	6.624	0.909	7.353
35	4.493	0.642	7.000	0.945	7.215
36	4.329	0.567	7.637	0.941	7.161
37	4.394	0.853	5.155	1.013	7.159
38	4.429	1.076	4.118	0.957	7.340
39	4.479	0.602	7.437	0.991	7.719
40	4.183	0.655	6.388	0.757	7.384
41	4.433	0.623	7.120	1.063	7.066
42	4.396	0.854	5.151	1.024	7.177
43	4.388	0.912	4.809	0.904	7.208
44	4.683	0.650	7.202	0.734	7.650
45	4.624	0.746	6.202	0.633	7.504