DETECTION OF ANOMALOUS FUND TRANSFERS BETWEEN DIFFERENT BANKS

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF INFORMATICS
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

ABDULLAH MERT TUNÇAY

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
MODELLING AND SIMULATION

SEP 2022

Approval of the thesis:

DETECTION OF ANOMALOUS FUND TRANSFERS BETWEEN DIFFERENT BANKS

submitted by **ABDULLAH MERT TUNÇAY** in partial fulfillment of the requirements for the degree of **Master of Science in Modelling and Simulation Department, Middle East Technical University** by,

Prof. Dr. Deniz Zeyrek Bozşahin
Dean, Graduate School of **Informatics** _____

Assoc. Prof. Dr. Elif Sürer
Head of Department, **Modelling and Simulation** _____

Assoc. Prof. Dr. Erdem Akagündüz
Supervisor, **Modelling and Simulation, METU** _____

Examining Committee Members:

Prof. Dr. Alptekin Temizel
Modelling and Simulation, METU _____

Assoc. Prof. Dr. Erdem Akagündüz
Modelling and Simulation, METU _____

Assist. Prof. Dr. İrem Ülkü
Computer Engineering, Ankara University _____

Date: 02.09.2022

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Surname: Abdullah Mert Tunçay

Signature :

ABSTRACT

DETECTION OF ANOMALOUS FUND TRANSFERS BETWEEN DIFFERENT BANKS

Tunçay, Abdullah Mert
M.S., Department of Modelling and Simulation
Supervisor: Assoc. Prof. Dr. Erdem Akagündüz

Sep 2022, 64 pages

There are various risks associated with fund transfers between banks, including frauds and abuse of the banking system. Any anomaly in a fund transfer network must be detected and learned, since unexpected messages or transfer events may occur. The purpose of this thesis is to detect anomalies in a fund transfer network using fund transfer packets. In such a network, different fund transfer protocols use different message types. Moreover, these messages have several different features, such as dates, participants, amounts, time intervals, etc. We utilize various statistical, machine learning and deep learning-based anomaly detection methods such as isolation forests, local outlier factors, k-nearest-neighbour and LSTM Autoencoders to detect anomalies in fund transfers. For this purpose, we collect a set of real-world fund transfer messages and utilize this set in our experiments. It is evident from our results that feature group selection has a significant impact on the accuracy of anomaly detection.

Keywords: Anomaly Detection, Money Transaction, Unsupervised Learning, LSTM

ÖZ

FARKLI BANKALAR ARASI ANORMAL FON TRANSFERLERİNİN TESPİTİ

Tunçay, Abdullah Mert
Yüksek Lisans, Modelleme ve Simülasyon Bölümü
Tez Yöneticisi: Doç. Dr. Erdem Akagündüz

2022 , 64 sayfa

Bankalar arasındaki fon transferleriyle ilgili, dolandırıcılık ve bankacılık sisteminin kötüye kullanılması gibi çeşitli riskler vardır. Beklenmeyen mesajlar veya aktarımlar meydana gelebileceğinden, bu fon transfer ağındaki herhangi bir anormallik tespit edilmeli ve öğrenilmelidir. Bu tezin amacı, fon transfer paketlerindeki anormallikleri tespit etmektir. Farklı fon transfer protokolleri farklı mesaj türleri kullanır. Ayrıca bu mesajların tarihler, katılımcılar, miktarlar, zaman aralıkları vb. gibi birçok farklı özelliği vardır. Biz, fon transferlerindeki anormallikleri tespit etmek için LSTM Otokodlayıcılar, izolasyon ormanları, yerel aykırı değer faktörleri ve k-en yakın komşu gibi çeşitli istatistiksel, makine öğrenimi ve derin öğrenme tabanlı anomali tespit yöntemlerini kullandık. Bu amaçla, bir dizi gerçek dünya fon transferi mesajı topladık ve bu seti deneylerimizde kullandık. Öznitelik grubu seçiminin anomali tespitinin doğruluğu üzerinde önemli bir etkisi olduğu sonuçlarımızdan açıkça görülmektedir.

Anahtar Kelimeler: Anormallik Tespiti, Para Faaliyeti, Denetimsiz Öğrenme, LSTM

To my supervisor, my little nephew Çınar Efe and my family ...

Thank you for all of your support along the way.

ACKNOWLEDGMENTS

My gratitude goes out to Assoc. Prof. Dr. Erdem Akagündüz for his guidance and support. In the course of this thesis and my graduate studies, he was more than just a supervisor to me. Throughout my academic journey, he provided me with unfailing assistance and a motivating approach.

Additionally, I would like to express my gratitude to the thesis jury members for providing me with valuable recommendations and feedback.

For their support of my graduate studies and establishing a healthy and productive professional environment, I would also like to thank the Central Bank of the Republic of Turkey and my superiors, Dr. Ahmet Buğday and Kemal Özgür Duman.

TABLE OF CONTENTS

LIST OF TABLES

TABLES

LIST OF FIGURES

FIGURES

LIST OF ABBREVIATIONS

AD	Anomaly Detection
ADS	Anomaly Detection Systems
AE	Auto-Encoder
AI	Artificial Intelligence
ANN	Artificial Neural Network
AUC	Area Under Curve
CBLOF	Cluster Based Local Outlier Factor
CNN	Convolutional Neural Network
DL	Deep Learning
DIF	Deep Isolation Forest
ECG	Electrocardiagram
EIF	Extended Isolation Forest
GIF	Generalised Isolation Forest
GRU	Gate Recurrent Unit
HBOS	Histogram Based Outlier Score
IF	Isolation Forest
KNN	K Nearest Neighbour
ML	Machine Learning
LOF	Local Outlier Factor
LSTM	Long Short-Term Memory
RNN	Recurrent Neural Network
ROC	Receiver Operator Characteristic

CHAPTER 1

INTRODUCTION

The term anomaly or outlier refers to something that is significantly different, abnormal, or unidentified when compared to a typical distribution. [1] identifies an anomaly as a data point that is significantly different from the rest. Hawkins [2] describes an outlier as follows: *"An outlier is an observation, which deviates so much from the other observations as to arouse suspicions that it was generated by a different mechanism"*. In most cases, the character of the data generated by a system provides insight into its activity. The analysis of these data would enable the identification of any unusual activity. Systems for applying and handling these analyses are called Anomaly Detection Systems (ADS), and the process itself is called Anomaly Detection (AD).

There are three types of anomalies: point anomalies, contextual anomalies, and collective anomalies according to Mohan et al.[3]. A point anomaly is regarded as a different behavior in each discrete data point; contextual anomalies are considered as abnormal behavior expressed in the same context of the data point, and collective anomalies are regarded as occurrences of data points that differ from those in other collections. Analyzing anomaly cases usually considers how much data is available and the features it includes.

There can be different definitions of an anomaly for different fields. For example, in biomedical engineering, according to the characteristics of a heart rate signal, there are various types of arrhythmia that can be identified as anomalies in electrocardiograms[4], [5]. The field of Application Performance Management[6] is another example. Anomaly detection methods are used in these fields to identify performance problems as anomalous[7]. The number of cases that might arise in money transactions varies depending on the environment, the country, the type of currency, and other factors. It can be considered an abnormal behavior for some people to transfer a large amount of money; whereas, it can be normal behavior for others. Hence, defining an anomaly, regardless of the application field usually requires a high-level expert opinion.

Financial transactions involving money are special types of transactions in which the buyer and the seller exchange money. It is possible to classify money transactions into two categories: those that occur between the accounts of the same bank, or between the accounts of different banks. Whenever a buyer and a seller account belong to the same bank, the money transaction is handled by that bank alone. In cases where one of the sides has an account at another bank, the middleware company and the receiver bank come into play, adding two new actors to the process. Senders often send related

information to middleware companies or national institutions, who transfer it to their respective receivers. In this way, the company can also include information related to both ends of the transaction.

In the past, money transfers between different banks took longer to secure compared to money transfers between the accounts of the same bank. This duration is recently reduced to similar levels with the development of new projects [8], resulting in new fraud issues and anomalies. Today, transferred funds can be immediately used after they are transferred. Prior to this, money transactions took longer, and banks had static controls on the amount of money you could transfer, the number of transactions that could take place in a certain amount of time, etc. Some of these controls are eliminated or simplified in the newly developed systems that handle money transactions in less time and make them available to users immediately. This is the reason why banks are sticking to very low fund limits per transaction, but the number of transactions is changing. We believe that all of these factors will inevitably create a research field that is more likely to become feasible by utilizing Artificial Intelligence (AI) based solutions due to the variety of limits, functionalities, and newly represented data.

Different monitoring systems are used by banks in order to create rules and detect anomalies. The responsible company develops applications such as those that enable the creation of rules, email alerts, and control databases in order to detect anomalies. Furthermore, monitoring solutions like AppDynamics, [1], DynaTrace [2] are available. Since they were first introduced, the quality and scope of products have been developed. As a result, they began offering AI-based solutions that evaluate and predict anomalies through the use of models. Depending on the field, the data features, and the anomaly concept, the proposed models vary.

In this thesis, using highly inclusive data sets, we benchmark various learning-based models for anomaly detection in fund transfers between different banks. Using real-life fund transactions without annotations, we introduce a data set for our purposes. The lack of anomaly cases in this set of data led us to specify anomaly scenarios based on features and create a test data set in the response. By analyzing features that characterize the fund transfers, we transform our main data set into additional subsets, in order to utilize large-scale experiments. Additionally, we develop an unsupervised deep learning model using Long Short-Term Memory Auto-Encoders (LSTM-AE) and train it with only normal data so it can learn the normal behavior of the transaction, thereby creating anomaly scores. In our experiments, we also utilize conventional machine learning algorithms to compare our results with our deep learning model.

The bank transfer data is structured data. In order to feed them into deep learning based models, we apply embedding to these features. The process of embedding a feature involves constructing an interpretable format for the network, based on the features in the data. By embedding these variables, any proposed model can save memory and build relationships [9]. Choosing different features for this study may change the amount and relationship of features, and that's a key thing we study in this thesis.

[1] https://www.appdynamics.com/
[2] https://www.dynatrace.com/

1.1 Motivation and Problem Definition

This thesis seeks to identify anomalies in fund transfer networks that are not well described, since the anomaly may differ depending on the perspective. By determining exactly what constitutes an anomaly, in addition to what features and types of algorithms should be used, we will be able to determine whether ensemble algorithms or models are more effective than algorithms or models used separately. Because the banking industry is extremely conservative when it comes to sharing data, the methods used to solve data science problems in this field, such as anomaly detection are usually nonpublic. It is also important to note that the structure of the systems changes depending on the bank. By doing so, it is possible to determine which approach is most appropriate for the particular problem every time a new system is introduced. In analyzing the data of the current system, new analyses and problems could be identified. We would benefit from the introduction of a model that would address these issues.

For an application to be maintainable, monitoring systems are necessary. When an application is deployed into a production environment and begins to interact with users, there will be use cases that have not been considered during development. As a result of these use cases, the system could be vulnerable to vulnerabilities that could affect resource, functionality, and reliance. In order to prevent such situations, there are applications that monitor abnormal behavior.

Based on how to cluster a pattern for a specific type of data, machine learning algorithms such as local outlier factor and random forest work. For an adaptable model for unexpected money transactions, we need to analyze time relations on certain flows more specifically.

A possible outlier in money transaction data is determined by taking into account several features. A certain type of embedding choice needs to be made when feeding these types of data to RNNs. This leads to the training process. Depending on the analysis area chosen, the model can be trained in a variety of ways. The problem and model results could be affected by creating different embedding architectures.

As long as the anomaly definition is clear, different algorithms can be used to identify anomalies. The traditional machine learning algorithms are not providing satisfactory results in terms of accuracy and coverage of anomalies when the anomaly definition is unclear or when there are multiple local anomaly cases in large data sets. Due to the fact that identifying an anomaly in a fund transfer network is crucial, and that the amount of anomalies is unknown, sequence-based identification networks are expected to result in multiple solutions since the number of anomalies in the network is unknown. Further, an ensemble of models trained for different scenarios would allow us to identify and cover more cases while analyzing the normal behavior of the network.

1.2 Research Questions

Given the discussions provided in the previous section, in this thesis, we are searching for answers to the following research questions:

- Can we find anomalies in an inter-bank fund transfer network using machine learning and deep learning methods?
- What are the possible anomaly scenarios for a fund transfer network?
- How are these anomalies correlated based on the type of feature set they use?
- Which machine learning algorithms are suitable for this problem?
- How does each ML-based method perform for a given benchmark?
- Is it possible to get better results with a deep learning-based model?
- Which feature combinations produce the best results?
- Is it possible to obtain a more accurate solution by ensemble methods?

1.3 Contributions and Novelties

There are five main contributions of this thesis:

- The private data set utilized for this thesis is collected only for this thesis study and has never been processed before. Hence, novel anomaly cases are defined based on this set.
- Because anomalies cannot be identified from only one perspective, it is essential to choose several feature combinations when assessing their character. In this thesis, combinations of various features will be analyzed to determine their reliability and their effect on anomaly detection in fund transfers.
- Subsets of the main data set are constructed based on different feature selections. The creation of subsets leads to the generation of anomaly detection cases that are used for training later on, which in turn enable us to examine anomalies from multiple perspectives and combine them accordingly.
- Our experiments utilize the relevant machine and deep learning algorithms and industry-based solutions to detect anomalies, hence, revealing the reliability of recent approaches and their improvement over existing approaches.
- Lastly, a Long Short-Term Memory Auto-encoder based unsupervised approach is applied to the featured data sets and compared with traditional machine learning methods. To the best of our knowledge of the scientific literature, LSTM-Auto-Encoders have not been used previously in an unsupervised fashion to detect anomalies in fund transfer networks.

1.4 The Outline of the Thesis

There are five main parts to the thesis. Preparation of data is the first step. As this is an unused data set, the features and their relationship to the problem must be analyzed to determine whether to remove it or not. Creating a decision tree for finding possible feature combinations for training subsets is the second step. As part of *Chapter* 3, we conduct experiments to address these issues. As we apply various methodologies to these data sets, we will find the advantages and disadvantages of each. Part 4 provides anomaly scores from LSTM auto-encoders based on subsets specified in *Chapter* 3 and represented in the *Table* 3.11. We explain the third and fourth parts of our main work in *Chapter* 4. *Chapter* 5 presents our experiments on the benefits of using the used approach and what kind of improvement it provided over a baseline of machine learning methods. A number of anomaly metrics and possible anomalies are used to record the results. The thesis is finalized with our conclusions in *Chapter* 6.

CHAPTER 2

LITERATURE REVIEW

Anomaly detection systems are embedded systems and architectures designed to detect abnormal activity in streaming data. It is possible to approach this problem in several ways, such as using conventional statistical methods or utilizing neural networks. In this chapter, we will discuss the problem of anomaly detection, the methods used in the field, and the current state-of-the-art that uses deep learning models.

2.1 Conventional Anomaly Detection Methods

Traditionally, several factors are taken into account when determining what approaches to use, including domain experts' threshold-based approach as well as previously trained clusters-based machine learning models. For example, to send e-mails to employees on-call, domain experts either use their own custom applications or industrial applications available with subscriptions. During our research, we analyzed several surveys for the purpose of further anomaly detection analysis. Similar to [10], the methods used can be classified according to their type. The following [11], [12] and [13] can be examined for further reference. The references that have been mentioned contain a number of results obtained by machine learning algorithms for anomaly detection on a variety of test cases.

2.1.1 Domain Expert Methods

An expert in a system is someone who has a lot of knowledge about it. Analyzing the system, discovering its drawbacks, and determining possible limits to apply to it would be the role of these people. Applications exist that allow users to control the behavior of applications and alert certain individuals.

As described in the [14], there are several different applications in the domains to solve different anomaly problems. There are different domains of application for each of these applications. There are several types of applications in these domains, including: Quality control applications [15], Financial applications [16], Weblog analytics[17], Intrusion detection applications [18], Medical applications [19], Text and social media applications [20], Earth science applications [21].

2.1.2 Machine Learning Algorithms

An analysis of a huge amount of data is often required in anomaly detection applied fields. Because these data have a low anomaly percentage, labeling or finding by hand before the training often requires a tremendous amount of work, therefore unsupervised learning algorithms are applied. There are two main approaches: nearest-neighbor and clustering. The reader may refer to [22] and [23] further analysis in evaluating different algorithms .

While finding out the anomalies in a data set, several machine learning approaches provide certain performances and they can be simply categorized as Nearest-Neighbor-based techniques, Clustering-based methods, and Statistical algorithms[24]. There several examples unsupervised anomaly detection algorithms in the literature such as: [24],[25], [26], [27] and [28].

2.1.2.1 Isolation Forest

It has been demonstrated that Zhou et al. [29] propose an approach for anomaly detection literature in which it can beat local outlier factor (LOF) or random forests on the basis of area under the curve (AUC) and processing time. Zhou et al. show that an anomaly is two or more times more likely to be separated from a normal point when this method is applied. A binary tree is formed by randomly choosing a feature until the tree reaches its limit or the nodes have the same value. By computing the path between the root node and the specific node, anomaly scores are calculated. When there are more than 1000 points in the data set, the algorithm performs reasonably well, according to the results.

Jahan et al. address the anomaly detection problem of water fabrication in semiconductor companies in [30]. AUC scores and F-1 scores are used to compare isolation forest algorithms with K-means and LOF.

The study presented in [31] proposes a novel method for detecting anomalies in hyperspectral images. They constructed an initial anomaly map by using a global isolation forest, which was then used to create a second local isolation forest. This would enable them to make use of spatial information. Their results suggest that isolation forest results are superior to other methods in terms of area under the curve, based on their experiments on two real hyperspectral data sets.

An extension of the isolation forest algorithm is proposed by Xu et al. in [32]. The extension focuses on the fact that, until the work of their team, all of the other extensions, as well as the default version, we're still partitioning data in an axis parallel linear fashion, which is ineffective in handling hard anomalies in high-dimensional and non-linear-separable data spaces, and worse still, it leads to an algorithmic bias that assigns unexpectedly high anomaly scores to artifact regions. According to the result, DIF can be used to (i) isolate anomalies more effectively, especially in data with intractable sparsity and non-linearity; (ii) free the isolation process from existing constraints to handle the artifact problem, and (iii) handle a variety of data types in a variety of ways.

As presented in [33], Hariri et al. suggest an extended version of the known isolation forest, which resolves issues associated with the assignment of anomaly scores to data points. In their proposal, heat maps would be used to represent anomaly scores. Several real-world data sets were examined using this approach. By extending the methodology, they were able to produce more reliable and robust anomaly scores, as well as, in some cases, to determine the structure of a dataset with greater accuracy. In comparison to the default isolation forest algorithm on AUC scores, their results did not show a significant improvement.

It is proposed by Lesouple et al. in [34] that an isolated forest without empty branches could be created. They argue that an empty branch in an isolation forest is contrary to the idea since the algorithm for creating branches is based on creating branches until the number of points equals one or until a given maximum depth has been reached. EIF (extended isolation forest) algorithms have this problem. There must be at least one data point on each branch of the tree according to their algorithm. Despite the similar results with EIF, their results on two synthetic data sets show a significant decrease in execution time and variable storage.

2.1.2.2 K-Nearest Neighbor

According to the study by Ramaswamy et al. [35], an approach is proposed to find anomalies based on where they are placed in relation to their neighbors. Using the method, a point is chosen and k points are calculated around it, then a distance calculation is made based on the points that surround it. A list of n possible anomalies is generated based on the results. Due to the algorithm's focus on subsets, the operations are suitable for determining whether a point isn't an outlier because the algorithm divides and focuses on subsets.

Akoglu et al. in [36] presented two versions of nearest neighbor-based algorithms and compared them with ensemble-based algorithms such as isolation forest, local outlier factor, etc. Their results are based on several geometric and analytic properties of the underlying distribution in order to determine the accuracy of distance-to-measure methods to detect anomalies. Using nearest neighbor-based methods, they were able to identify anomalous instances with good results.

A simple kNN method is compared with semi-supervised and unsupervised methods for detecting anomalies in deep images in [37]. Even though the simple method is simple, it outperforms the state-of-the-art methods in terms of accuracy, training time, robustness to impurities in the input data, and robustness to dataset type.

According to [38], Tsigkritis et al. have proposed a method that uses KNN algorithms to detect threatening behavior within a computer network or cloud. The log database was derived from a communication system developed by PCCW Global. The system could be checked using this method to determine if it was consistent with the data.

According to [39], Amer et al. introduce an extension to RapidMiner that provides several anomaly detection algorithms. Using these algorithms, they were able to obtain an AUC score on Pen Global data. The results indicate that K NN algorithms are more effective when applied to global cases, and provide good results when applied

to local outlier factors. The results indicate that computation times differ whenever the data set is larger, since the computation time of algorithms based on clusters is $O(n^2)$, while that of algorithms based on files is $O(n \log n)$.

2.1.2.3 Local Outlier Factor

The Local outlier factor (LOF) is a method for identifying outliers in multidimensional data sets proposed by Breunig et al. proposed in [40]. In this method, the anomaly score of each point is calculated based on the points of a restricted neighborhood. The method looks for anomalies in smaller groups while taking into account rather than analyzing the entire data set globally. Depending on the group in which a point belongs, it may seem unimportant from a global perspective. In this method, the problem is attempted to be solved.

A paper by Paulauskas et al. proposes using LOF for network flow anomaly detection in [41]. The data set contains 74 features grouped into 15 categories. Due to the increasing amount of time when using k-nearest neighbor methods, they compared the F-score, precision, and recall values of the best feature selection based on time and anomaly scores. As far as operation amount, data collection, and feature selection are concerned, the work is similar to ours. We differ from them in that we focus on embedding layers in our feature selection process.

The study by Yuan et al., published in [42], proposes an unsupervised anomaly detection method based on the local outlier factor for in-vehicle network traffic in the intrusion detection system. The data they collect comes from public sources. They select n features for each of the m sliding windows generated by their method by sliding the data into m sliding windows. Following the selection of these features, these data are fed into the local outlier factor algorithm in order to obtain optimal anomaly contamination and k nearest neighbors parameters.

Tao et al. present a method for detecting specific workload patterns that could be considered abnormal in their paper [43]. A TCP-W benchmark was used to experiment with their approach. In addition to detecting anomalies using LOF, their method is able to improve the performance of enterprise applications by detecting different types of anomalies. Through the use of LOF, they were able to create an accurate model.

2.1.2.4 Histogram-based Outlier Score

Goldstein et al. propose a method that works linearly while decreasing the influence of multivariate features on anomaly detection in [44]. In this method, a histogram is created for a single dimension, and each histogram is normalized so that the effect of each histogram is similar. Goldstein et al.'s research show that AUC is better when used on global data sets with small features and small amounts of data than when used with the k-nearest neighbor algorithm. As a result, the results on local data sets were worse since histograms could not accurately model the local densities.

The purpose of [45] is to examine the use of a histogram-based outlier score (HBOS) to detect anomalies within a computer network. In contrast to supervised methods, HBOS does not require training or learning phases; therefore, it does not require data labeling. It is only necessary to specify the histogram bins. As indicated by F-measure, static bin selections were more successful in 3 out of 4 cases, while dynamic bin selections were more successful in finding rare events.

Kind et al. [46] describes a new approach to highlighting anomalies in traffic by using histograms of different traffic features, modeling histogram patterns, and identifying deviations from the created models. By modeling the detailed characteristics of histograms directly rather than using entropy to identify coarse-grained differences between distributions, they differ from previous feature-based techniques for the detection of anomalies. They reported a medium number of false positives at the end of their study, but they were unable to identify false positives.

Child Location trackers are used in this study[47] to detect anomalies as outlying events. Aliyu et al. compare a number of different anomaly detection techniques, such as the local outlier factor and the knn with a histogram-based outlier score. It is only possible to detect global outliers using the kNN Global Outlier Score when k exceeds 15. While HBOS performs better than the KNN Global Outlier Score, it is not able to learn new locations that become part of the normal pattern as they become part of the normal distribution.

2.1.2.5 Cluster Based Local Outlier Factor

A method similar to LOF has been proposed by Zengyou et al. in [48]. In this method, the outliers are considered local, but their importance of them to the problem is determined by a different value, i.e., Cluster-based local outlier factor (CBLOF), which is the distance between points and the clusters multiplied by cluster size. Because the number of outliers is very small, the difference from its normal conjugate is vital to identifying meaningful outliers. Based on the comparison of the results with the existing methods, the results are promising.

In [49], Zengan et al. propose a method for detecting suspicious transactional behavior associated with money laundering based on CBLOF. By using this method, the speed of cluster-based algorithms can be combined with the detection accuracy of distance-based algorithms. In accordance with their method, they were able to rank the highest five scores in decreasing order, which is anomalous.

A method for Ping End-to-End Reporting was presented in [50] by Ali et al. A framework for measuring internet performance across 170 countries is provided in this report. To fill in the missing data, a KNN-based algorithm is used, followed by a K-means clustering algorithm, and finally, a LOF algorithm using CBLOF is used to identify anomalies. A computational improvement over the general method is provided by the k-means clustering method. Based on the results, they were able to identify the anomalous cluster with the highest score.

2.2 Anomaly Detection using Neural Networks

Karimi et al. propose using neural networks to detect anomalies in Border Gateway Protocols in [51]. Classifiers are used to classify anomalies in their methods. A supervised learning technique is used to identify their data because it can be classified before training. Because there are only two classes, the Hyperbolic Tangent Sigmoid is used as the activation function. Accurate results are obtained.

2.2.1 Anomaly Detection using Recurrent Neural Networks

A Recurrent Neural Network (RNN) is a type of Artificial Neural Network (ANN) that analyzes sequential data and focuses on temporal behaviour[52], by utilizing a feedback connection within its architecture. Although this ability of RNNs to promising, RNNs in their basic form, namely the vanilla RNN, are very difficult to train. This fact is clearly discussed in [53], where the authors show there is a problem with RNN's ability to utilize memory. In cases where a sequence is longer than a certain size, a vanilla RNN will have trouble carrying the previous information. The other way around, if an RNN uses a short period of time for a sequence, the effect of this network will be relatively small, and there is no point in using it. In order to solve this shortcoming of the vanilla RNN, a new model is proposed by [53], namely Long Short-Term Memory (LSTM). This novel approach aims to overcome exploding and vanishing gradient problems outlined in [54] by utilizing an architecture that allows efficient gradient flow during back-propagation. Due to the fact that LSTM can look up to long sequences without compromising its short memory advantages, it's still an appropriate choice for this kind of application and is widely used in the literature.

Auto-encoders are neural networks that encode unsupervised data to create representations by using encoding techniques. The AE is composed of the encoder, the bottleneck, the decoder, and the reconstruction loss parts. By compressing unsupervised data, its dimension is reduced and it is converted into encoded representation. In the next step, the decoder tries to recreate the encoded representation. The difference between the reconstructed version and the original version at the end will demonstrate the performance of the model. LSTM AE processes sequential and unsupervised data using LSTM network architectures as encoders and decoders.

With LSTM AE, the reconstruction loss could be used to detect anomalies in ECG signals, according to [55]. Here, the reconstruction loss could be expressed as anomaly scores directly or as a series of steps until a reliable metric is determined. Due to its sequential data analysis approach and the fact that most of the data used are unlabeled, LSTM AE can detect anomalies in several different problems.

It is similar to our work that Malhotra et al. propose in [56], where they propose using Long Short-Term Memory Auto-Encoder for anomaly detection for multisensor data. But the difference between our work is in how we analyze anomalies from different perspectives. To detect normal behavior of normal time-series data and reconstruct it, this work employs LSTM AE, which includes a single hidden layer. A reconstruction error is created in this way, and this is used to determine if a point is anomalous. This method is being used to detect anomalies in data sets such as ECGs, engines, power

demands, and space shuttle data. This experiment shows a positive likelihood ratio that is higher than 1.0. LSTM AE provides higher scores for anomalies across all data sets.

RNNs, according to Radford et al. in [57], understand sequences of communication between computers through normal behavior. Because most of the industry uses rule-based approaches to manage computer networks, their work focuses on this challenging problem. Researchers are seeking a more suitable approach since these approaches do not identify unknown patterns. By tokenizing the network flow, they attempt to teach the LSTM to learn the complex relationships between the words and sentences in this language. The use of LSTMs and anomaly sequences in their work is similar to ours, but the context and embedding approach are different.

According to Latif et al. [58], RNNs can be used to automate cardiac auscultation and detect abnormal heartbeats. Their work is based on the Physionet Challenge 2016 data set, which is publicly available. PCG signal analysis relies on the selection of features and the combination of those features. Due to the fact that abnormal heart sounds differ in their temporal context from normal ones. The use of these algorithms relies on bidirectional LSTMs and Gate Recurrent Units(GRU). As a result of the nature of sequence modeling, they decide on which frames should be analyzed in the past and in the future. Accordingly, BLSTM delivers better results in terms of accuracy and sensitivity, while GRU provides more specificity. An analysis of different RNN architectures in anomaly detection concepts is presented in their work.

A method for anomaly detection based on LSTM AutoEncoders has been proposed by Wei et al. in [59]. Their research focuses on detecting anomalies in indoor air quality, which is an important issue since it directly affects human health. The model was trained by using multiple LSTM cells so that it could learn the long-term dependencies of the data in the sequence of time series. In their analysis, they were able to generate an anomaly score similar to that we use when analyzing data using reconstruction loss. In order to demonstrate the effectiveness of their method, they tested it on a real-world data set obtained from New Zealand schools. Consequently, they were able to obtain a reliable accuracy rate when compared to previous studies.

Salahuddin et al. discuss how distributed denial of service (DDoS) attacks draw attention to the systems that are being attacked in [60]. According to their research, these attacks are affecting healthcare systems, education sectors, and financial institutions. Using autoencoders based on time series sequences, they provide a novel approach. This resulted in very high accuracy and F1 score.

A model based on LSTM that is used to detect anomalies is presented by Zhao et al. in [61]. The authors propose a novel method of extracting features and then train an LSTM model that outperforms popular anomaly detection algorithms including support vector machines (SVMs), principal component analysis, etc.

2.3 Anomaly Detection in Financial Applications

In financial applications, goods, money, assets, and funds are exchanged. Customers rely on companies that exchange them to ensure the flow continues. The provision of

this reliance is not easy, and companies use different methods to identify performance issues, frauds, etc. As a result, their system could be improved and customers' trust could be maintained.

Interactions between financial participants are ubiquitous, and they are often tracked in order to gain a competitive advantage. As an example, in the case of mobile phone fraud, there may not be any financial transactions involved, but rather an interaction between two customers. It is usually possible to analyze values on the edges in most of these cases [14].

An application of multidimensional anomaly detection is essentially involved once a multidimensional representation of the claims has been created. In order to develop an anomaly detection system, the key step is to extract the relevant features from the insurance claim documents. It is highly domain-specific to extract features in insurance claim scenarios since it involves identifying indicators that are highly specific to the type of claim at hand. We use a similar strategy for feature extraction, and this explains the importance of understanding the domain in order to detect anomalies[14].

2.3.1 Anomaly Detection in Banking Applications

An anomaly detection problem context is explained by means of bank applications in [3]. A brief discussion of the effects of these applications on the public is presented by Chilukuri K. Mohan et al. Considering how important these applications are, it explains how anomaly detection could be beneficial to them. Several techniques are discussed in the area, including clustering, nearest-neighbor, and sequential methods. We differ from their study in terms of data perspective and the method that we are using, as they noted in their conclusions and future work. The proposed methods are based on hypothetical data, which are analyzed mathematically.

2.3.2 Anomaly Detection in Money Transactions

B.Dumitrescu et al. propose in [62] a paper about Anti-Money Laundering which points out both its importance and its data set issues. Specifically, their problem focuses on the detection of network anomalies where bank clients might be involved in money laundering and their relationship may affect the system. Adapting the problem as a directed graph in which nodes represent accounts and edges represent transactions, we can visualize the problem as a directed graph.

A major objective of this document [63] is to develop an algorithm for predicting fraudulent monetary transactions capable of overcoming the problems associated with the detection of anomalies and having a high level of accuracy. They compared three different methods, namely the k-means algorithm, the SVM algorithm, and the decision tree algorithm. Consequently, they selected an isolation forest that provided the best test evaluation and response time.

As described in [64], Ounacer et al. emphasize that the use of credit cards has increased significantly in recent years, and fraudulent activities are becoming an in-

creasingly important issue in this field. The authors stated that current approaches have difficulties dealing with high-volume data, which results in the discovery of anomalous cases. In terms of AUC and accuracy score, they were focused on improving the performance of isolation forests. In comparison with KNN and Decision Tree based models, the IF algorithm provided the highest AUC score.

2.4 Categorical Embedding

Several methods are described in the literature for handling the features contained in the data set, and these methods have a significant impact on the anomaly score results for the given problem.

In [9], Guo et al. propose a method for calculating the Euclidean Distance of points by mapping categorical variables into functional approximations. Structured data is the focus of their work. This data is structured as a table of columns corresponding to different features of a collection of data. In this context, this problem is similar to one of our research questions. In order to find a normal representation of an anomaly, it is important to know how to represent the features of the data when feeding them into our network. Unlike the account number of participant code in our features, this method provides a discrete variable, such as age, bus line number, etc. When sorted or ordered, these features are meaningless. The embedding layer will be enhanced with a one-hot encoding layer during the feed to the network. Feature concatenation is therefore possible. Through these methods, they are able to create networks that do not know anything about the context in which they are operating. By using high-dimensional data and the embedding layer that has been created, the embedding space can place similar points in a distribution.

According to [65], Zhou et al. propose a method to detect structural irregularities since anomalies may be defined in different ways. Similar to the way we are trying to understand what anomaly means in our context, their problems focus on the same concept. However, our method relies more on context-specific and data-driven methods. Feature selection and querying rely on our expertise in the field while we are trying to understand the data. Anomaly detection problems, such as time-series and network anomaly detection, are examined in this research. Graphical data shows a significant improvement in network performance when the embedding layer and function are optimized through the context.

CHAPTER 3

MONEY TRANSACTION DATA SET

The following section presents the constructed data sets, a masking operation together with a number of processes involved, and the features extracted.

3.1 The Constructed Data Set

In this thesis, we construct a data set by collecting various transactions between different banks. In the case of a transaction between different banks, the money transaction request is sent to the middleware company that controls these types of transactions for all of these banks in the specified country. In *Figure* 3.1, representation of the transaction flow is depicted.

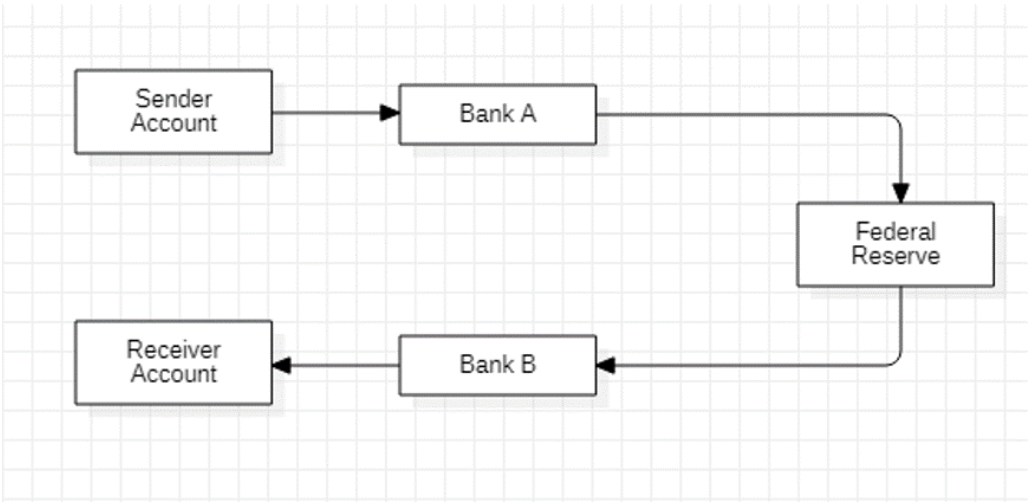


Figure 3.1: Sample representation of a transaction flow

3.1.1 Masking Operation based on The Law on The Protection of Personal Data

The constructed data set is a collection of real-time transactions, and is protected by the law on the protection of personal information [1] There is no personal informa-

[1] https://www.kvkk.gov.tr/

tion contained in this data. Given that our data contains several different aspects of the sender and receiver's identities along with the bank identities, we worked on a method of masking all these characteristics while retaining the unique characteristics of each transaction. As a result of the uniqueness of these characteristics, categorical data points are more likely to be relevant in a particular context. The database hash function [66] is used to create a unique hash value for each identity that contains sensitive information. Text is passed into the hash function, which calculates the hash of the string and returns it in hexadecimal form. An example query and its result can be found in the following *Table* 3.1.

Table 3.1: Hash algorithm query result

Query	Result
select hash('Anomaly Detection');	aac9771bc17dccacy62hc7fb9j55a846

3.1.2 Features and Explanations

This data set contains a month's worth of transaction information among individuals. Over 100 million entities were originally included in the data. Each of these entities represents a transaction between two banks. Time-related information such as entry time and completion time is included in these entities. Furthermore, the corpus includes information about the money that's transferred along with identification information so that we can determine which transactions belong to whom.

Transactions contain multiple features, as described in the features section. Conventionally, a transaction's amount of money is usually the main factor in determining whether it is normal or not. Considering that the funds are being transferred between different banks, each bank has a different set of rules. Despite the fact that the amount in a transaction is limited by the middleware, there is still the possibility of anomalous behavior occurring when it comes to the total number of transactions. There may be a lower amount of money per transaction than is expected, such as 0.001 units per transaction, or the total amount of money per day may be approximately 1 million units, which is at least 500 transactions per day. As a result of these patterns, a bank may reject future transactions for that individual as a result of their anomalous behavior appearing on their statement.

The original data transaction entities include the following properties:

1. Transactions contain 17 features, consisting of date-time, categorical, and numerical attributes.
2. Following the elimination of features not directly related to our anomaly detection processes, such as application identities for request directions, eight of which are unique identities for the senders and receivers, including name, nationality, account number, and participant identity
3. As described above, these identities are masked in accordance with The Law of Personal Data Protection.

4. As a final feature, we have the amount and the time of the entry of the transaction.

5. The system limits the amount of each of these transactions to 2000 in the data that is collected.

Table 3.2: All features

Features	Uniqueness
Receiver Individual Id	Unique
Sender Individual Id	Unique
Receiver Bank Id	Unique
Sender Bank Id	Unique
Amount of Money	Non-unique
Entry Time	Non-unique
Completion Time	Non-unique

In light of the fact that each entity represents a unique transaction between two different individuals, the analysis began by grouping transactions that were made by a certain sender account on a specified date by their amount. The sum of these amounts is called "AmountOfMoney", while their count is called "AmountOfTransaction" as described in *Table* 3.3. Different sizes of different subsets are generated in order to enable our network to learn the pattern of a normal transaction. In order to train and determine initial results, the generated subset is divided into train, validation, and test data sets. In the following *Table* 3.4, the numbers can be viewed.

Table 3.3: Daily transaction/amount scenario

Features	Uniqueness
Total Amount of Transaction	Non-unique
Total Amount of Money	Non-unique
Receiver Bank Id	Unique
Sender Bank Id	Unique
Entry Time	Non-unique
Completion Time	Non-unique

Table 3.4: Transaction data set

Data Set Id	Train	Validation	Test	Total
1	25500	3015	1485	30000
2	127500	15075	7425	150000
3	425000	50250	24750	500000
4	1275000	150750	74250	1500000
5	2550000	301500	148500	3000000

3.1.3 Sequence Based Sub Data Sets

The data set we introduced consists of several different connections between the actors in the process. According to these actions, we are able to create different scenarios for our feature combination. Different amounts of data determine the intensity of transactions per individual, which aids in identifying different scenarios. We are able to train the data set using different combinations of features according to scenarios, but the amount of data remains fixed in order to examine their effect on the scenarios equally according to their intensities. The reason for this is that every time an individual increases their transaction amount, their intensity increases in a different way, resulting in different behaviors. As we attempt to understand this, we will focus on three cases, namely the maximum, average, and minimum training amounts[2].

3.1.3.1 Based on Same Receiver or Sender

The transactions, by nature, connects a receiver and sender account. This connection creates an individual transaction history as a specific case. For a given month, certain senders and receivers may appear more than others. It is more likely that their anomaly cases will differ in certain situations. To accomplish this, the data for these individuals are subtracted and used to create a subset which can be used in the next step. The features are described in *Table* 3.5.

In order to create different subsets, different individuals with different transaction histories are used. Various individuals with different identifications were involved in the money transaction from their perspective. In the *Table* 3.6, the top 10 maximum number of transactions is listed ascendingly from sender perspective.

In the *Table* 3.7, the top 10 maximum number of transactions is listed ascendingly from receiver perspective.

[2] Our system may combine several individuals if there are not enough minimum transactions.

Table 3.5: Individual based scenario features

Features	Uniqueness
Receiver Individual Id	Unique
Sender Individual Id	Unique
Amount of Money	Non-unique
Entry Time	Non-unique
Completion Time	Non-unique

Table 3.6: Total number of transaction per month highlights, from the sender perspective

Sender Account Number(Hashed)	Total Number of Transaction per Month
6259197ea...	751516
ca0e2f1e4...	259244
4055224c4...	206336
6695da274...	146597
693b4e4db...	17777
0acf6b302...	14111
7f22a09fc...	11566
84d0f40fd...	9154
473bbe174...	1793
fe0c885df...	1774

3.1.3.2 Based on Same Participant or Bank

Since every accounts are belong to a certain bank, these transaction could be identified in more general perspective where the transaction history could be analysed in terms of these banks perspective as described in *Table* 3.8. We examined several different amounts of transactions per month in order to cover different aspects of the problem.

A subset of the original data was created by choosing three different banks or participants. Each of these three subsets has a maximum, average, and minimum amount of transactions[3]. In *Figure* 3.9, we can see how many transactions are retrieved from the query results in terms of receiver perspective. According to that figure, Bank 1

[3] The training lower limit was created by selecting a bank that has a minimum amount of transactions per month. A minimum of 30000 transactions are required per month for training purposes.

Table 3.7: Total number of transaction per month highlights from receiver perspective

Receiver Account Number(Hashed)	Total Number of Transaction per Month
504fb224...	519194
bb167600...	177698
0639b4a0...	164178
07cfea17...	155086
d1dc4583...	122684
a2b7a2ed...	117290
bf3891df...	104145
d5bff409...	70633
6695da27...	65164
b9a8f13d...	62093

Table 3.8: Bank based scenario features

Features	Uniqueness
Receiver Bank Id	Unique
Sender Bank Id	Unique
Amount of Money	Non-unique
Entry Time	Non-unique
Completion Time	Non-unique

had the highest transaction volume, Bank 9 had the average transaction volume, and Bank 19 had the lowest transaction volume for approximately both cases.

As in this case, we are choosing mutual maximum, minimum, and average cases to maximize training efficiency. Considering the *Figure* 3.10, it is clear that Bank 19 has more than 30000 transactions, which makes it optimal for the minimum scenario.

3.1.4 Testing Data Set

In order to conduct testing, a test data set is required. Since there are no previous anomalies in the system, we generate our own training data set using expert information. It is true that finding an anomaly using a machine learning algorithm could be considered trivial since we design and create the given anomalies, however, we never use them to feed into our learning system. In order to train our LSTMS, we only use

normal behavior data. There are different anomaly cases depending on the perspective[4], as we mentioned earlier. Transactions can be viewed as amounts of money and amounts of transactions when viewed as a scale, and the amount of money limit will create an anomaly when viewed as a scale.

In preparing the anomaly following the test data set, the following features were taken into account in terms of the amount and number of transactions per person per day: MA@N (Maximum Amount For N Transaction), MAXT (Maximum Transaction per Month), ATAM (Abnormal Total Amount of Money per Person), ATAT (Abnormal Total Amount of Transaction per Person).

MA@N. With a limited amount of money per transaction, there is no anomaly when sending one transaction per person. Because each transaction can only contain 2000 units. However, there is no standard limit on the number of transactions. A number of participants in the system have no limit on the number of transactions they can send per day. On the basis of the analysis that was run on the whole data set, 751516 transactions were possible per month. Participants are assigned a maximum value based on the total transactions they make per day. During the creation of an abnormal transaction, the metric used is displayed.

$$ATAM = rand(0.8, 1.5) * MA@N \tag{3.1}$$

ATAM. This metric, which can be seen in *Equation* 3.1, illustrates how abnormal amounts of money are calculated per person.

MAXT. The amount of transactions per day for an individual is calculated using several values related to total money. An anomaly factor per person is determined by their monthly maximum, which is inserted into the training data. The metric usage is displayed during the creation of an abnormal transaction.

$$ATAT = rand(0.8, 1.5) * MAXT \tag{3.2}$$

ATAT. According to *Equation* 3.2, this metric illustrates the calculation of abnormal transactions of money per individual. By utilizing the equations above, the anomalies for training and test data are computed. As much as the outlier detection ratio, these data were combined with the normal data for the purpose of training our machine learning models.

Several different features are being considered when preparing a data set for individual scenarios. We will have to create different anomaly data sets for these individuals, as suggested in the explanations in the *Section* 3.1.3.1 and *Section* 3.1.3.2 above along with the *Table* 3.7, *Table* 3.6, *Table* 3.9 and *Table* 3.10.

It is important to consider the following additional metrics when creating the data set for the scenarios explained: AVGM(Average Amount for Person per Month), MATD(Maximum Amount of Transaction per Day), PTT(Possible Transaction Time), PET(Potential Entry Time), PCT(Potential Completion Time), AAM(Atypical Amount

[4] We have created all of these cases and have had the domain experts validate them.

of Money per Person per Day), AAT(Atypical Amount of Transaction per Person per Day).

$$AAM = min(2000, rand(0.8, 1.5) * AVGM) \tag{3.3}$$

The internal systems specify t seconds as the maximum processing time for a successful transaction. Transactions that take more than t times will time out and be marked as unsuccessful. By subtracting entry time from completion time in *Equation* 3.4, we are trying to estimate the process time. Last but not least, we are providing a time that is longer than t in order to discover any possible system failures as well as the abnormal number of transactions per month for a particular individual. Our training still uses PETs and PCTs as features that create sequences of time sets, even though we are processing PTT as described below.

$$PTT = PCT - PET \tag{3.4}$$

We are able to detect issues with an enormous amount of timeout in the system by monitoring applications, according to previous problems. As a result, we are not concentrating on creating a processing time that is higher than t, as indicated by the range of random values[5] given by the following *Equation* 3.5. Adding another possible scenario for this situation is all that we are doing.

$$PTT = random(0.5, 1.1) * PTT \tag{3.5}$$

The monthly transaction history of these individuals could also explain their abnormal behavior. With the amount per transaction is limited, we create a large number of transactions per individual using an average amount using their maximum amount of transactions according to the *Equation* 3.6.

$$AAT = rand(0.8, 1.5) * MATD \tag{3.6}$$

Finally, we create a general data set that allows us to generate different anomaly scores using the models created by the above scenarios. Data sets for general testing include the original data unfiltered as well as several additions. In addition to these data, we will focus on the previous examples that have already been discussed. As opposed to focusing on an individual previously considered, we are using random selections of individuals this time. As with the previous group of individuals, we are selecting a random set of them in order to alter their behavior.

[5] Applications that appear in the system are analyzed to select a range for random values that will be used in these equations.

Table 3.9: Number of transaction per month highlights from receiver bank perspective

Receiver Bank	Total Number of Transaction per Month
Bank 1	22050538
Bank 2	14752544
Bank 3	11939023
Bank 4	9890790
Bank 5	9015631
Bank 6	8825460
Bank 7	8198513
Bank 8	5136149
Bank 9	4106790
Bank 10	3565670
Bank 11	2692033
Bank 12	1101383
Bank 13	730983
Bank 14	722730
Bank 15	559890
Bank 16	233771
Bank 17	231089
Bank 18	120787
Bank 19	45246
Bank 20	43944
Bank 21	18926
Bank 22	15741
Bank 23	14036

Table 3.10: Number of transaction per month highlights from sender bank perspective

Sender Bank	Total Number of Transaction per Month
Bank 1	18711334
Bank 2	18680220
Bank 4	13465512
Bank 5	12306423
Bank 6	11069938
Bank 7	6155109
Bank 10	5806264
Bank 9	5599989
Bank 3	4951996
Bank 8	2739938
Bank 11	1779402
Bank 12	1231743
Bank 14	470444
Bank 13	246895
Bank 16	223776
Bank 15	176004
Bank 17	160916
Bank 18	84799
Bank 19	75946
Bank 22	32879
Bank 20	23383
Bank 21	10980
Bank 23	7785

Table 3.11: Transaction test data set

Id	Problem Case	Individual Id	# of Feature	Size
1	General Testing Set	Random	6	50000
2	Person with Max Transaction in Receiver	504fb224...	6	30000
3	Person with Avg Transaction in Receiver	07cfea17...	6	30000
4	Person with Min Transaction in Receiver	b9a8f13d...	6	30000
5	Person with Max Transaction in Sender	6259197ea...	6	30000
6	Person with Avg Transaction in Sender	6695da274...	6	30000
7	Person with Min Transaction in Sender	693b4e4db...	6	30000
8	Bank with Max Transaction	Bank 1	4	30000
9	Bank with Avg Transaction	Bank 9	4	30000
10	Bank with Min Transaction	Bank 19	4	30000
11	Amount of Money and Transaction	Random	4	30000
12	Amount of Money and Transaction	Random	4	150000
13	Amount of Money and Transaction	Random	4	500000
14	Amount of Money and Transaction	Random	4	1500000
15	Amount of Money and Transaction	Random	4	3000000

CHAPTER 4

ANOMALY DETECTION USING LSTM AE NETWORK

This chapter provides the details of the proposed LSTM-based deep learning architecture.

4.1 General Flow of the Proposed Method

LSTM AEs have proven to be an effective method for detecting anomalies in the literature. However, to the best of our knowledge, no research has been conducted using LSTM AE in the context of money transactions. Additionally, we are focusing on the fact that this problem contains multiple anomaly cases. Ultimately, we could create sequences that are different from one another and train them individually. In order to create an architecture, all these cases are combined after they are trained. A number of models and approaches will be included in this architecture in order to detect anomalies. A depiction of the proposed architecture for this problem is shown in the *Figure* 4.1.

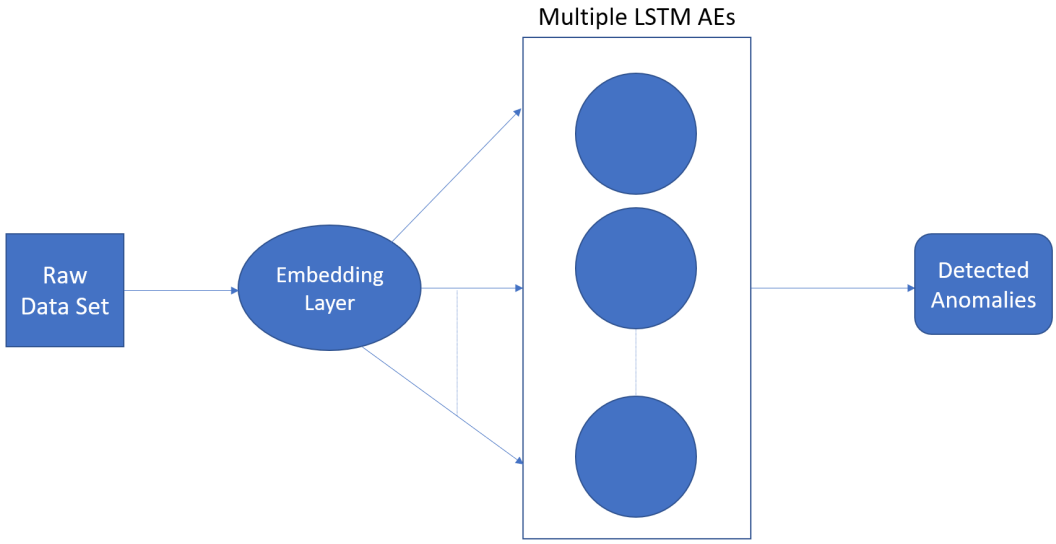


Figure 4.1: General flow of the proposed architecture 1

Aside from the first architecture, we also attempted to combine the general findings

of the machine learning algorithms used which is shown in *Figure* 4.2. In accordance with the results of the machine learning architecture, we are using the most accurate results. Due to the fact that this result shows N tuples that are most likely to be anomalies. Following the discovery of possible anomalies, the results are used to detect more accurate results. A sector research study indicates that detecting all anomalies is of paramount importance. The failure of a model to identify anomalies in a system can have more serious consequences, such as system failure or in our case problematic transactions.

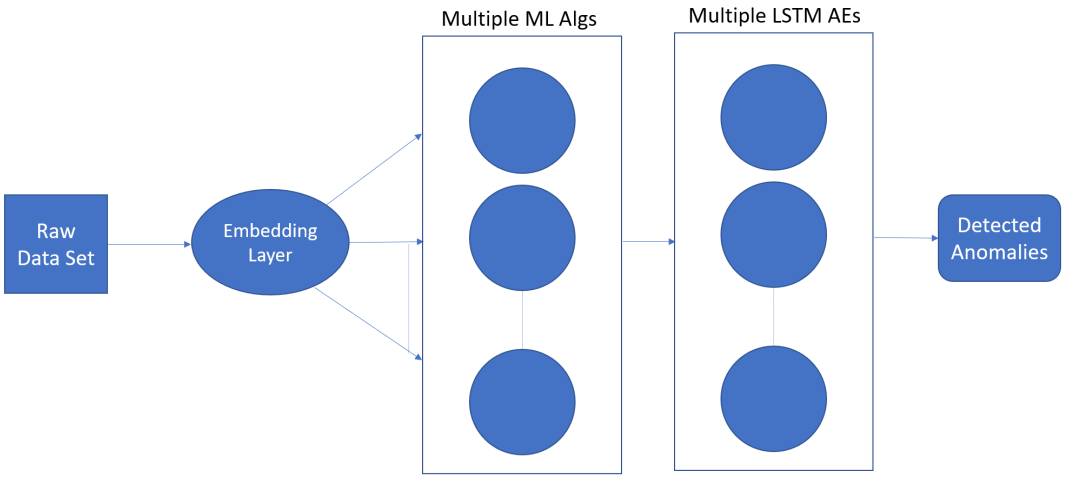


Figure 4.2: General flow of the proposed architecture 2

4.2 Layer Architecture

We have used two LSTMs per encoder and decoder in our LSTM AE. Furthermore, an embedding layer represents the given features by creating a representation of them.

Auto-encoder networks consist of two main networks, namely encoders and decoders, which are used to handle the process. Rather than creating a low-level representation of the input x, encoders create a compressible version of the input that can be reconstructed. This input is then processed by the Decoder Network, which attempts to reconstruct it using the information at hand. In the final step, an error reconstruction is calculated based on the comparison between the given output and the given input. A possible anomaly in the data could be identified by using this reconstruction error as a score for anomaly detection. It is possible to see a representation of the sample network in *Figure* 4.3.

An encoder-decoder LSTM is used to read a data set of sequences, encode them, decode them, and recreate them based on the input sequence. During the evaluation of the model, it is examined whether or not it is capable of recreating the input sequence. LSTM AE is a special kind of AE that utilizes LSTM networks in both the encoder

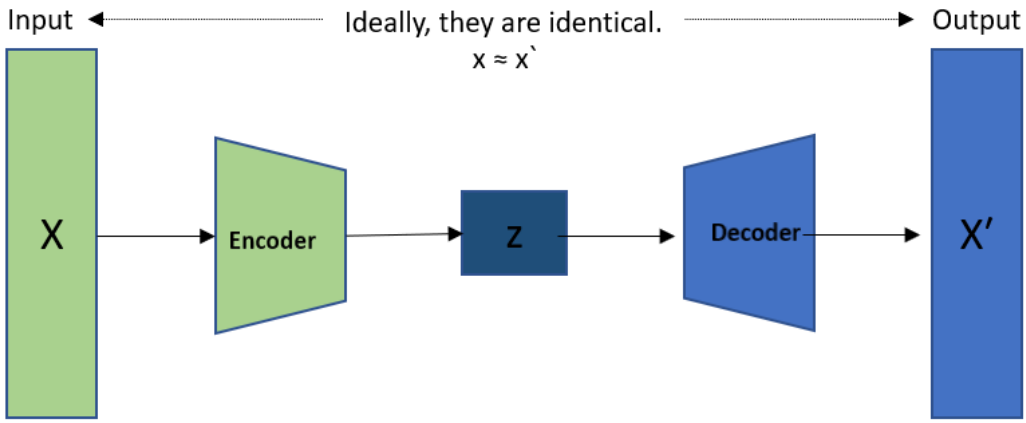


Figure 4.3: Auto-encoder networks representation

and decoder sections to create a reconstruction error, which can be used to examine the effects of past and future data. A representation of this can be seen in the following *Figure* 4.4.

4.2.1 Embedding Layer Based on Sequence Models

Every different subset that is presented is creating different scenarios for anomaly detection. In our work, we are trying to analyze this concept in terms of these perspectives and trying to create an anomaly score based on all these cases.

Our embedding layer architecture is structured in such a way that the features chosen will result in different sets of features in the final implementation. Following their selection, we attempt to create a reasonable representation of these features that the network could understand.

During the process of combining these features, previous domain problems are carefully considered. By converting selected features into numeric values, this embedding layer creates a suitable representation of selected features such as date, time, and amount values. The following *Figure* 4.5 illustrates the procedure for selecting features. Following the selection of these features, an LSTM AE is trained.

As each of these models is trained per case, it is then tested on a general test data set as well as the specialized data set for each case.

4.2.1.1 Decision Tree for Selecting Scenarios

A set of features is considered when selecting a scenario for our anomaly detection data set. An order has been established for the consideration of these features. As

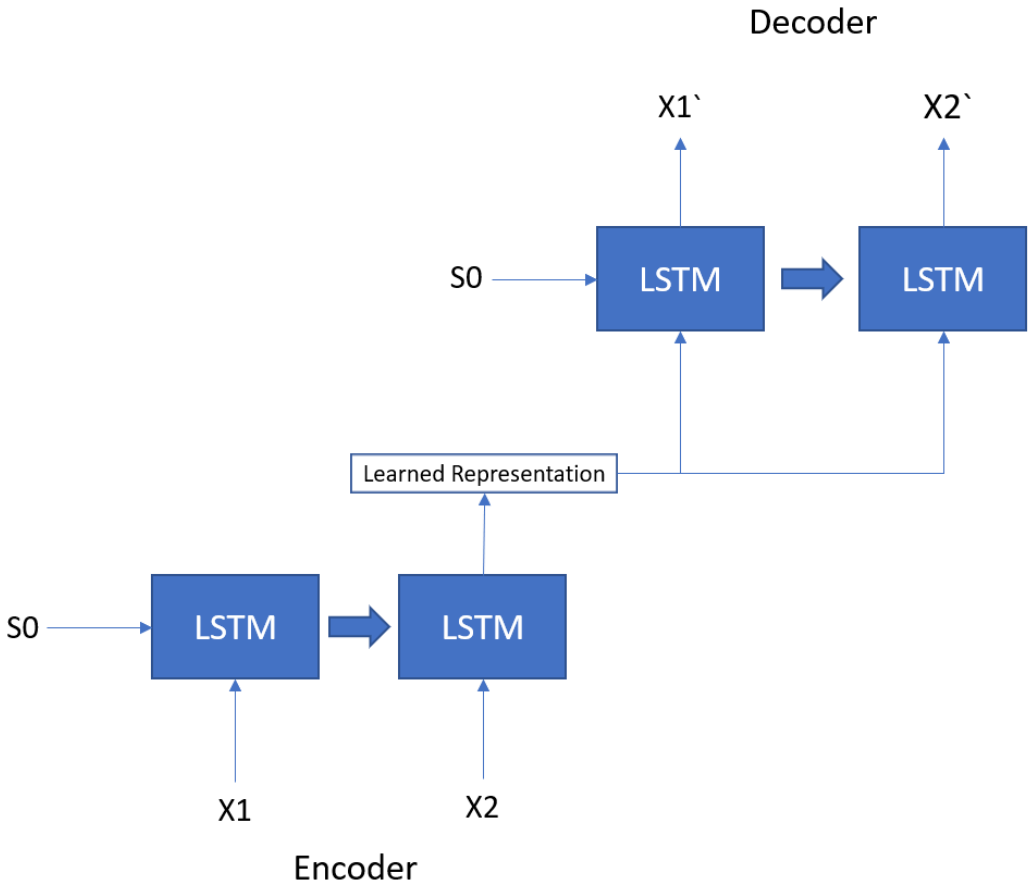


Figure 4.4: Long short-term memory auto-encoder networks representation

soon as the scope of the transaction is determined, we are able to select a feature bankId or accountId in this case. Taking the next step will lead to another decision that needs to be further considered and analyzed. Once the direction is determined, the identity of the bank or account can be determined from either the sender or receiver's perspective. Lastly, the amount of the transaction or the amount of money is considered. The analysis in this study focuses on both the number of transactions and the amount of money. An illustration of these paths can be found in *Figure* 4.5.

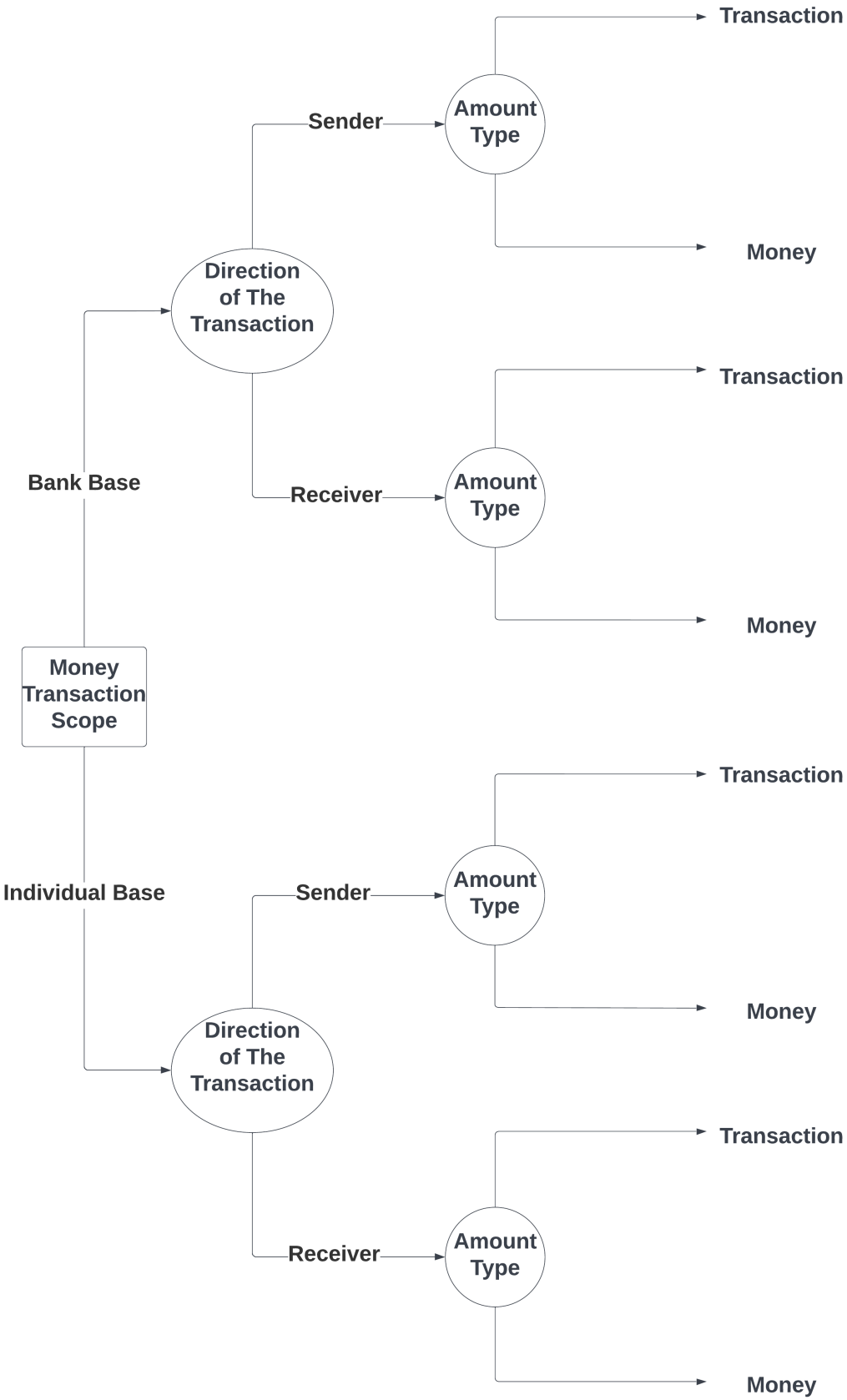


Figure 4.5: Handcrafted decision tree for selecting features

CHAPTER 5

IMPLEMENTATION AND EVALUATION

5.1 Implementation

Python programming language is used for the implementations while training models and analyzing the results. One of the reasons for choosing Python is the existence of several frameworks. These frameworks provide necessary tools that allow the programmers to take advantage of faster coding performance and built-in interpreters [67].

PyTorch[68] is a machine learning framework developed by Facebook. Powered by Torch, an open-source library, PyTorch is a widely used open-source framework. Computationally efficient libraries and flexibility are provided by PyTorch when implementing deep neural networks. Unlike some other deep learning frameworks, PyTorch uses dynamic computation graphs. Dynamic graphs are created on the fly through forward computation rather than static computation. Graphs of static computations are defined prior to their execution. In this way, every time the graph is iterated, it is recreated from the ground up.

Several of the machine learning models used in this work were imported from the Python Outlier Detection Framework [69], which is an open-source Python toolbox for detecting outliers on multivariate data that is scalable and easy to use. A series of anomaly detection algorithms and visualization tools are being used as part of the process to detect anomalies.

5.2 Experimental Evaluation

In spite of the fact that some parameters are kept similar for the purpose of fair comparison, there are several different metrics that may have an impact on some utilized algorithms, such as the nearest-neighbors and clustering-based machine learning techniques. By combining different features and sizes of data sets, we hope to achieve several different results in our network.

5.2.1 Anomaly Detection Performance Evaluation

It is considered more challenging to compare the performance of unsupervised anomaly detection algorithms with supervised classification algorithms. In order to compare an unsupervised anomaly detection algorithm, it would be necessary to compare more than just the accuracy or precision/recall values. In a classification problem, identifying a value with the wrong class would be considered an error, whereas in anomaly detection, if x anomalies were to appear at the top x+10 data points, this would be considered a good result. In this case, ranking anomaly values from first to last according to some threshold would result in X tuple values which represent True Positive Rates and False Positive Rates respectively. These values would be used to draw a receiver operator characteristic (ROC) curve [70], and the area under the curve (AUC) would be used to compare the performances [24].

$$TPR = TP/(TP + FN) \tag{5.1}$$

5.1 and 5.2 are the equations that describe how these rates are calculated.

$$FPR = FP/(FP + TN) \tag{5.2}$$

Area under curve (AUC) is a measure of the area under the curve calculated by Simpson's Rule [71]. AUC is a measure of the quality of our algorithm and the higher the score, the better.

5.2.2 Experiments

There are a variety of methods that were applied to different subsets of the original data, as well as abnormal transactions that were generated. The application of these methods allows us to compare several different machine learning algorithms on various data sets. Tables containing all the related results are presented in Appendix A. A number of factors are taken into account when selecting these algorithms. HBOS, which is a fast algorithm, is challenged by multivariate data sets. As it is further discussed in the following sections, although LOF and KNN show a higher level of accuracy, LOF and CBLOF lack global detection capabilities.

5.2.2.1 Machine Learning Methods

For a given feature space, when the locality (or proximity) of the data point is sparsely populated, it is considered to be an outlier. There are several ways in which a data point can be considered close, each of which is subtly different [72]. The following definitions of proximity are typically used in outlier analysis: cluster-based, distance-based, and density-based. The following algorithms are used in this study to detect outliers based on proximity. In selecting our algorithms, we referred to the benchmark results presented in the paper[73].

1. Local Outlier Factor (LOF): An index of local deviation of density measures the disparity between a sample's density and that of its neighbors. Anomaly scores are local in that they depend on how isolated the object is from its surroundings. Locality can be determined by calculating the distance between the k-nearest neighbors, which is used to estimate the local density. One can identify samples with a substantially lower density than their neighbors by comparing the local density of a sample with the local density of its neighbors.

2. Cluster-based Local Outlier Factor(CBLOF): CBLOF utilizes a clustering algorithm to generate a cluster model from a data set. With the help of the parameters alpha and beta, it categorizes the clusters into small and large clusters. In order to calculate the anomaly score, the point's size and distance from the nearest large cluster are taken into account.

3. Histogram-base Outlier Detection (HBOS): A variable is constructed as a histogram by HBOS. Data points can be scored according to the height of the bin in which they are located. We can inverse the height of a bin to be used as an outlier score for a data point to account for the fact that we would prefer to see a small score for normal data and a large score for an outlier. Histograms are normalized to a maximum height of 1.0. By doing so, it is possible to sum up all univariate scores equally.

4. K Nearest Neighbors(KNN): Nearest-neighbor theory is predicated on the assumption that similar observations are clustered together and that outliers are usually isolated observations located further away from the cluster.

5. Average K Nearest Neighbors: As a result, the outlier score is calculated by averaging all the neighbors.

Using the given database identifiers, all of these machine learning methods are iteratively trained. As shown in the results, blue is used to represent the range of anomaly scores between a minimum and threshold value, orange is used to represent the range of anomaly scores between a threshold and maximum value.

- Outlier Contamination
- Number of Neighbors
- Check State

Based on the following specifications, these metrics are defined and used.

- Outlier Contamination refers to the number of outliers in the data set. Using this amount, the threshold can be determined during the fit of the model. In addition, anomalies are generated for each case and the anomalies are combined with the normal data set.
- Defining k neighbors queries requires a set of neighbors. It was decided to use the default value of k, which in this case is 5.
- Upon setting True, the base estimator is checked for consistency with the Sklearn[74] standard. Once again, we used the default value, which is false.

According to the results shown in *Figure* 5.1, each anomaly detection algorithm produces outliers as black dots. It is likely that each dot is an outlier the closer it is to the origin. Similarly, the other white dots represent inliers, which represent normal behavior in the cases that have been presented.

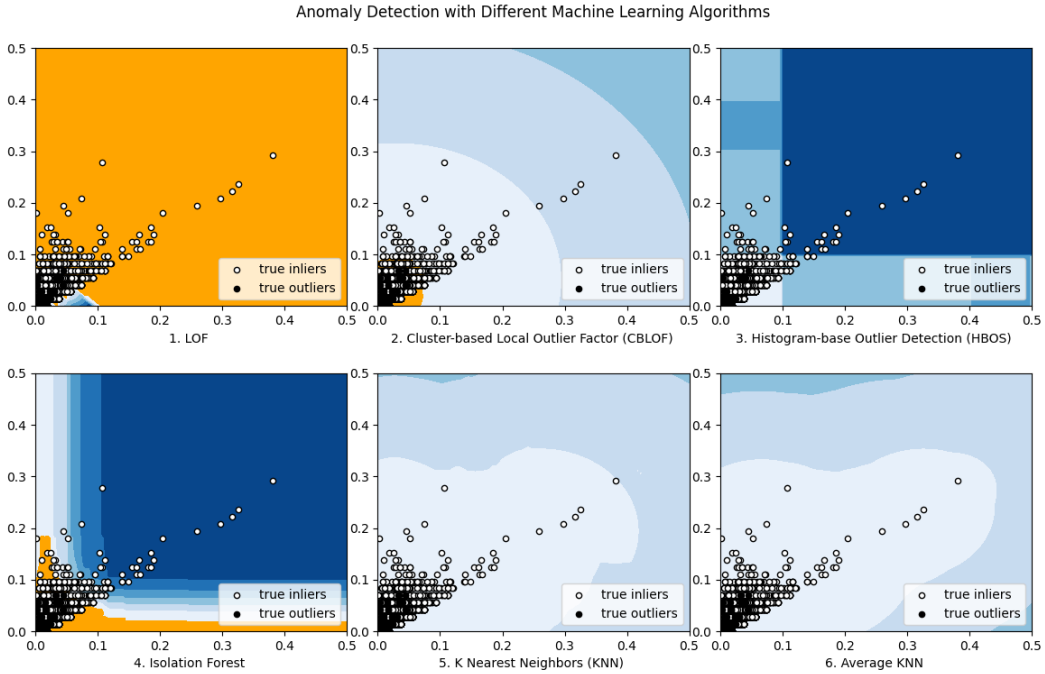


Figure 5.1: Anomaly detection results on database id 11

There is an outlier detention rate which determines how much an outlier is likely to appear in a data set, despite the orange line representing the area between the threshold value and the maximum value of the anomaly score.

HBOS algorithm performance decreases as the amount of data increases, as shown in *Figure* 5.3 and *Figure* 5.4. HBOS does not take into account these relations since the number of relationships increases and their influence on each other increases gradually. *Table* A.10 and *Table* A.11 illustrate this as well.

There is a clear difference between the algorithms that we use in each of the cases. For the second flow that we have represented, CBLOF and IF yield more reliable results.

Based on the ROC curve graph presented above, we were able to determine that the IF and CBLOF provide better performance for our field. CBLOF results in better performance in comparison with [24].

5.2.2.2 Scenario Based Training

Based on the given features, and the number of transactions distributed throughout the day, different models are created. As part of the process of detecting an anomaly,

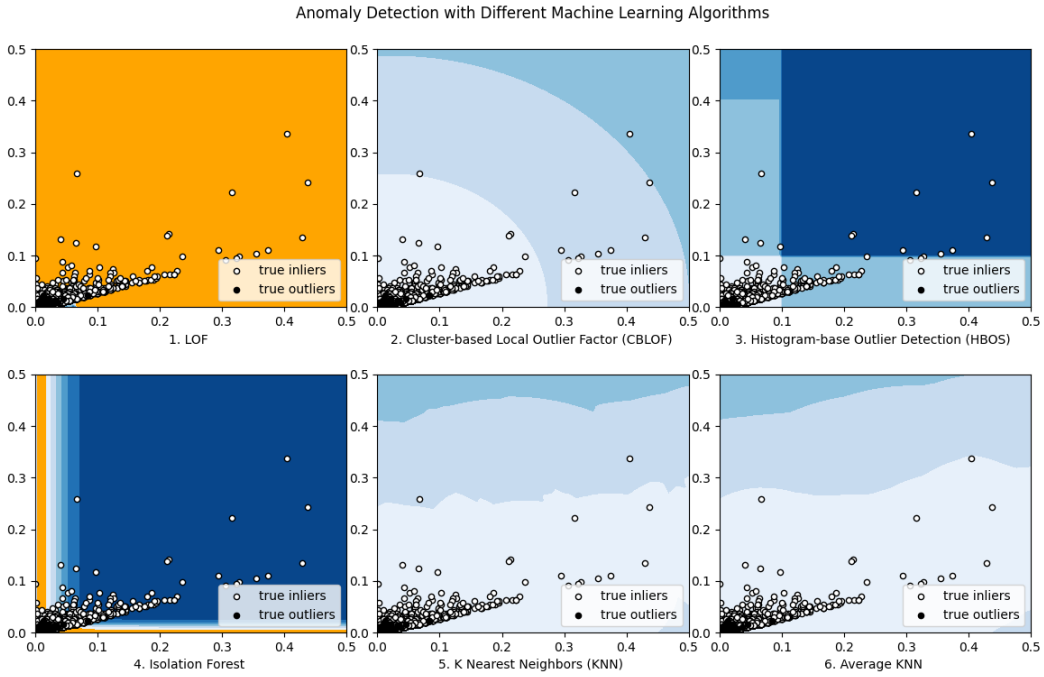


Figure 5.2: Anomaly detection results on database id 12

each of these data sets is used to develop an anomaly score. A comparison is made between each of these anomaly scores and the threshold for normal behavior for that scenario. This means that each threshold will hold a different value depending on the data set. We are focusing on the fact that each of these scenarios has a different percentage affecting general anomaly scores, whereas the detection on the general data set is based on the analysis. As a result, we are combining these anomaly scores in different percentages to see how well our models could predict them.

5.2.2.3 LSTM AE

An outlier detection factor is a learning parameter that helps the model to understand how much of the data is anomalous since in anomaly cases, the amount of anomalous data is so small that it could cause a training problem.

In order to determine the performance of the algorithm, several parameters are taken into account. The parameter values were selected according to the [73].

Compared to other optimization algorithms, Adam's results are generally superior, it has a faster computation time, and requires fewer parameters for tuning. All of these factors contribute to Adam being recommended for most applications as the default optimizer [75].

LSTM AE training history in *Figure* 5.8 indicates that our model converges around 135 epochs for Total Amount and Transaction case. In the following steps, we will evaluate the performance of our models using training and test data frames that have

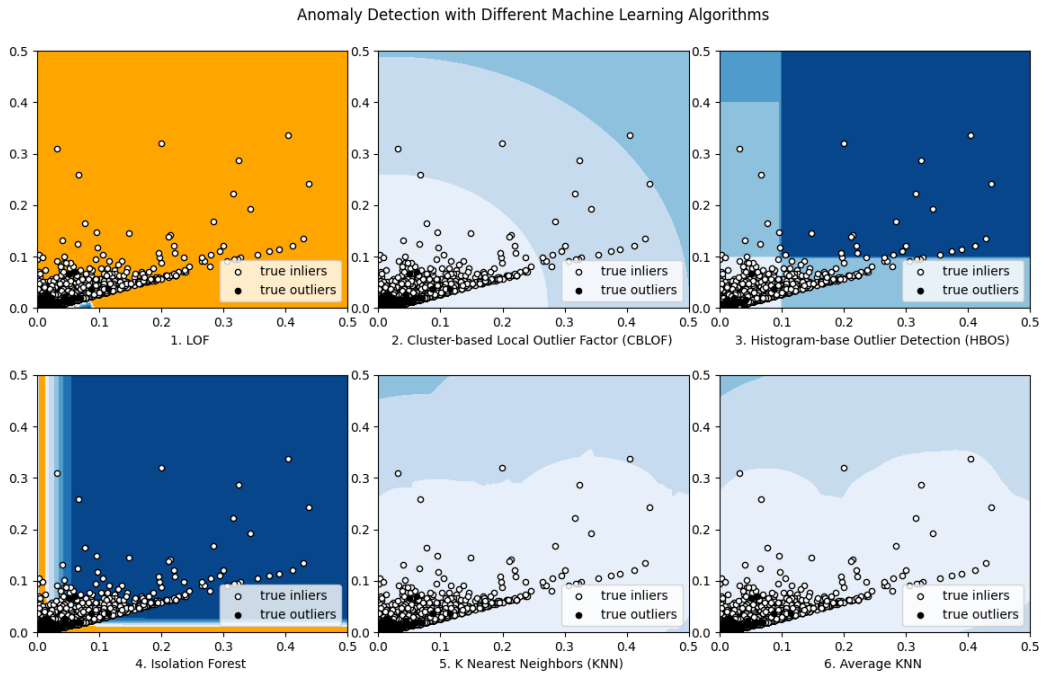


Figure 5.3: Anomaly detection results on database id 13

been separated from the general data. As a result, a threshold value can be determined for the given values. In this way, a threshold value can be determined, which helps us to identify anomalous behavior. The purpose of putting that threshold is to find a value that is not overfitting in our case. The following step is to plot anomaly cases, which are shown in *Figure* 5.9. By examining the graph, we were able to see the distribution of values and the number of values that were separated. Having placed the threshold, the tables and their accuracy are highly reliable in terms of accuracy. While several normal transactions are being conducted alongside the anomalies, the anomalies are easily detectable.

The following tables show the threshold value results. As can be seen from the table, if we use a threshold value that is too high, the model overfits. Consequently, finding anomalies becomes more difficult and anomaly detection becomes less effective. As a result, we choose a threshold value that is suitable for both cases in order to achieve better performance. Furthermore, we should select a value that finds all anomalies since marking anomalies as normal is a problematic practice.

There is a significant difference between the amount of training time required for these two cases and the first case. These scenarios do not converge within the first 200 epochs, as shown in the *Figure* 5.10.

Both receiver/sender and participant cases are analyzed using the same methodologies. Based on the results shown in the following *Figure* **??**, the distribution is more scattered due to the fact that their behavior includes a greater number of cases.

Based on the analysis of these figures 5.12a, 5.12b, 5.13a and 5.13b, we can determine that there are three specific areas that are separate. In most cases, these areas explain

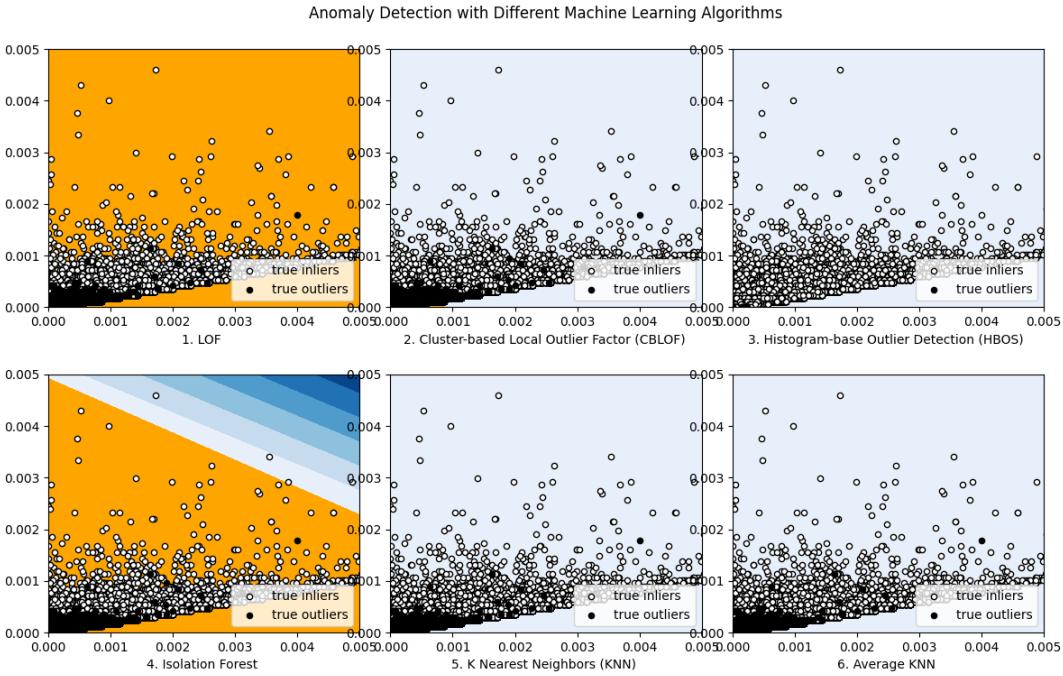


Figure 5.4: Anomaly detection results on database id 14

the monthly distribution of the given scenario. These figures indicate that there are certain patterns that can be understood based on these figures. All of these training are conducted using the specified data sets. Based on our analysis, we were able to determine that there are similar patterns among these individuals.

Based on different thresholds, we were able to find predictions for inliers and outliers at the end of our study. As a result of our analysis, we have been able to understand that our models are making good progress in the partitioning of money transaction data into normal and abnormal categories. As shown in the *Tables* A.15, A.16 and A.17 provided, combining several different algorithms creates better results in terms of accuracy.

5.3 Results

Thus, as described in the following *Table* 5.1, our experiment uses a general data set to compare all the performances. By using the measures Date, Transaction Direction, Amount, we are able to examine different scenarios and understand the anomaly scenario for the given field. Each of these measurements enables us to understand why we should focus on more than one anomaly case. In contrast to a multiple analysis, when we attempt to analyze transactions in a single direction, it gradually fails. Furthermore, we were able to observe that considering the amount of money and the amount of transactions gives a more generalized perspective on the problem according to the *Table* A.17. Therefore, increasing the anomaly score percentage of that model resulted in an increase in results. As a general rule, the amount of money and

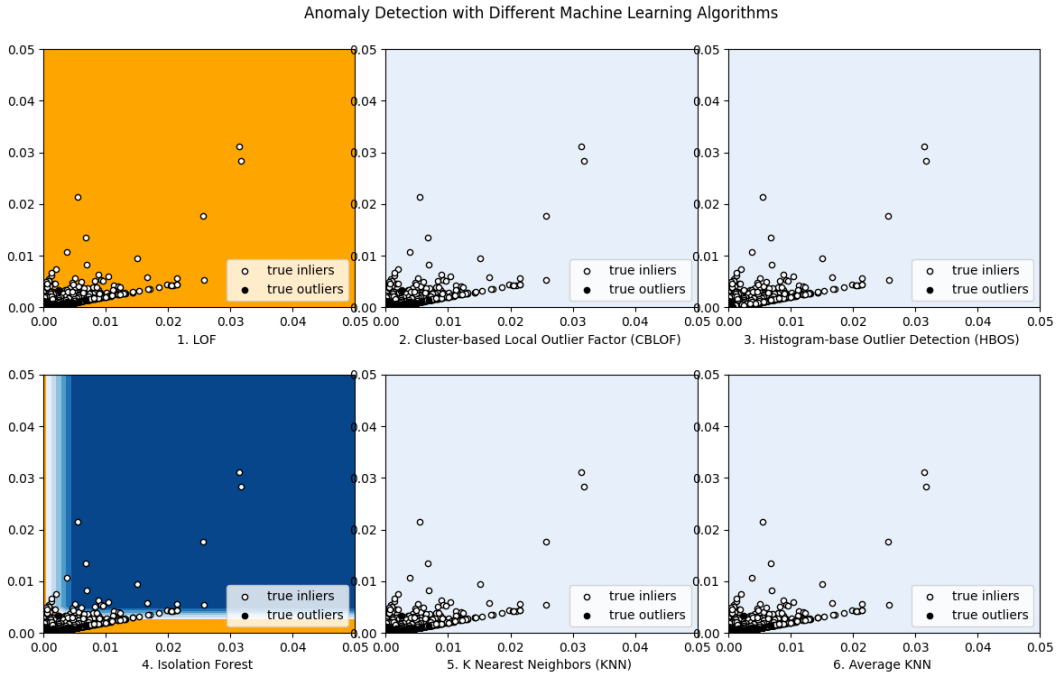


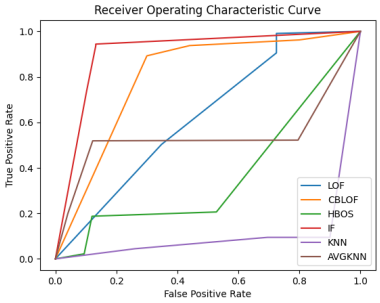
Figure 5.5: Anomaly detection results on database id 15

the transaction are more significant than the participant case, while the participant case is more significant than the individual case as described in the *Table* 5.5. The final step is to test another approach.

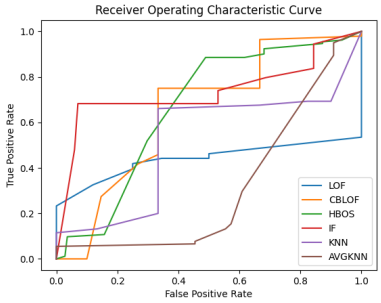
Table 5.1: Machine learning algorithm model AUC results over data sets

Databases	LOF	CBLOF	HBOS	IF	KNN	AVG KNN
1	0.439	0.650	0.664	**0.734**	0.517	0.316
2-4	0.346	**0.892**	0.602	0.574	0.371	0.244
5-7	0.439	0.650	0.664	**0.735**	0.517	0.316
8-10	0.307	**0.750**	0.266	0.500	0.220	0.125
11-15	0.626	0.802	0.368	**0.906**	0.110	0.539

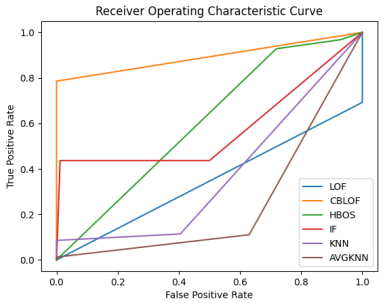
After completing all the experiments, we were able to produce different results for our scenarios using different machine learning algorithms. Our results indicate that if we do not use the IF algorithm, the ML algorithm alone would perform poorly as can be seen in *Table* 5.1. These results, however, indicate that the accuracy and expected anomaly scores are not sufficient because there are still uncovered anomalies. First, we began with the idea of finding these cases by using LSTM AEs. We have been able to establish a threshold that distinguishes anomalies from normal transactions based on our results presented in the tables A.15 and A.16. Our combination of models



(a) Curve for sender scenario



(b) Curve for participant scenario



(c) Curve For Receiver Scenario

Figure 5.6: ROC graph for ml algorithms

eliminates the disadvantage we suspected by identifying several anomalies we would have missed if we had used only one model in *Table* 5.5.

Through our research, we have been able to provide anomaly detection models based on LSTM AEs in a variety of different scenarios as well as their combinations. Despite not appearing in our literature review, the sequence-based approach to money transaction problems proved useful in terms of accuracy and global detection perspectives. Through our approach, we have also demonstrated the importance of understanding what an anomaly is. Several different approaches were used with different algorithms to miss the other scenarios without incorporating other models where their results are shown in *Table* A.16.

However, despite the small amount of improvement according to the *Table* 5.5, using this model on big data sets will enable us to identify more than thousands of cases in millions of transactions. We believe that the combination of our two models will enable us to improve the applications for money transactions in a real-life context.

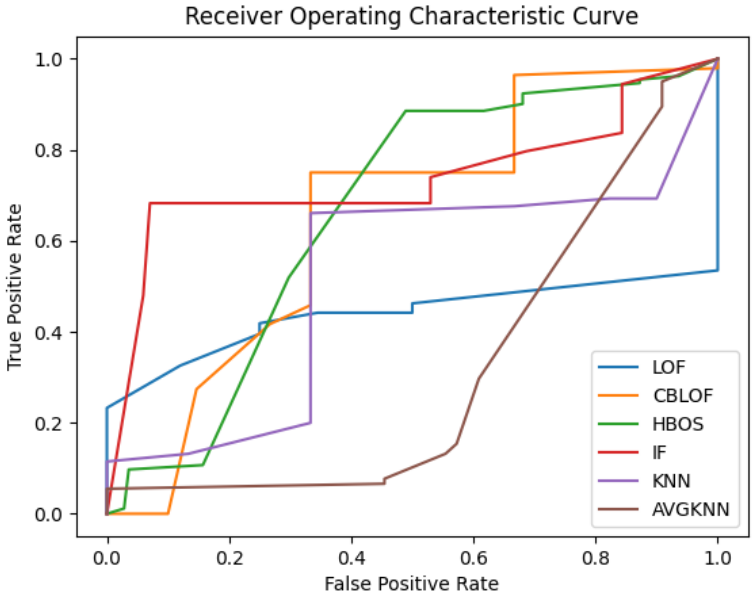


Figure 5.7: ROC curve for the general case

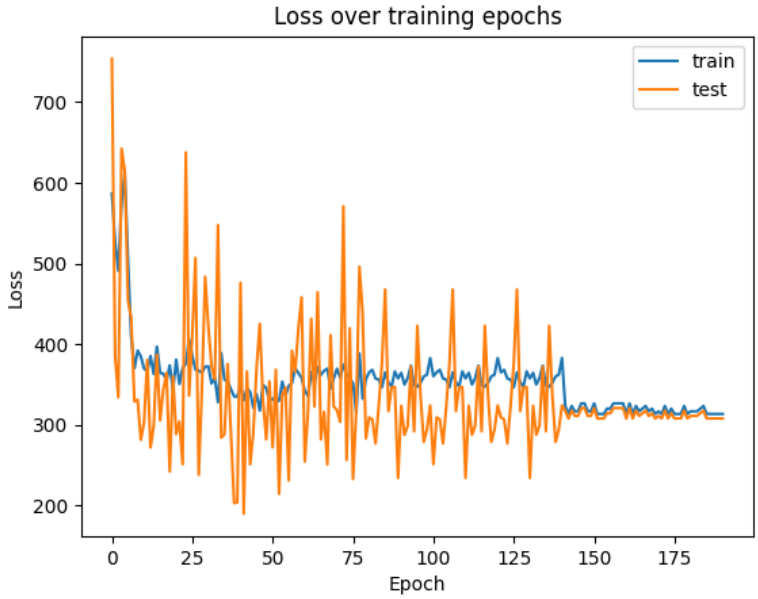


Figure 5.8: LSTM auto-encoder training history for total amount & transaction case

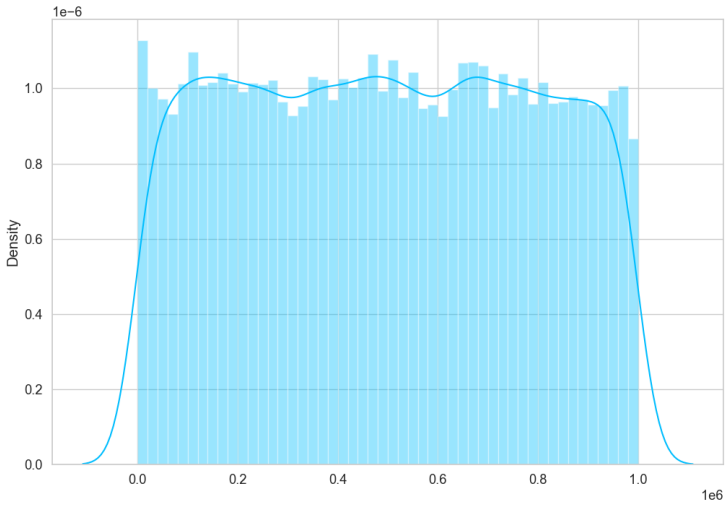
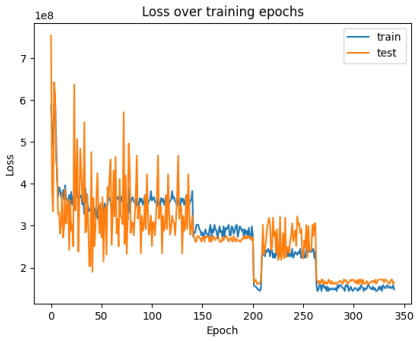
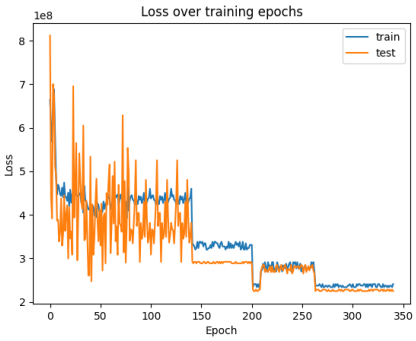


Figure 5.9: Anomaly distribution of the total amount & transaction case

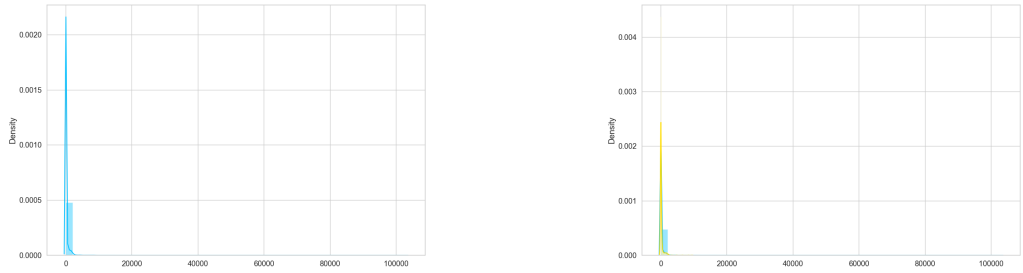


(a) LSTM auto-encoder training history for individual



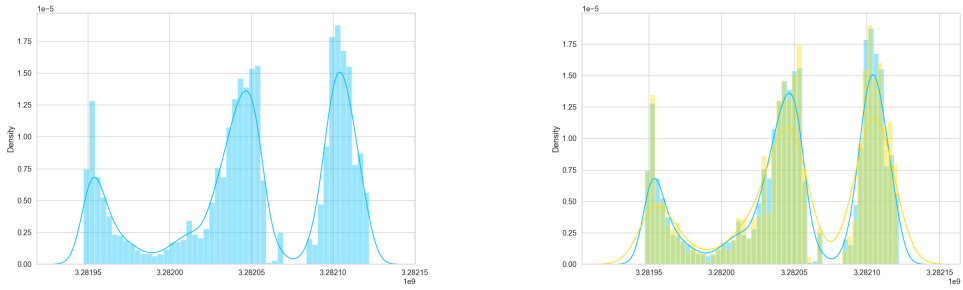
(b) LSTM autoencoder training history for sender/receiver

Figure 5.10: Training history of LSTM AE for the participant and sender/receiver case



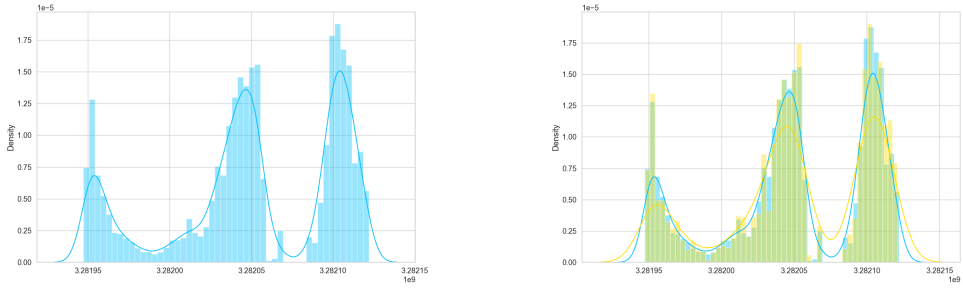
(a) Training Data (b) Test Data

Figure 5.11: LSTM AE results normal behavior distribution in total amount & transaction case



(a) Training data (b) Test data

Figure 5.12: LSTM AE results normal behavior distribution in participant case



(a) Training data (b) Test data

Figure 5.13: LSTM AE results normal behavior distribution in sender/receiver case

Table 5.2: Recommendation table on money transaction architecture

Algorithms	Accuracy	Speed	Global Detection
LOF	o	–	o
CBLOF	o	++	+
HBOS	–	++	–
IF	o	+	o
KNN	+	–	-
AVG-KNN	o	–	-
LSTM AE Case 1	+	–	+
LSTM AE Case 2	+	–	+
LSTM AE Case 3	+	–	+
Multiple LSTM AEs	++	–	++

On a qualitative basis, judgments range from very poor – over average (o) to very good (++) Alg.

Table 5.3: Computation time For different algorithms - 1

Databases	LOF	CBLOF	HBOS	IF
1	1.64	1.98	3.67	5.89
2-4	1.02	1.78	2.27	4.46
5-7	0.69	1.86	1.87	2.71
8-10	1.07	2.20	1.93	3.41
11-15	5892.76	18.04	2.22	116.63

Table 5.4: Computation time for different algorithms - 2

Databases	KNN	AVG KNN	LSTM AE	LSTM AEs
1	8.60	8.98	No Training	No Training
2-4	7.30	8.83	2.10 days	2.30 days
5-7	4.19	4.96	2.24 days	2.16 days
8-10	5.60	6.59	2.40 days	2.20 days
11-15	25569.26	1022.77	2.70 days	**2.80 days**

A machine learning algorithm measures in seconds, while a LSTM algorithm measures in days. Because of the computation time, machine learning models are tested 20 times each, while LSTM AEs could be tested three times each.

Table 5.5: Improvements of LSTM AEs using multiple scenarios on general data set

Rate of Improvement on the Least Accurate Model	LSTM Scneario Combinations
0.0025	1
0.0066	1 and 2
0.0054	2 and 3
0.0064	1 and 3

As shown in the table, scenario 1 represents the total amount as well as the transaction scenario, scenario 2 represents the participant base, and scenario 3 represents the receiver/sender case scenario.

CHAPTER 6

CONCLUSION AND FUTURE WORK

6.1 Conclusion

The objective of this work was to develop an anomaly detection architecture utilizing LSTM AEs and different machine learning algorithms in the given field. In the proposed method, two architectures are presented, an LSTM AE-based one and an ML and LSTM AE-based one. To begin with, several different models are trained for different scenarios and attempts are made to create anomaly scores that can be applied to the entire case. In order to obtain reliable results, each of these methods is combined according to a certain ratio. Unlike the first method, the second method uses a trained model after the first ML algorithm for more accurate results without any exceptions for anomalies.

An analysis of the performance of the architecture based on the trained models was conducted through a series of experiments. Data is collected from real-life transactions in order to conduct experiments. A subset of the data set was then created for each of the training models. According to results, focusing on different scenarios will yield better results with LSTM AE while using ML as a firsthand will result in no exception anomalies.

As a result of our experiments, we have been able to understand the following important points:

- A number of machine learning algorithms were used to identify anomalies. We have found that the isolation forest algorithm yields more accurate results in the general case.
- By analyzing the results presented with LSTM AE, it has been demonstrated that each network was able to learn the normal behavior of the transactions. As well as that, combining these network models enhances the results.
- As a result of our analysis, we found that analyzing total transactions and total money per person per day could provide more general results regarding accuracy. However, by combining this model with a participant-based model, we were able to enhance the performance of our networks as shown in the *Table* A.17.

The results of our research answered the questions that we outlined before we started our research, as described in the previous paragraph. Using deep learning and ma-

chine learning models, we were able to find anomaly detection results. As part of our research, we created different animal scenarios based on features and used them to train our models. Using this training and testing process, it was found that isolation forest-based machine learning algorithms are more suitable while LSTM AEs are improved by combining different scenario combinations. We found that our research showed good accuracy for anomaly cases generated during our research by using machine learning algorithms and LSTM AEs.

6.2 Future Work

As part of our future work, we will examine a greater variety of scenarios when training our models. Additionally, the detection process still takes a long time, even though we would be able to increase the accuracy of our models. In this type of transaction, anomaly detection must be fast due to the fact that it is streaming. However, as of right now, detecting anomalies in generated cases is not fast enough and will not be capable of working in real-time streaming data.

GPU resources will be used in future work. During the training of our models, we were unable to use more than 30000 data since we were using powerless resources as opposed to a GPU. In the event that GPUs are available, we would be able to generalize our LSTM AEs model by analyzing much more data so as to cover a greater number of anomaly cases at the end of the process.

REFERENCES

[1] C. C. Aggarwal, *Outlier Analysis*. Springer New York, 2013.

[2] D. M. Hawkins, *Identification of Outliers*. Springer Netherlands, 1980.

[3] C. K. Mohan and K. G. Mehrotra, "Anomaly detection in banking operations," *IDRBT Journal of Banking Technology*, vol. 01, pp. 16–48, 2017.

[4] A. E. Hassanien, M. Kilany, and E. H. Houssein, "Ecg signals classification: a review," *International Journal of Intelligent Engineering Informatics*, vol. 5, no. 4, p. 376, 2017.

[5] P. Liu, X. Sun, Y. Han, Z. He, W. Zhang, and C. Wu, "Arrhythmia classification of lstm autoencoder based on time series anomaly detection," *Biomedical Signal Processing and Control*, vol. 71, p. 103228, 01 2022.

[6] C. Heger, A. van Hoorn, M. Mann, and D. Okanović, "Application performance management: State of the art and challenges for the future," pp. 429–432, 04 2017.

[7] D. Elsner, P. Aleatrati Khosroshahi, A. Maccormack, and L. Robert, "Multivariate unsupervised machine learning for anomaly detection in enterprise applications," 01 2019.

[8] A. Elliott, M. Cucuringu, M. M. Luaces, P. Reidy, and G. Reinert, "Anomaly detection in networks with application to financial transaction networks," 2019.

[9] C. Guo and F. Berkhahn, "Entity embeddings of categorical variables," 2016.

[10] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: A survey," *ACM Comput. Surv.*, vol. 41, 07 2009.

[11] M. Ahmed, A. Naser Mahmood, and J. Hu, "A survey of network anomaly detection techniques," *Journal of Network and Computer Applications*, vol. 60, pp. 19–31, 2016.

[12] S. Yuan and X. Wu, "Trustworthy anomaly detection: A survey," 02 2022.

[13] D. Samariya and A. Thakkar, "A comprehensive survey of anomaly detection algorithms," *Annals of Data Science*, 11 2021.

[14] C. C. Aggarwal, *Applications of Outlier Analysis*, pp. 399–422. Cham: Springer International Publishing, 2017.

[15] L. Poon, S. Farshidi, N. Li, and Z. Zhao, "Unsupervised anomaly detection in data quality control," pp. 2327–2336, 12 2021.

[16] W. Hilal, S. A. Gadsden, and J. Yawney, "Financial fraud: A review of anomaly detection techniques and recent advances," *Expert Systems with Applications*, vol. 193, p. 116429, 2022.

[17] M. Siwach and S. Mann, "Anomaly detection for web log data analysis: A review," *Journal of Algebraic Statistics*, vol. 13, pp. 129–148, 05 2022.

[18] G. Nascimento and M. Correia, "Anomaly-based intrusion detection in software as a service," *Proceedings of the International Conference on Dependable Systems and Networks*, 06 2011.

[19] T. Fernando, H. Gammulle, S. Denman, S. Sridharan, and C. Fookes, "Deep learning for medical anomaly detection – a survey," 2020.

[20] R. Yu, H. Qiu, Z. Wen, C. Lin, and Y. Liu, "A survey on social media anomaly detection," *SIGKDD Explor. Newsl.*, vol. 18, p. 1–14, aug 2016.

[21] M. Flach, F. Gans, A. Brenning, J. Denzler, M. Reichstein, E. Rodner, S. Bathiany, P. Bodesheim, Y. Guanche, S. Sippel, and M. D. Mahecha, "Multivariate anomaly detection for earth observations: a comparison of algorithms and feature extraction techniques," *Earth System Dynamics*, vol. 8, no. 3, pp. 677–696, 2017.

[22] L. Ruff, J. R. Kauffmann, R. A. Vandermeulen, G. Montavon, W. Samek, M. Kloft, T. G. Dietterich, and K.-R. Muller, "A unifying review of deep and shallow anomaly detection," *Proceedings of the IEEE*, vol. 109, pp. 756–795, may 2021.

[23] S. Wang, J. F. Balarezo, S. Kandeepan, A. Al-Hourani, K. G. Chavez, and B. Rubinstein, "Machine learning in network anomaly detection: A survey," *IEEE Access*, vol. 9, pp. 152379–152396, 2021.

[24] M. Goldstein and S. Uchida, "A comparative evaluation of unsupervised anomaly detection algorithms for multivariate data," *PloS one*, vol. 11, p. e0152173, 04 2016.

[25] M. Manish Kumar and G. R. Ramya, "Performance comparison of anomaly detection algorithms," in *Inventive Communication and Computational Technologies* (G. Ranganathan, X. Fernando, and F. Shi, eds.), (Singapore), pp. 761–776, Springer Nature Singapore, 2022.

[26] K. S. Killourhy and R. A. Maxion, "Comparing anomaly-detection algorithms for keystroke dynamics," in *2009 IEEE/IFIP International Conference on Dependable Systems Networks*, pp. 125–134, 2009.

[27] K. L. Ingham and H. Inoue, "Comparing anomaly detection techniques for http," in *Recent Advances in Intrusion Detection* (C. Kruegel, R. Lippmann, and A. Clark, eds.), (Berlin, Heidelberg), pp. 42–62, Springer Berlin Heidelberg, 2007.

[28] K. S. Killourhy and R. A. Maxion, "Comparing anomaly-detection algorithms for keystroke dynamics," Jun 2009.

[29] F. T. Liu, K. M. Ting, and Z.-H. Zhou, "Isolation forest," in *2008 Eighth IEEE International Conference on Data Mining*, pp. 413–422, 2008.

[30] I. Jahan, M. Alam, F. Ahmed, and Y. M. Jang, "Anomaly detection in semiconductor cleanroom using isolation forest," pp. 795–797, 10 2021.

[31] K. Zhang, X. Kang, and S. Li, "Isolation forest for anomaly detection in hyperspectral images," pp. 437–440, 07 2019.

[32] H. Xu, G. Pang, Y. Wang, and Y. Wang, "Deep isolation forest for anomaly detection," 06 2022.

[33] S. Hariri, M. Kind, and R. Brunner, "Extended isolation forest with randomly

oriented hyperplanes," *IEEE Transactions on Knowledge and Data Engineering*, vol. PP, pp. 1–1, 10 2019.

[34] J. Lesouple, C. Baudoin, M. Spigai, and J.-Y. Tourneret, "Generalized isolation forest for anomaly detection," *Pattern Recognition Letters*, vol. 149, 06 2021.

[35] S. Ramaswamy, R. Rastogi, and K. Shim, "Efficient algorithms for mining outliers from large data sets," 2000.

[36] X. Gu, L. Akoglu, and A. Rinaldo, "Statistical analysis of nearest neighbor methods for anomaly detection," 2019.

[37] L. Bergman, N. Cohen, and Y. Hoshen, "Deep nearest neighbor anomaly detection," 2020.

[38] T. Tsigkritis, G. Groumas, and M. Schneider, "On the use of amp;lt;iamp;gt;kamp;lt;/iamp;gt;-nn in anomaly detection," 2018.

[39] M. Amer and M. Goldstein, "Nearest-neighbor and clustering based anomaly detection algorithms for rapidminer," 08 2012.

[40] M. Breunig, H.-P. Kriegel, R. Ng, and J. Sander, "Lof: Identifying density-based local outliers.," vol. 29, pp. 93–104, 06 2000.

[41] N. Paulauskas and F. Bagdonas, "Local outlier factor use for the network flow anomaly detection," *Security and Communication Networks*, vol. 8, 08 2015.

[42] Y. Linghu, M. Xu, X. Li, and H. Qian, "Weighted local outlier factor for detecting anomaly on in-vehicle network," pp. 479–487, 12 2020.

[43] T. Wang, W. Zhang, J. Wei, and H. Zhong, "Workload-aware online anomaly detection in enterprise applications with local outlier factor," in *2012 IEEE 36th Annual Computer Software and Applications Conference*, pp. 25–34, 2012.

[44] M. Goldstein and A. Dengel, "Histogram-based outlier score (hbos): A fast unsupervised anomaly detection algorithm," 09 2012.

[45] N. Paulauskas and A. Baskys, "Application of histogram-based outlier scores to detect computer network anomalies," *Electronics*, vol. 8, p. 1251, 11 2019.

[46] A. Kind, M. Stoecklin, and X. Dimitropoulos, "Histogram-based traffic anomaly detection," *Network and Service Management, IEEE Transactions on*, vol. 6, pp. 110 – 121, 07 2009.

[47] M. B. Aliyu, A. Amr, and I. S. Ahmad, "Anomaly detection in wearable location trackers for child safety," *Microprocessors and Microsystems*, vol. 91, p. 104545, 2022.

[48] Z. He, X. Xu, and S. Deng, "Discovering cluster based local outliers," *Pattern Recognition Letters*, vol. 24, pp. 1641–1650, 06 2003.

[49] G. Zengan, "Application of cluster-based local outlier factor algorithm in anti-money laundering," pp. 1 – 4, 10 2009.

[50] S. Khokhar, G. Wang, R. Cottrell, and T. Anwar, "Detecting anomalies from end-to-end internet performance measurements (pinger) using cluster based local outlier factor," pp. 982–989, 12 2017.

[51] M. Karimi, A. Jahanshahi, A. Mazloumi, and H. Zamani Sabzi, "Border gateway protocol anomaly detection using neural network," pp. 6092–6094, 12 2019.

[52] B. I. C. Education, "What are recurrent neural networks?."

[53] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, pp. 1735–80, 12 1997.

[54] S. Hochreiter, "Untersuchungen zu dynamischen neuronalen netzen," 04 1991.

[55] I. Skarga-Bandurova, T. Biloborodova, I. Skarha-Bandurov, Y. Boltov, and M. Derkach, *A Multilayer LSTM Auto-Encoder for Fetal ECG Anomaly Detection*, vol. 285. 10 2021.

[56] P. Malhotra, A. Ramakrishnan, G. Anand, L. Vig, P. Agarwal, and G. Shroff, "Lstm-based encoder-decoder for multi-sensor anomaly detection," 2016.

[57] B. Radford, L. Apolonio, A. Trias, and J. Simpson, "Network traffic anomaly detection using recurrent neural networks," 03 2018.

[58] S. Latif, M. Usman, J. Qadir, and R. Rana, "Abnormal heartbeat detection using recurrent neural networks," 01 2018.

[59] Y. Wei, J. Jang-Jaccard, W. Xu, F. Sabrina, S. Camtepe, and M. Boulic, "Lstm-autoencoder based anomaly detection for indoor air quality time series data," 04 2022.

[60] M. Salahuddin, M. F. Bari, H. Alameddine, V. Pourahmadi, and R. Boutaba, "Time-based anomaly detection using autoencoder," 11 2020.

[61] Z. Zhao, C. Xu, and B. Li, "A lstm-based anomaly detection model for log analysis," *Journal of Signal Processing Systems*, vol. 93, pp. 1–7, 07 2021.

[62] B. Dumitrescu, A. Baltoiu, and S. Budulan, "Anomaly detection in graphs of bank transactions for anti money laundering applications," *IEEE Access*, vol. 10, pp. 47699–47714, 2022.

[63] H. Jihal, S. Ounacer, M. Ghomari, H. Elbour, and M. Azzouazi, *Anomaly Detection: Case of Mobile Money Transactions with Isolation Forrest*, pp. 659–665. 01 2022.

[64] S. Ounacer, H. Jihal, S. Ardchir, and M. Azzouazi, *Anomaly Detection in Credit Card Transactions*, pp. 132–140. 02 2020.

[65] X.-G. Zhou and L.-Q. Zhang, "Abnormal event detection using recurrent neural network," pp. 222–226, 11 2015.

[66] M. Desai, R. Mehta, and D. Rana, "A survey on techniques for indexing and hashing in big data," 07 2019.

[67] G. Van Rossum and F. L. Drake, *Python 3 Reference Manual*. Scotts Valley, CA: CreateSpace, 2009.

[68] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Köpf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "Pytorch: An imperative style, high-performance deep learning library," 2019.

[69] Y. Zhao, Z. Nasrullah, and Z. Li, "Pyod: A python toolbox for scalable outlier detection," *Journal of Machine Learning Research*, vol. 20, no. 96, pp. 1–7, 2019.

[70] T. Fawcett, "An introduction to roc analysis," *Pattern Recognition Letters*, vol. 27, no. 8, pp. 861–874, 2006. ROC Analysis in Pattern Recognition.

[71] European Mathematical Society, "Simpson formula," in *Encyclopedia of Mathematics*, EMS Press, Feb. 2021.

[72] C. C. Aggarwal, *Proximity-Based Outlier Detection*, pp. 111–147. Cham: Springer International Publishing, 2017.

[73] S. Han, X. Hu, H. Huang, M. Jiang, and Y. Zhao, "Adbench: Anomaly detection benchmark," 06 2022.

[74] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[75] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014.

APPENDIX A

TABLES OF EXPERIMENT RESULTS

All of the results reported in the following tables are the average of 20 experiments each.

Table A.1: Analysis of the data set id 2 with an outlier contamination score 0.005 using different machine learning algorithms

Amount of Outliers	Amount of Inliers	Algorithm Name	Total Time(Seconds)
127	29873	LOF	1.56
138	29862	CBLOF	1.99
136	29864	HBOS	2.47
144	29856	IF	6.29
95	29905	KNN	11.27
45	29955	AVG KNN	13.76

Table A.2: Analysis of the data set id 3 with an outlier contamination score 0.005 using different machine learning algorithms

Amount of Outliers	Amount of Inliers	Algorithm Name	Total Time(Seconds)
136	29864	LOF	0.72
141	29859	CBLOF	1.87
143	29857	HBOS	2.19
148	29852	IF	3.48
92	29908	KNN	5.24
37	29963	AVG KNN	6.29

Table A.3: Analysis of the data set id 4 with an outlier contamination score 0.005 using different machine learning algorithms

Amount of Outliers	Amount of Inliers	Algorithm Name	Total Time(Seconds)
140	29860	LOF	0.79
128	29872	CBLOF	1.49
135	29865	HBOS	2.17
150	29850	IF	3.61
97	29903	KNN	5.39
38	29962	AVG KNN	6.45

Table A.4: Analysis of the data set id 5 with an outlier contamination score 0.005 using different machine learning algorithms

Amount of Outliers	Amount of Inliers	Algorithm Name	Total Time(Seconds)
156	29844	LOF	0.36
137	29867	CBLOF	1.56
128	29872	HBOS	2.03
149	29851	IF	2.01
103	29897	KNN	2.91
40	29960	AVG KNN	3.14

Table A.5: Analysis of the data set id 6 with an outlier contamination score 0.005 using different machine learning algorithms

Amount of Outliers	Amount of Inliers	Algorithm Name	Total Time(Seconds)
130	29870	LOF	0.98
139	29861	CBLOF	1.56
138	29862	HBOS	1.80
150	29850	IF	2.70
110	29890	KNN	4.71
39	29961	AVG KNN	5.68

Table A.6: Analysis of the data set id 7 with an outlier contamination score 0.005 using different machine learning algorithms

Amount of Outliers	Amount of Inliers	Algorithm Name	Total Time(Seconds)
126	29874	LOF	0.74
147	29853	CBLOF	2.46
139	29861	HBOS	1.77
149	29851	IF	3.43
85	29915	KNN	4.96
32	29968	AVG KNN	6.08

Table A.7: Analysis of the data set id 8 with an outlier contamination score 0.005 using different machine learning algorithms

Amount of Outliers	Amount of Inliers	Algorithm Name	Total Time(Seconds)
140	29860	LOF	1.01
146	29854	CBLOF	2.34
134	29866	HBOS	1.80
144	29856	IF	3.31
108	29892	KNN	5.24
59	29941	AVG KNN	6.29

Table A.8: Analysis of the data set id 9 with an outlier contamination score 0.005 using different machine learning algorithms

Amount of Outliers	Amount of Inliers	Algorithm Name	Total Time(Seconds)
130	29870	LOF	1.46
147	29853	CBLOF	2.70
138	29862	HBOS	1.89
149	29851	IF	3.86
110	29890	KNN	5.79
39	29961	AVG KNN	6.79

Table A.9: Analysis of the data set id 10 with an outlier contamination score 0.005 using different machine learning algorithms

Amount of Outliers	Amount of Inliers	Algorithm Name	Total Time(Seconds)
112	29888	LOF	0.75
127	29873	CBLOF	1.56
146	29854	HBOS	2.11
150	29850	IF	3.07
103	29897	KNN	5.65
44	29956	AVG KNN	6.70

Table A.10: Analysis of the data set id 11 with an outlier contamination score 0.005 using different machine learning algorithms

Amount of Outliers	Amount of Inliers	Algorithm Name	Total Time(Seconds)
109	29891	LOF	1.17
143	29857	CBLOF	1.87
77	29923	HBOS	1.31
149	29851	IF	2.77
131	29869	KNN	4.66
124	29876	AVG KNN	6.09

Table A.11: Analysis of the data set id 12 with an outlier contamination score 0.005 using different machine learning algorithms

Amount of Outliers	Amount of Inliers	Algorithm Name	Total Time(Seconds)
640	149360	LOF	17.55
750	149250	CBLOF	3.65
107	149893	HBOS	1.49
750	149250	IF	16.67
750	149354	KNN	37.94
646	149418	AVG KNN	43.28

Table A.12: Analysis of the data set id 13 with an outlier contamination score 0.005 using different machine learning algorithms

Amount of Outliers	Amount of Inliers	Algorithm Name	Total Time(Seconds)
2156	497844	LOF	156.71
2499	497501	CBLOF	10.91
295	499705	HBOS	1.59
2496	497504	IF	51.31
2140	497860	KNN	224.66
1928	498072	AVG KNN	238.86

Table A.13: Analysis of the data set id 14 with an outlier contamination score 0.005 using different machine learning algorithms

Amount of Outliers	Amount of Inliers	Algorithm Name	Total Time(Minutes)
6019	1493981	LOF	50.14
7497	1492503	CBLOF	0.45
307	1499693	HBOS	0.03
7500	1492500	IF	3.14
6585	1493415	KNN	62.90
5931	1494069	AVG KNN	63.46

Table A.14: Analysis of the data set id 15 with an outlier contamination score of 0.005 using different machine learning algorithms

Amount of Outliers	Amount of Inliers	Algorithm Name	Total Time(Hours)
12970	2987030	LOF	7.3000
15000	2985000	CBLOF	0.0130
498	2999502	HBOS	0.0007
14894	2985106	IF	0.0900
12958	2987042	KNN	5.9800
11869	2988131	AVG KNN	6.0500

Table A.15: Training results of LSTM AE which trained with no abnormal data on different data sets with threshold value = 26

Ratio of Inliers	Threshold Value	Data Set Id	Number of Epoch
45/1498	26	11	150
258/7425	26	12	150
793/24750	26	13	150
2482/74250	26	14	150
4968/148500	26	15	150

Table A.16: Training results of LSTM AE which trained with no abnormal data on different data sets with threshold value = 1500

Ratio of Inliers	Threshold Value	Data Set Id	Number of Epoch
1485/1498	1500	11	150
7415/7425	1500	12	150
24727/24750	1500	13	150
74189/74250	1500	14	150
148370/148500	1500	15	150

Table A.17: Test results of LSTM AEs combinations which trained with no abnormal data on general the data set on different scenarios with threshold value = 1500

Ratio of Outliers	Threshold Value	LSTM Scneario Combinations
29800/30000	1500	1
29725/30000	1500	2
29736/30000	1500	3
29922/30000	1500	1 and 2
29886/30000	1500	2 and 3
29918/30000	1500	1 and 3

As shown in the table, scenario 1 represents the total amount as well as the transaction scenario, scenario 2 represents the participant base, and scenario 3 represents the receiver/sender case scenario.