

**ANOMALY DETECTION IN FINANCIAL  
TRANSACTIONS USING DEEP LEARNING  
METHOD**

**A MINOR PROJECT REPORT**

*Submitted by*

**Jaswanth V      [RA2111027020032]  
Rajavarman G    [RA2111027020038]  
Chavali Teja     [RA2111027020058]**

**Under the guidance of**

**Mrs.P.Jayalakshmi, M.E.,(Ph.D).,**

**(Assistant Professor, Department of Computer Science and Engineering)**

*in partial fulfilment for the award of the degree*

*of*

**BACHELOR OF TECHNOLOGY**

*in*

**COMPUTER SCIENCE AND ENGINEERING**

**of**

**FACULTY OF ENGINEERING AND TECHNOLOGY**



**SRM INSTITUTE OF SCIENCE AND TECHNOLOGY**

**RAMAPURAM, CHENNAI -600089**

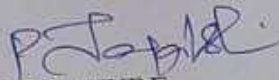
**NOVEMBER 2024**

**SRM INSTITUTE OF SCIENCE AND  
TECHNOLOGY**


(Deemed to be University U/S 3 of UGC Act, 1956)

**BONAFIDE CERTIFICATE**

Certified that this minor project report titled "ANOMALY DETECTION IN FINANCIAL TRANSACTIONS USING DEEP LERANING METHOD" is the bonafide work of JASWANTH V [REGNO: RA2111027020032], RAJAVARMAN G [REGNO: RA2111027020038], CHAVALI TEJA [REGNO: RA2111027020058] who carried out the minor project work under my supervision. Certified further, that to the best of my knowledge, the work reported herein does not form any other project report or dissertation on the basis of which a degree or award was conferred on an occasion on this or any other candidate.


  
SIGNATURE

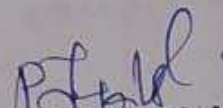
Mrs. P. Jayalakshmi, M.E., (Ph.D).,  
Assistant Professor  
Department of Computer Science and  
Engineering,  
SRM Institute of Science and Technology,  
Ramapuram, Chennai.

  
SIGNATURE

Dr. K. RAJA, M.E., Ph.D.,  
Professor and Head  
Department of Computer Science and  
Engineering,  
SRM Institute of Science and Technology,  
Ramapuram, Chennai.

Submitted for the project viva-voce held on 06-11-2024 at SRM Institute of Science and Technology, Ramapuram, Chennai 600089.

  
INTERNAL EXAMINER 1  
6/11/24

  
INTERNAL EXAMINER 2

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY  
RAMAPURAM, CHENNAI - 89

DECLARATION

We hereby declare that the entire work contained in this minor project report titled "Anomaly Detection in Financial Transactions using Deep Learning method" has been carried out by JASWANTH V [RA2111027020032], RAJAVARMAN G [RA2111027020038], CHAVALI TEJA [RA2111027020058] at SRM Institute of Science and Technology, Ramapuram Campus, Chennai- 600089, under the guidance of Mrs. P. Jayalakshmi, M.E,(Ph.D), Assistant Professor, Department of Computer Science and Engineering.

Place: Chennai

Date: 30/10/24

  
JASWANTH V

  
RAJAVARMANG

  
CHAVALITEJA





# Own Work Declaration Department of Computer Science and Engineering

SRM Institute of Science & Technology

## Own Work Declaration Form

This sheet must be filled in (each box ticked to show that the condition has been met). It must be signed and dated along with your student registration number and included with all assignments you submit – work will not be marked unless this is done.

To be completed by the student for all assessments

Degree/ Course : B. Tech  
Student Name : JASWANTH V, RAJAVARMAN G, CHAVELI TEJA  
Registration Number : RA2111027020032, RA2111027020038, RA2111027020058  
Title of Work : Anomaly Detection in Financial Transaction using Deep Learning method

I / We hereby certify that this assessment complies with the University's Rules and Regulations relating to Academic misconduct and plagiarism, as listed in the University Website, Regulations, and the Education Committee guidelines.

I / We confirm that all the work contained in this assessment is my / our own except where indicated, and that I / We have met the following conditions.

- Clearly references / listed all sources as appropriate
  - Referenced and put in inverted commas all quoted text (from books, web, etc.)
  - Given the sources of all pictures, data etc. that are not my own
  - Not made any use of the report(s) or essay(s) of any other student(s) either past or present
  - Acknowledged in appropriate places any help that I have received from others (e.g. fellow students, technicians, statisticians, external sources)
  - Compiled with any other plagiarism criteria specified in the Course handbook / University website
- I understand that any false claim for this work will be penalized in accordance with the University policies and regulations.


### DECLARATION:

I am aware of and understand the University's policy on Academic misconduct and plagiarism and I certify that This assessment is my / our own work, except where indicated by referring, and that I have followed the good academic practices noted above.

If you are working in a group, please write your registration numbers and sign with the date for every student in your group.

  
RA2111027020032

  
RA2111027020038

  
RA2111027020058

## ACKNOWLEDGEMENT

We place on record our deep sense of gratitude to our lionized Chairman **Dr.R.SHIVAKUMAR** for providing us with therequisite infrastructure throughout the course.

We take the opportunity to extend our hearty and sincere thanks toour **Dean, Dr.M. SAKTHI GANESH., Ph.D.**, for maneuvering us into accomplishing the project.

We take the privilege to extend our hearty and sincere gratitude to the **Professor and Head of the Department, Dr.K.RAJA,M.E.,Ph.D.**, for his suggestions, support and encouragement towards the completion of the project with perfection.

We express our hearty and sincere thanks to our guide **Mrs.P. Jayalakshmi M.E., (Ph.D)**, **Assistant Professor**, Computer Science and Engineering Department forher encouragement, consecutive criticism and constant guidance throughout this project work.

Our thanks to the teaching and non-teaching staff of the Computer Science and Engineering Department of SRM Institute of Science and Technology, Ramapuram Campus, for providing necessary resources forour project.

## **ABSTRACT**

Anomaly detection is highly critical in various domains such as cybersecurity, finance, and healthcare, and it can detect unusual patterns in data that may prevent many losses. This paper goes in the direction of how deep learning is applied in anomaly detection and discusses the capability of Autoencoders, Variational Autoencoders (VAEs), and Generative Adversarial Networks (GANs). Anomaly detection in these models is done through unsupervised learning, finding anomalies by learning complex representations and patterns in data. The experimental results on benchmark datasets show that the deep learning methods typically outperform the classical approaches in accuracy, robustness, and scalability. Deep learning might need larger datasets of normal instances, but networks learn how to classify such patterns as deviant and thereby may indicate potential anomalies. Deep learning techniques can be designed to handle high-dimensional data along with unstructured information, such as images, text, and time-series. Hence, deep learning is very effective in applications on fraud detection, cybersecurity, and industrial monitoring.

## TABLE OF CONTENTS

Chapter No	Title	Page.No
	<b>ABSTRACT</b>	<b>v</b>
	<b>LIST OF FIGURES</b>	<b>xi</b>
	<b>LIST OF TABLES</b>	<b>xii</b>
	<b>LIST OF ACRONYMS AND ABBREVIATIONS</b>	<b>x</b>
<b>1</b>	<b>INTRODUCTION</b>	<b>1</b>
1.1	Introduction	1
1.2	Problem Statement	2
1.3	Objective	2
1.4	Project Domain	3
1.5	Scope of the Project	3
1.6	Methodology	3
<b>2</b>	<b>LITERATURE SURVEY</b>	<b>4</b>
<b>3</b>	<b>PROJECT DESCRIPTION</b>	<b>6</b>
3.1	Existing System	6
3.2	Proposed System	6
3.3	Advantages	7
3.4	Feasibility Study	7
3.4.1	Effectiveness in handling high-dimensional data	8
3.4.2	Handling Imbalanced datasets	8
3.4.3	Scalability and Real time processing	8

3.5	System Specification	8
3.5.1	Hardware Specification	8
3.5.2	Software Specification	9
<b>4</b>	<b>PROPOSED SYSTEM</b>	<b>10</b>
4.1	Introduction	10
4.2	Dataset	10
4.3	Architecture Diagram	11
4.4	Modules Description	11
4.4.1	Module for Data Preprocessing	11
4.4.2	Module for Feature Extraction	12
4.4.3	Module Autoencoder	12
4.4.4	SVM Classification Module	12
4.4.5	Optimization Module (SGD)	12
<b>5</b>	<b>IMPLEMENTATION AND TESTING</b>	<b>13</b>
5.1	Understanding the Components	13
5.1.1	Input Phase	13
5.1.2	Training Phase	15
5.1.3	Evaluation Phase	15
5.2	Testing	17
5.2.1	Unit Testing	17
5.2.2	Integration Testing	17
5.2.3	Functional Testing	18
5.3	Test Results	19



<b>6</b>	<b>RESULTS AND DISCUSSIONS</b>	<b>20</b>
<b>7</b>	<b>CONCLUSION AND FUTURE ENHANCEMENTS</b>	<b>22</b>
7.1	Conclusion	22
7.2	Future Enhancements	22
<b>8</b>	<b>SOURCE CODE</b>	<b>23</b>
8.1	Sample Code	23
	<b>References</b>	<b>39</b>
	<b>Appendix</b>	
	<b>Proof of Publication/Patent filed/ Conference Certificate</b>	

## LIST OF FIGURES

<b>Fig No</b>	<b>Figure Name</b>	<b>PageNo</b>
4.1	Architecture Diagram	15
5.1	Input Data	19
6.1	Individual Transactions view	21
6.2	Individual Transactions view	21

## **LIST OF ACRONYMS AND ABBREVIATIONS**

SGD	Stochastic Gradient Descent
SVM	Support Vector Machine
LSTM	Long Short Term Memory

# CHAPTER 1

## INTRODUCTION

### 1.1 Introduction

Anomaly detection in financial transactions is very important because it could help to reveal irregularities or outliers that might indicate fraud, errors, or unusual patterns to be focused upon. In the financial system, it processes millions of transactions daily, and this kind of high-volume data is very challenging to detect anomalies from but essential at the same time. Traditional anomaly detection techniques, whether rule-based or based on statistical models fail to provide detection concerning complex patterns or adaptivity over time. Deep learning approaches promise a more appropriate and robust solution to these problems. Deep learning models, with neural networks in particular, have gained favor in anomaly detection because they can learn from large datasets, and pattern classification is very nonlinear. In particular, autoencoders, RNNs, and CNNs can be modeled over normal transaction behavior. Once trained, the model starts flagging transactions abnormal and drastically deviating from normal behavior.

Autoencoders are one of the deepest models used to apply unsupervised anomaly detection in financial transactions. Such models compress the input data into a smaller representation, and then try to reconstruct the original input from this compressed form. Hence, trained on normal transactions, autoencoders work with financial transaction data. Given that the input is non-recurrent in nature, mapping anomalies to a regime of high reconstruction errors is very common, thus pointing out potential problems. Given the rarity of financial frauds compared with legitimate transactions, autoencoders are excellent candidates for anomaly detection without labeled fraudulent data.

While there are promising results that have been shown with deep learning methods on anomaly detection for financial transactions, there are a few challenges. Critical areas that need improvement in this area include scarcity of fraudulent labeled data, which is a major bottleneck for anomaly detection by supervised deep learning models. Deep learning models are not easily explainable on why such transactions become anomalous; thus, better model interpretability improvements and incorporation of domain knowledge into deep learning frameworks are absolutely essential before such approaches can be widely adopted to real financial systems.

## **1.2 Problem Statement**

Deep learning techniques are being applied in anomaly detection in financial transactions. Fraudulent activities or some kinds of patterns can cause huge losses in a financial system. Thus, finding outlier behavior or deviation from normal transaction behavior is highly relevant to avoid all kinds of losses triggered by these illegal practices in transactions. Autoencoders, which are deep unsupervised neural networks, are primarily trained to learn a compressed representation of usual transaction data, and through reconstruction errors, detect anomalies. It uses strong classification capability through SVM and identifies a hyperplane that separates normal from anomalous transactions. The system is designed to be highly accurate and scalable, improving financial transaction security and reliability by using Autoencoders for feature extraction and SVM for anomaly detection.

## **1.3 Objective of the Project**

To identify unusual or fraudulent activities by learning complex patterns in transaction data, improving the accuracy and efficiency of detecting suspicious behavior. This approach aims to minimize false positives and enhance real-time fraud detection.

## **1.4 Project Domain**

The domain of the project is Deep Learning. The progress of Deep learning techniques enhances accuracy and efficiency in detecting outliers, ensuring robust financial security and risk management.

## **1.5 Scope of the Project**

It will involve training neural networks on historical transaction data to distinguish normal patterns from potential threats. The project will evaluate the model's effectiveness in real-time fraud detection, ensuring high accuracy and minimal false positives.

## **1.6 Methodology**

Deep learning techniques-based anomaly detection in financial transactions usually identify unusual transaction patterns that indicate a possible fraud or anomaly. Methodologically, the technique initiates with data preprocessing, this includes missing values handling, features normalization of the transactions, and class imbalance since anomalies like fraud are sparse. Typically, the unsupervised technique of Autoencoders or more specifically Variational Autoencoders (VAEs) is used to learn compressed representations of normal transaction behavior. During the inference step, those transactions are flagged as anomalies for which the reconstruction error is high. With the labeled anomaly data, One-Class SVMs or the supervised techniques using LSTMs/CNNs can be used. The further development of the detection process is done by using the post-processing methods like threshold setting or clustering, and model evaluation metrics like precision, recall, and F1-score quantify the performance.



## **CHAPTER 2**

### **LITERATURE SURVEY**

Traditional Rule-Based Approaches: - Paper: "Credit Card Fraud Detection: A Realistic Modelling and a Novel Learning Strategy" by Ahmed et al. (2016) Paper discusses traditional rule-based approaches for credit card fraud detection and introduces a new learning strategy based on cost-sensitive learning.[1]

Anomaly Detection Techniques: -Paper: "Isolation Forest" by Liu et al. (2008) This paper introduces the Isolation Forest algorithm, which is an effective anomaly detection method widely used in fraud detection due to its ability to handle high-dimensional data.[2]

Machine Learning for Fraud Detection: - Paper: "A Survey of Fraud Detection Techniques" by Bhattacharyya et al. (2019) - Summary: This comprehensive survey paper provides an overview of various machine learning techniques applied to fraud detection, including ensemble methods, deep learning, and feature engineering.[3]

Deep Learning and Neural Networks A Survey" by Chalapathy et al. (2019) - Summary: This survey paper explores the use of deep learning models, such as autoencoders and recurrent neural networks, for anomaly detection, including their application in fraud detection.[4]

Real-Time Fraud Detection: Paper: "Online Anomaly Detection in Cloud Data Streams" by Xu et al. (2013) Summary: This paper focuses on real-time fraud detection in data streams, a critical requirement for financial institutions and online payment platforms[5]

Explainable AI (XAI) in Fraud Detection: Paper: "Explainable AI for Fraud Detection: A Review" by Xu et al. (2021) Summary: This review paper explores the importance of explainable AI techniques in fraud detection to enhance transparency and trust in automated fraud detection systems.[6]

Blockchain and Fraud Prevention: Paper by Zheng et al. (2017) Summary: This paper discusses the potential of blockchain technology in preventing fraud by providing transparency, immutability, and traceability of financial transactions.[7]

Emerging Challenges and Future Directions: Paper: "Future Research Directions in

Cyber Crime" by Holt et al. (2019) Summary: This paper outlines emerging challenges in cybercrime and fraud, including the need for advanced techniques to detect new forms of fraud in the era of digital innovation.[8]

N. Srivastava, G. E. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting", *The Journal of Machine Learning Research*, 15(1), 1929- 1958, 2014.[9]

B. Xu, N. Wang, T. Chen, and M. Li, "Empirical Evaluation of Rectified Activations in Convolution Network", *ICML Deep Learning Workshop*, pages 1–5, 2015. malicious traffic has drawn increased attention. Most well-known offensive security assessment today's are heavily focused on pre-compromise. The amount of anomalous data in today's context is massive. Analyzing the data using primitive methods would be highly challenging. Solution to it is: If we can detect adversary behaviors in the early stage of compromise, one can prevent and safeguard themselves from various attacks including ransomwares and Zero-day attacks.[10]

Kingma and J. Ba, "Adam: A method for stochastic optimization", *International Conference on Learning Representations (ICLR)*. 2015. Research on Time Series Anomaly Detection Algorithm and Application 2019 IEEE 4th Advanced Information Technology, Electronic and Automation Control Conference (IAEAC).[11]

2022 19th International SoC Design Conference (ISOCC) Performance Comparison of Soiling Detection Using Anomaly Detection Methodology. To protect a network, a strong intrusion detection system (IDS) is necessary. Reviewing the literature reveals that while some research has been conducted in this area, a thorough and in-depth investigation has not yet been carried out.[12]

DeepWindow: An Efficient Method for Online Network Traffic Anomaly Detection 2019 IEEE 21st International Conference on High Performance Computing and Communications; IEEE 17th International Conference on Smart City; IEEE 5th International Conference on Data Science and Systems (HPCC/SmartCity) [13]

Nupur Sawarkar, Pranay Yenagandula, Devang Shetye and Prof. Shruti Agrawal, "Expense Manager: An Expense Tracking Application using Image Processing", *International Research Journal of Engineering and Technology (IRJET)*, vol.09,no.04,pp.2395-0056.[14]

## **CHAPTER 3**

### **PROJECT DESCRIPTION**

#### **3.1 Existing System**

Methods for anomaly detection in financial transactions based on deep learning techniques typically detect unusual patterns of transactions that may denote fraud or any anomalies. Methodologically, this technique begins with data preprocessing, which involves missing values handling, features normalization of the transactions, and class imbalance since fraud, for example, is sparse. The unsupervised technique used typically is Autoencoders, or in many scenarios, Variational Autoencoders (VAEs), to learn compressed representations of normal transaction behavior. Such transactions are tagged as anomalies in the inference step, and the reconstruction error is high. One-Class SVMs or the supervised methods using LSTMs/CNNs can be utilized with the labeled anomaly data. The detection process is further evolved using the post-processing methodologies of threshold setting or clustering, and it measures performance in terms of precision, recall, and F1-score.

#### **3.2 Proposed System**

A deep-learning-based anomaly detection system for financial transactions could be based upon a hybrid approach such as Autoencoder and Support Vector Machine (SVM). The system would begin by utilizing an unsupervised autoencoder to learn the normal patterns of transaction data. Auto-encoders are neural networks that are designed to reconstruct their input data and minimize the reconstruction error on normal data, which makes them efficient at detecting anomalies. These transactions, which contain the highest reconstruction errors, most deviate from the patterns learned, therefore labeling them as potential anomalies. These high-error transactions are fed into a very strong supervised SVM to classify them better as normal or fraudulent using the high-dimensional features learned by the Autoencoder. A combination of Autoencoder with SVM creates a robust multistage model wherein Autoencoder does the feature extraction as well as outlier detections.

### 3.3 Advantages

**High Accuracy:** Ensures precise predictions and minimizes errors in outcomes.

**Unsupervised Learning:** Operates without labeled data, uncovering hidden patterns independently.

**Scalability:** Handles large datasets efficiently, supporting growth without loss in performance.

**Adaptive to Dynamic Data:** Adjusts to changes in real-time data, maintaining relevance.

**Reduced False Positives:** Minimizes incorrect positive identifications, improving trust in results.

### 3.4 Feasibility Study

A Feasibility study is carried out to check Autoencoders and SVMs, is feasible due to their ability to model complex patterns and identify outliers.

- Effectiveness in Handling High-Dimensional Data
- Handling Imbalanced Datasets
- Scalability and Real-Time Processing

### **3.4.1 Effectiveness in handling high-dimensional data**

Autoencoders can efficiently learn compressed representations of normal transaction data, making them highly effective in detecting anomalies (e.g., fraud) that deviate from these patterns. Their ability to capture non-linear relationships in high-dimensional data is crucial for identifying subtle irregularities in complex financial transactions.

### **3.4.2 Handling Imbalanced datasets**

Financial transactions often have a highly imbalanced dataset (few anomalies compared to a large volume of normal transactions), which poses challenges for training both Autoencoders and SVM. Deep learning methods may require careful tuning (e.g., weighted loss functions) to avoid bias toward the majority class and ensure effective detection of rare anomalies.

### **3.4.3 Scalability and Real time processing**

Deep learning models, including Autoencoders and SVM (Support Vector Machines), can be scaled to handle large datasets typical in financial systems. Autoencoders, being unsupervised, can be trained on vast amounts of unlabeled transaction data, while SVM, though computationally more expensive in higher dimensions, can be used effectively for anomaly classification if anomalies are labeled.

## **3.5 System Specification**

### **3.5.1 Hardware Specification**

- Processor - Intel i5-8250 CPU @1.60GHz 1.80GHz
- 512 GB SSD
- NVIDIA GEFORCE RTX
- CPU QUAD CORES

### 3.5.2 Software Specification

- ANACONDA
- PYTHON
- VISUAL STUDIO
- **TensorFlow:** For deep learning model implementation
- **Pytorch:** Build deep learning models for tasks
- **Scikit-Learn:** Implement ML Models and Anomaly Detection.



## CHAPTER 4

### PROPOSED WORK

#### 4.1 Introduction

A hybrid deep learning model that used autoencoders, support vector machines, and stochastic gradient descent to handle nonlinear patterns and adaptive categorization in transaction data served as the foundation for the system designed to detect anomalies in financial transactions. In order to minimize the reconstruction error during the training phase, it first employed an autoencoder to learn the latent representation of typical transaction behavior. Transactions with significant reconstruction mistakes are flagged as anomalies by the autoencoder after it has been trained on normal data. After that, these learnt representations are used to classify data using an SVM. This makes it even easier for the model to discern between typical and unusual transactions. However, in order to respond to evolving data patterns over time, it iteratively optimizes the model parameters via SGD.

#### 4.2 Dataset

The dataset, which includes identities and quantities for financial transactions, has 10 columns and 533,009 items. Transaction ID `BELNR`, currency code `WAERS`, company code `BUKRS`, account type `KTOSL`, profit center `PRCTR`, posting key `BSCHL`, general ledger account `HKONT`, amount in local currency `DMBTR`, and amount in foreign currency `WRBTR` are among the important columns. In order to facilitate the training and validation of anomaly detection models, the dataset also includes a `label` column that indicates whether a transaction is "regular" or perhaps anomalous.

A particularly notable column, label, categorizes each transaction with terms like "regular," which could indicate whether a transaction follows expected patterns. This column implies that the dataset could be leveraged for anomaly detection tasks, where machine learning models can be trained to distinguish between regular and potentially anomalous transactions. The combination of categorical, numeric, and labeled data in this dataset makes it well-suited for supervised learning algorithms aimed at identifying irregularities in financial data.

### 4.3 Architecture Diagram

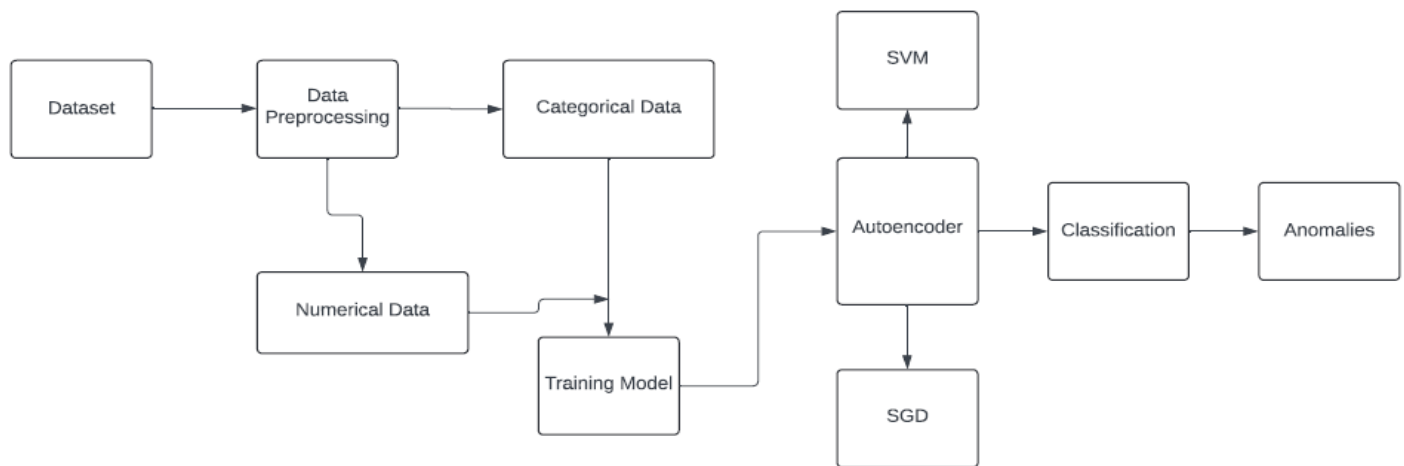


Figure 4.1: **Architecture Diagram**

Figure 1 The Architecture for anomaly detection in financial transactions using deep learning involves collecting and preprocessing transaction data, then performing feature engineering and feature selection to highlight important attributes. This refined data is fed into a deep learning model for training and subsequent validation to ensure accuracy. The trained model is then used to detect anomalies in new transactions, classifying suspicious activities. Alerts are generated for these anomalies, and detailed reports are created for further analysis and action.

## 4.4 Module Description

### 4.4.1 Module for Data Preprocessing

The transaction data is prepared for the anomaly detection model by this module. To guarantee conformity with deep learning and machine learning methodologies, it will do data cleaning, normalization, and encoding of categorical variables. In order to facilitate feature extraction and analysis, it transforms the transactions into a structured format.

#### **4.4.2 Module for Feature Extraction**

The key characteristics of financial transactions are extracted by this module, which then converts them into representations that show patterns that point to both typical and unusual behavior. Featurescaling and other dimensionality reduction methods may also be usedto get the data ready for the models to process.

#### **4.4.3 Module Autoencoder**

By encoding and decoding the transaction data, the autoencoder model discovers the fundamental structure of typical transaction activity. It learns latent representations of normal patterns by minimizing reconstruction errors on regular data during training. Transactions with large reconstruction errors are flagged as possible anomalies during inference because they diverge from learned usual behaviour.

#### **4.4.4 SVM Classification Module**

Following anomaly scoring using an autoencoder, transactions werecategorized as normal or anomalous using an SVM classifier based onthe retrieved characteristics and the aforementioned score. With only little variations between typical and anomalous transaction patterns, the SVM model offers better classification.

#### **4.4.5 Optimization Module (SGD)**

This module optimizes parameters using the Stochastic Gradient Descent algorithm. Large datasets are made possible by it, and it neverstops learning as fresh transaction data is added. It enhances detectionaccuracy over time and enables real-time fine-tuning in dynamic settings.

## **CHAPTER 5**

### **IMPLEMENTATION AND TESTING**

#### **5.1 Understanding the components**

This type of anomaly detection based on financial transactions with methods such as deep learning and machine learning detects unusual patterns and identifies fraudulent or error-filled data. This system exploits the ability of an autoencoder to learn normal patterns from the data within a financial transaction set. An autoencoder is a kind of neural network model designed to specialize in reconstructing the data; an autoencoder compresses it into a lower-dimensional latent space, then reconstructs the compressed data in this process and aims to minimize the reconstruction error. The reconstructed transactions having high reconstruction errors - greatly different from what it learned as normal transactions - indicate an anomaly in the application to new data. It is well suited for a lot of unsupervised elements because fraud data are scarce to get labeled cases while it learns normal transaction behavior purely from the accuracy in reconstruction, thus flagging unusual patterns without predefined fraud labels.

This tries to make the model robust by combining elements from SVM and SGD. This uses the learned latent representations for classifying normal and anomalous transactions using SVM. Then, the model enhances its parameters through SGD where the evolving patterns of data will iteratively find an optimum position. As it is a hybrid approach that utilizes deep learning strength for complex data representation and machine learning for classification, this provides scalability and adaptability to anomaly detection in the high-volume financial transaction environments.

##### **5.1.1 Input Phase**

Load and preprocessing of dataset. The primary reason to do so is the fact that financial data tends to be very unstructured; hence the stage of processing missing values, normalizing the numeric attributes, and encoding of categorical features. It ensures that common scales are supported by feature standardization while allowing features to train efficaciously in models, such as Autoencoder, SVM, and SGD. The training and testing subsets should be obtained by splitting the dataset. The training data would ideally contain "typical" patterns of

transaction behavior, allowing the model to learn these usual patterns better. With the data preprocessed, the next step is feature selection or extraction.

Because high-dimensional data would impede model performance and lead to overfitting, relevant features need to be selected or dimensionality reduction techniques, like PCA, must be used. For Autoencoder models, it means input dimensions affect network architectures, architected to capture normal-class patterns and minimize reconstruction errors for anomalous cases. For SVM and SGD models, optimal feature selection maximizing class separation could really boost detection accuracy. Scaling features is especially relevant here because SVM is sensitive to feature magnitudes, while SGD operates better with normalized data that results in faster convergence.

Finally, training and testing are done, with different approaches in each case. Auto-encoders are trained with an unsupervised approach; they learn the patterns involved in transactions and can identify an anomaly based on high reconstruction errors. SVMs are used one-class classification mainly for anomalous detection. They try to draw a decision boundary, which encloses the normal class, and mark outliers from it. The SGD models are used more with linear classification or regression; however, they still require iterative training and tuning for optimal separation. Some of the metrics for evaluation of performance of models include precision, recall, and F1-score with a tweak to address the issue of imbalance in anomaly detection to correctly identify fraudulent transactions.

	BELNR	WAERS	BUKRS	KTOSL	PRCTR	BSCHL	HKONT	DMBTR	WRBTR	label
0	288203	C3	C31	C9	C92	A3	B1	280979.60	0.00	regular
1	324441	C1	C18	C7	C76	A1	B2	129856.53	243343.00	regular
2	133537	C1	C19	C2	C20	A1	B3	957463.97	3183838.41	regular
3	331521	C4	C48	C9	C95	A2	B1	2681709.51	28778.00	regular
4	375333	C5	C58	C1	C19	A3	B1	910514.49	346.00	regular
5	327203	C1	C15	C6	C68	A1	B2	357627.56	704520.00	regular
6	292545	C4	C47	C2	C28	A2	B3	955576.84	128328.00	regular
7	335839	C1	C19	C1	C17	A1	B1	41769.26	0.00	regular
8	369064	C4	C40	C9	C97	A2	B1	44309.79	0.00	regular
9	138724	C6	C69	C1	C12	A2	B1	466720.45	43843.00	regular

**Figure 5.1 : Input Data**

### 5.1.2 Training Phase

The first process in the creation of an anomaly detection system for financial transactions, using Autoencoders, SVM, and SGD is data preprocessing. This step includes handling missing values, scaling features, and encoding categorical variables if needed. Most financial transaction data consists of different attributes from a transaction like amount, time, and merchant code that needs normalization since neural networks are sensitive to scales of features. It requires a training-validation split as it saves a portion of the data for validating the performance of the model to correctly respond to unseen examples. It is essentially a semi-supervised anomaly detection problem as the normal transaction data alone is utilized in the training of the model.

The Autoencoder model forms the core deep learning component for any anomaly detection model. After training, the Autoencoder learns to compress and then reconstruct normal transactions with minimum reconstruction error. The reconstruction error increases significantly when anomalous data, such as fraudulent transactions, is fed into it, which aids in the detection of outliers. The model may be trained with Mean Squared Error as the loss function, and thresholding is determined on validation data, which flags transactions with high reconstruction error as anomalies. This method captures complex transaction patterns but may be supplemented by other models.

After training the Autoencoder, other methods such as SVM and SGD have been used for further perfecting the detection. SVM can be a one-class classifier to differentiate normal instances from abnormal ones from the latent features obtained using the bottleneck layer of Autoencoder. In contrast, SGD classifier is used in online learning to update anomalies within the shortest possible time at the changing environments of transaction. A combination of these models increases the robustness further and lets detection systems generalize well to the variations of anomalies, lowering false positives.

### 5.1.3 Evaluating Phase

The evaluating step in the application of anomaly detection on financial transactions by implementing Autoencoders, SVM, and SGD will be preparation of proper data. To this



effect, preparation shall entail acquiring the data as per a comprehensive strategy but to guarantee representation of both norm and an atypical sequence. Most significant preprocessing steps incorporate the missing value handling approach, numerical features normalization technique, and categorical variables' encoding. The dataset should also be balanced. Class imbalance is prevalent for financial data, wherein good transactions overshadow anomalies. Improving model performance for discovering rare fraudulent transactions requires various techniques: oversampling of the minority class and undersampling of the majority class.

Once the preprocessed data is available, it is time to train selected models. Autoencoders are neural networks that learn a compact representation of the input data to reconstruct normal transactions, thus reporting anomalies whenever reconstruction errors exceed a certain threshold. In SVM, a hyperplane is established to classify transactions with the aim of maximizing the margin between classes. This will allow the applicability of SVM in distributions of data which are non-linearly separable and can be adapted with the help of the kernel trick, and the tuning of parameters also makes up the critical step of performance tuning. The use of SGD is made when a number of models are optimized since one can tune the weights in an iterative manner for further improved classifications. Cross-validation and grid search will turn out to be very fundamental in hyperparameter optimization for this stage so that these models generalize well to unobserved data.

For the final phase, a range of metrics specifically suited for anomaly detection is used to measure model performance - precision, recall, F1-score, and ROC-AUC area under the curve. These allow one to assess the aptness of models to be able to distinguish between normal and anomalous transactions. The confusion matrix also now becomes an important source of insight on true positives, false positives, true negatives, and false negatives. This would mean an algorithm that has good precision but poor recall may be a conservative model, while an algorithm with high recall might actually end up producing more false positives than necessary. Thus, the output will provide the capacity to fine-tune these models and pick the best performing approach for real-world deployment. Further, ensemble methods or hybrid models might make detection even better for a robust system in financial anomaly detection.

## 5.2 Testing

Use standard transaction data to train an autoencoder. The autoencoder should produce significant reconstruction mistakes on anomalies or unseen data since it learns to recreate regular data. It is more akin to a standard autoencoder, but it has probabilistic additions that can aid discover unusual events by capturing possible uncertainty in the data.

### 5.2.1 Unit Testing

**Data Preprocessing Verification:** Verify that feature scaling, data normalization, and missing value handling are done correctly. Verify that the training and test datasets have been appropriately divided and that any label encoding or modifications have been done uniformly to both sets.

**Model Accuracy and Reconstruction Error:** Verify that autoencoders have a model reconstruction error that is higher for anomalies and lower for typical transactions. It is important to confirm threshold values in order to correctly categorize abnormalities. Verify that SGD correctly trains the model by convergent to an optimal solution in a reasonable amount of iterations.

**Precision and Recall of Anomaly Detection:** Use unit tests on synthetic data with known anomalies to assess the model's accuracy and recall in detecting abnormalities. Verify that the number of false positives and false negatives is kept to a minimum, paying particular attention to the impact.

### 5.2.2 Integration Testing

Integration testing should, therefore confirm whether data preprocessing is homogenous in training, validation, and testing stages of this project since anomaly detection tends to rely significantly on quality of data. The unit tests and integration tests for autoencoders should be used to verify whether the model is capable of reconstructing normal data while detecting anomalies.

Tests for SGT need to check the ability of the model to distinguish between normal and anomalous sequences, as well as verify accuracy, recall, and precision under varied conditions. It should ensure that the model's outputs perfectly integrate with the

downstream alerting and logging systems. For instance, if the anomaly has been detected, the integration tests would guarantee the correct alerts are triggered and appropriately routed to the systems.

### **5.2.3 Functional Testing**

It is important to conduct functional testing for anomaly detection in financial transactions using deep learning methods like Autoencoders, Support Vector Machines (SVM), and Stochastic Gradient Descent (SGD) for the reliability and performance of the system. First, validate the functionality of the data preprocessing pipeline that includes feature extraction, normalization, and transformation. The test cases have to cover scenarios involving diversified inputs such as varied types of transactions, various sizes, and anomalies since the system must not produce errors or data loss for such diverse inputs.

Each model is to be tested. Testing an Autoencoder must involve functional testing where the error of reconstruction is correctly being portrayed as an anomaly score. Otherwise, the model cannot distinguish between normal transactions and anomalous ones. Tests for SVM should be to validate the boundary the model provides, check for precision and recall in anomaly detection. While testing SGD optimization technique entails checking whether the model converges appropriately during training while maintaining resilience as one changes hyperparameters.

**Tests Integration Testing** Integration testing is critical to test the workflow from the input of data all through anomaly detection and reporting. This will involve ascertaining whether what the models are spitting out corresponds to what is forecasted, in terms of the number of false positives or false negatives. Testing frameworks and frameworks for automation can easily simulate a number of different transaction scenarios continuously while validating the models that keep flowing into the applications as data streams. Performance metrics, for instance, processing speed and model accuracy, should also be monitored to ensure that the system meets operational requirements and can scale effectively under different loads.

## 5.3 Results

The anomaly detection process in financial transactions is more critical since fraudulent activities lead to serious losses. Traditional methods face significant challenges, however, in dealing with complex and high-dimensional nature of the transaction data. Deep learning methods, including Autoencoders, Support Vector Machines, and Stochastic Gradient Descent, have emerged as some of the effective techniques for enhancing anomaly detection. The interesting thing about the models mentioned above is the ability of these models to automatically discover sophisticated patterns in data. So the work is best fitted for operations related to slight moves away from what is normal.

Autoencoder is a deep neural network built to learn how to provide representations of data economically. Regarding a financial transaction's case, one trains an autoencoder to learn normal forms of patterns in transactions but uses that training to reconstruct new input information. In the reconstruction phase, anomalies typically cause higher reconstruction errors because they do not fit into learned patterns. This allows thresholding of such errors to effectively identify outliers. Research has proven that Autoencoders, when strengthened with dropout and denoising, significantly improve the robustness of anomaly detection as they can generalize well to unseen data and learn to adapt to any changes in transaction behavior.

SVM is another anomaly detection technique if the data is linear separable. SVM determines the appropriate hyperplane, which maximally differentiates between normal transactions and anomaly, making use of kernel function in case of non-linear data. And the optimization is done by using SGD method in case of large dataset because it converges very fast. The incrementally updated model weights make it possible for SGD to deal with big data. Combining SVM and SGD can help to improve the detection rates, decrease false positives, and hence make these methods quite effective for real-time anomaly detection in financial transactions. Altogether, these techniques represent the strength of deep learning for the enhancement of security within the financial sector.

## **CHAPTER 6**

### **RESULTS AND DISCUSSIONS**

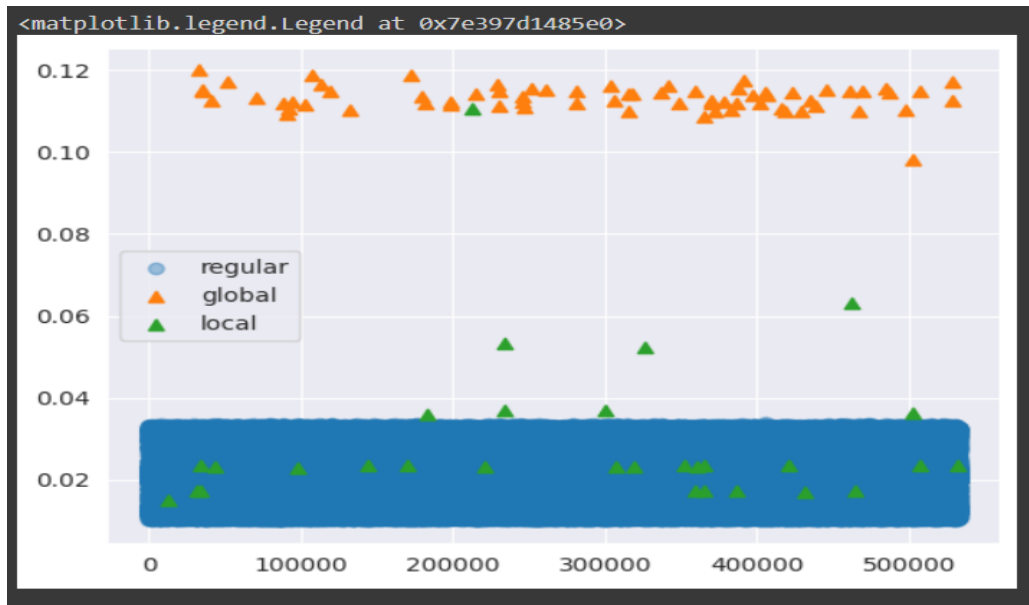
#### **6.1 Efficiency of the Proposed System**

The suggested approach for detecting anomalies in financial transactions employing deep learning techniques—specifically, support vector machines (SVM), autoencoders, and stochastic gradient descent (SGD) shows excellent effectiveness in spotting odd or fraudulent patterns. The main technique for feature extraction is an autoencoder, which reduces dimensionality while keeping important transaction attributes and captures non-linear relationships. SGD is used to streamline the training procedure, guaranteeing quicker convergence and effective management of the enormous datasets seen in financial data. Last but not least, SVM, which is renowned for its excellent classification performance, uses the encoded features to precisely distinguish between typical and anomalous transactions. When combined, these techniques offer a strong framework that improves detection precision and reduces false positives, both of which are critical for real-time financial monitoring. Combining the ability of deep learning to represent complicated data with traditional.

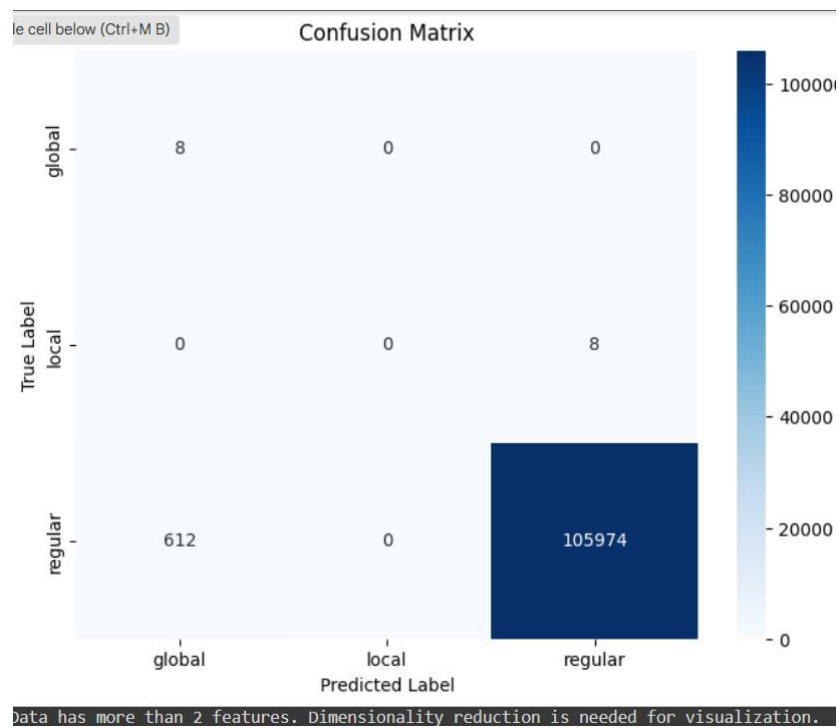
#### **6.2 Comparison of Existing and Proposed System**

Existing systems for detecting financial anomalies frequently use machine learning models such as Support Vector Machines (SVM) and Stochastic Gradient Descent (SGD) classifiers, or more conventional statistical and rule-based techniques. These issues are addressed by suggested deep learning algorithms, especially autoencoders, which are very good at identifying outliers in high-dimensional financial data by learning compressed representations of typical transaction patterns. Autoencoders are more able to adjust to changes in data patterns than SVM and SGD. They can also be improved by using designs such as recurrent neural networks (RNNs) or variational autoencoders (VAEs) to capture sequential dependencies, which further improves anomaly identification.

## Output



**Figure 6.1 Individual Transactions view**



**Figure 6.2 : Anomaly in Confusion matrix view**



## CHAPTER 7

### CONCLUSION AND FUTURE ENHANCEMENTS

#### 7.1 Conclusion

In conclusion, by utilizing their capacity to identify intricate, high-dimensional patterns in data, deep learning techniques like Autoencoders, Stochastic Gradient Descent (SGD), and Support Vector Machines (SVM) offer practical strategies for identifying irregularities in financial transactions. Particularly noteworthy are autoencoders, which record condensed representations of common transaction behaviors, facilitating the identification of variations suggestive of fraud. In linearly separable data, SGD and SVM are useful due to their interpretability and efficiency; however, they may not be able to handle complex, non-linear transaction patterns, where Autoencoders are superior. Overall, by lowering false positives, responding to changing fraud strategies, and providing scalable solutions for huge transaction datasets, deep learning-based anomaly detection improves fraud detection and is therefore ideal for contemporary financial systems.

#### 7.2 Future Enhancements

**Integration of Hybrid Models:** By combining autoencoders with more conventional models, such as SVM and SGD, a hybrid strategy may be able to capitalize on each of their advantages. To increase accuracy and lower false positives, autoencoders, for example, may process high-dimensional data and extract pertinent features that could be put into an SVM or SGD classifier.

**Graph Neural Networks (GNNs):** GNNs can be used to identify unusual patterns in networked data and represent intricate interactions between entities (such as users, accounts, and transactions). This is especially helpful in identifying coordinated fraudulent activity, as unusual account links or sequences may indicate fraud.

**Real-time and Adaptive Learning:** The system may be able to adjust in real-time to changing fraud trends by utilizing online or incremental learning strategies with autoencoders, SVM, and SGD models.

## CHAPTER 8

### SOURCE CODE

#### 1. Import Libraries

```
# importing utilities

import os

import sys

from datetime import datetime

# importing data science libraries

import pandas as pd

import random as rd

import numpy as np

# importing pytorch libraries

import torch

from torch import nn

from torch import autograd

from torch.utils.data import DataLoader

# import visualization libraries

import matplotlib.pyplot as plt

from mpl_toolkits.mplot3d import Axes3D

import seaborn as sns

from IPython.display import Image, display

sns.set_style('darkgrid')

# ignore potential warnings

import warnings

warnings.filterwarnings("ignore")
```

```

# Allow for Jupyter notebook inline plotting:

%matplotlib inline

# print CUDNN backend version

now = datetime.utcnow().strftime("%Y%m%d-%H:%M:%S")

print('[LOG {}] The CUDNN backend version: {}'.format(now,
torch.backends.cudnn.version()))

# If CUDNN and GPU's are available let's still specify if we want to use both:

USE_CUDA = True

# print current Python version

now = datetime.utcnow().strftime("%Y%m%d-%H:%M:%S")

print('[LOG {}] The Python version: {}'.format(now, sys.version))

# print current PyTorch version

now = datetime.utcnow().strftime("%Y%m%d-%H:%M:%S")

print('[LOG {}] The PyTorch version: {}'.format(now, torch.__version__))

# init deterministic seed

seed_value = 1234 #4444 #3333 #2222 #1111 #1234

rd.seed(seed_value) # set random seed

np.random.seed(seed_value) # set numpy seed

torch.manual_seed(seed_value) # set pytorch seed CPU

if (torch.backends.cudnn.version() != None and USE_CUDA == True):

    torch.cuda.manual_seed(seed_value) # set pytorch seed GPU

ori_dataset = pd.read_csv('Anomaly_dataset.csv')

ori_dataset.head()

# inspect the datasets dimensionalities

now = datetime.utcnow().strftime("%Y%m%d-%H:%M:%S")

print('[LOG {}] Transactional dataset of {} rows and {} columns loaded'.format(now,
ori_dataset.shape[0], ori_dataset.shape[1]))

```

```

# inspect top rows of dataset

ori_dataset.head(10)

# number of anomalies vs. regular transactions

ori_dataset.label.value_counts()

# remove the "ground-truth" label information for the following steps of the lab

label = ori_dataset.pop('label')

# inspect top rows of dataset

ori_dataset.head(10)

```

## 2. Pre processing of Categorical transactions attributes

```

fig, ax = plt.subplots(1,2)

fig.set_figwidth(20)

# plot the distribution of the posting key attribute

g = sns.countplot(x=ori_dataset['BSCHL'], ax=ax[0])

g.set_xticklabels(g.get_xticklabels(), rotation=90)

g.set_title('Distribution of BSCHL attribute values')

# plot the distribution of the general ledger account attribute

g = sns.countplot(x=ori_dataset['HKONT'], ax=ax[1])

g.set_xticklabels(g.get_xticklabels(), rotation=90)

g.set_title('Distribution of HKONT attribute values')

# select categorical attributes to be "one-hot" encoded

categorical_attr_names = ['KTOSL', 'PRCTR', 'BSCHL', 'HKONT', 'WAERS', 'BUKRS']

# encode categorical attributes into a binary one-hot encoded representation

ori_dataset_categ_transformed = pd.get_dummies(ori_dataset[categorical_attr_names])

# inspect encoded sample transactions

ori_dataset_categ_transformed.head(10)

```

### 3. Pre processing of Numerical transactions attributes

```
# plot the log-scaled "DMBTR" as well as the "WRBTR" attribute value distribution

fig, ax = plt.subplots(1,2)

fig.set_figwidth(20)

# plot distribution of the local amount attribute

g = sns.distplot(ori_dataset['DMBTR'].tolist(), ax=ax[0])

g.set_title('Distribution of DMBTR amount values')

# plot distribution of the document amount attribute

g = sns.distplot(ori_dataset['WRBTR'].tolist(), ax=ax[1])

g.set_title('Distribution of WRBTR amount values')


# select "DMBTR" vs. "WRBTR" attribute

numeric_attr_names = ['DMBTR', 'WRBTR']

# add a small epsilon to eliminate zero values from data for log scaling

numeric_attr = ori_dataset[numeric_attr_names] + 1e-7

numeric_attr = numeric_attr.apply(np.log)

# normalize all numeric attributes to the range [0,1]

ori_dataset_numeric_attr = (numeric_attr - numeric_attr.min()) / (numeric_attr.max() -
numeric_attr.min())

# append 'label' attribute for colour distinction

numeric_attr_vis = ori_dataset_numeric_attr.copy()

numeric_attr_vis['label'] = label

# plot the log-scaled and min-max normalized numeric attributes

g = sns.pairplot(data=numeric_attr_vis, vars=numeric_attr_names, hue='label')

g.fig.suptitle('Distribution of DMBTR vs. WRBTR amount values')

g.fig.set_size_inches(15, 5)
```

## 4. Merge categorical and numeric subsets

```
ori_subset_transformed = pd.concat([ori_dataset_categ_transformed, ori_dataset_numeric_attr],
axis = 1)

# inspect final dimensions of pre-processed transactional data

# ori_subset_transformed = ori_subset_transformed.sample(frac=0.50)

ori_subset_transformed.shape

import gc

gc.collect()
```

## 5. Autoencoders Neural Network

```
import torch

import torch.nn as nn

# implementation of the shallow encoder network

# containing only a single layer

class encoder(nn.Module):

    def __init__(self):

        super(encoder, self).__init__()

        # specify layer 1 - in 618, out 3

        self.encoder_L1 = nn.Linear(in_features=ori_subset_transformed.shape[1],
out_features=3, bias=True) # add linearity

        nn.init.xavier_uniform_(self.encoder_L1.weight) # init weights according to [9]

        self.encoder_R1 = nn.LeakyReLU(negative_slope=0.4, inplace=True) # add non-linearity
according to [10]

    def forward(self, x):

        # define forward pass through the network

        x = self.encoder_R1(self.encoder_L1(x)) # don't apply dropout to the AE bottleneck

        return x

# init training network classes / architectures
```

```

encoder_train = encoder()

# push to cuda if cudnn is available

if (torch.backends.cudnn.version() != None and USE_CUDA == True):

    encoder_train = encoder().cuda()

# print the initialized architectures

now = datetime.utcnow().strftime("%Y%m%d-%H:%M:%S")

print('[LOG { }] encoder architecture:\n\n{ }\n'.format(now, encoder_train))

# implementation of the shallow decoder network

# containing only a single layer

class decoder(nn.Module):

    def __init__(self):

        super(decoder, self).__init__()

        # specify layer 1 - in 3, out 618

        self.decoder_L1 = nn.Linear(in_features=3,
out_features=ori_subset_transformed.shape[1], bias=True) # add linearity

        nn.init.xavier_uniform_(self.decoder_L1.weight) # init weights according to [9]

        self.decoder_R1 = nn.LeakyReLU(negative_slope=0.4, inplace=True) # add non-linearity
according to [10]

    def forward(self, x):

        # define forward pass through the network

        x = self.decoder_R1(self.decoder_L1(x)) # don't apply dropout to the AE output

        return x

# init training network classes / architectures

decoder_train = decoder()

# push to cuda if cudnn is available

if (torch.backends.cudnn.version() != None) and (USE_CUDA == True):

    decoder_train = decoder().cuda()

```

```
# print the initialized architectures

now = datetime.utcnow().strftime("%Y%m%d-%H:%M:%S")

print('[LOG { }] decoder architecture:\n\n{ }\n'.format(now, decoder_train))
```

## 6. Autoencoder Training

```
# define the optimization criterion / loss function

loss_function = nn.BCEWithLogitsLoss(reduction='mean')

# define learning rate and optimization strategy

learning_rate = 1e-3

encoder_optimizer = torch.optim.Adam(encoder_train.parameters(), lr=learning_rate)

decoder_optimizer = torch.optim.Adam(decoder_train.parameters(), lr=learning_rate)
```

## 7. Training Parameters

```
num_epochs = 5

mini_batch_size = 128

import torch

import numpy as np

from torch.utils.data import DataLoader

import pandas as pd # Import pandas

# Assuming ori_subset_transformed is your pandas DataFrame

# Convert all columns to numeric dtype, handling errors as needed

# You might need to adjust the 'errors' argument based on how you want to handle invalid
values

numeric_df = ori_subset_transformed.apply(pd.to_numeric, errors='coerce')

# Explicitly convert to a supported dtype before creating the tensor

# Choose float32 or float64 based on your needs

torch_dataset = torch.from_numpy(numeric_df.values.astype(np.float32))

# Rest of your code remains the same
```



```

dataloader = DataLoader(torch_dataset, batch_size=mini_batch_size, shuffle=True,
num_workers=0)

if (torch.backends.cudnn.version() != None) and (USE_CUDA == True):

    dataloader = DataLoader(torch_dataset.cuda(), batch_size=mini_batch_size, shuffle=True)

# init collection of mini-batch losses

losses = []

# convert encoded transactional data to torch Variable

data = autograd.Variable(torch_dataset)

# train autoencoder model

for epoch in range(num_epochs):

    # init mini batch counter

    mini_batch_count = 0

    # determine if CUDA is available at compute node

    if(torch.backends.cudnn.version() != None) and (USE_CUDA == True):

        # set networks / models in GPU mode

        encoder_train.cuda()

        decoder_train.cuda()

    # set networks in training mode (apply dropout when needed)

    encoder_train.train()

    decoder_train.train()

    # start timer

    start_time = datetime.now()

    # iterate over all mini-batches

    for mini_batch_data in dataloader:

        # increase mini batch counter

        mini_batch_count += 1

```

```

# convert mini batch to torch variable

mini_batch_torch = autograd.Variable(mini_batch_data)

    # run forward pass

z_representation = encoder_train(mini_batch_torch) # encode mini-batch data

mini_batch_reconstruction = decoder_train(z_representation) # decode mini-batch data

    # determine reconstruction loss

reconstruction_loss = loss_function(mini_batch_reconstruction, mini_batch_torch)

    # reset graph gradients

decoder_optimizer.zero_grad()

encoder_optimizer.zero_grad()

# run backward pass

reconstruction_loss.backward()

    # update network parameters

decoder_optimizer.step()

encoder_optimizer.step()

# print training progress each 1'000 mini-batches

if mini_batch_count % 1000 == 0:

    # print the training mode: either on GPU or CPU

    mode = 'GPU' if (torch.backends.cudnn.version() != None) and (USE_CUDA ==
True) else 'CPU'

    # print mini batch reconstruction results

    now = datetime.utcnow().strftime("%Y%m%d-%H:%M:%S")

    end_time = datetime.now() - start_time

    print('[LOG {}] training status, epoch: [{:04}/{:04}], batch: {:04}, loss: {},
mode: {}, time required: {}'.format(now, (epoch+1), num_epochs, mini_batch_count,
np.round(reconstruction_loss.item(), 4), mode, end_time))

    # reset timer

```

```

start_time = datetime.now()

# set networks in evaluation mode (don't apply dropout)
encoder_train.cpu().eval()
decoder_train.cpu().eval()

# reconstruct encoded transactional data
reconstruction = decoder_train(encoder_train(data))

# determine reconstruction loss - all transactions
reconstruction_loss_all = loss_function(reconstruction, data)

# collect reconstruction loss
losses.extend([reconstruction_loss_all.item()])

# print reconstruction loss results
now = datetime.utcnow().strftime("%Y%m%d-%H:%M:%S")

print('[LOG {}] training status, epoch: [{:04}/{:04}], loss: {:.10f}'.format(now,
(epoch+1), num_epochs, reconstruction_loss_all.item()))

# save trained encoder model file to disk
encoder_model_name = "ep_{ }_encoder_model.pth".format((epoch+1))

torch.save(encoder_train.state_dict(), os.path.join("/content/sample_data/Anomaly
folder", encoder_model_name))

# # save trained decoder model file to disk
decoder_model_name = "ep_{ }_decoder_model.pth".format((epoch+1))

torch.save(decoder_train.state_dict(), os.path.join("/content/sample_data/Anomaly
folder", decoder_model_name))

gc.collect()

from google.colab import drive
drive.mount('/content/drive')

# plot the training progress

```

```

plt.plot(range(0, len(losses)), losses)

plt.xlabel('[training epoch]')

plt.xlim([0, len(losses)])

plt.ylabel('[reconstruction-error]')

#plt.ylim([0.0, 1.0])

plt.title('AENN training performance')

# Evalauting the Autoencoder

import os

import torch

# restore pretrained model checkpoint

# Correct the file paths to load the appropriate models

encoder_model_name = "/content/sample_data/Anomaly
folder/ep_5_encoder_model.pth" # Changed to encoder model path

decoder_model_name = "/content/sample_data/Anomaly
folder/ep_5_decoder_model.pth"

# init training network classes / architectures

encoder_eval = encoder()

decoder_eval = decoder()

```

## 8. Load trained models

```

encoder_eval.load_state_dict(torch.load(encoder_model_name)) # Removed
os.path.join, assuming models are in the same directory

decoder_eval.load_state_dict(torch.load(decoder_model_name)) # Removed
os.path.join, assuming models are in the same directory

```

## 9. Pre-Trained model

```

# convert encoded transactional data to torch Variable

data = autograd.Variable(torch_dataset)

# set networks in evaluation mode (don't apply dropout)

```

```

encoder_eval.eval()

decoder_eval.eval()

# reconstruct encoded transactional data

reconstruction = decoder_eval(encoder_eval(data))

# determine reconstruction loss - all transactions

reconstruction_loss_all = loss_function(reconstruction, data

# print reconstruction loss - all transactions

now = datetime.utcnow().strftime("%Y%m%d-%H:%M:%S")

print('[LOG {}] collected reconstruction loss of: {:06}/{:06} transactions'.format(now,
reconstruction.size()[0], reconstruction.size()[0]))

print('[LOG {}] reconstruction loss: {:.10f}'.format(now, reconstruction_loss_all.item()))

```

## 10. Assessment of individual transactions

```

# init binary cross entropy errors

reconstruction_loss_transaction = np.zeros(reconstruction.size()[0])

# iterate over all detailed reconstructions

for i in range(0, reconstruction.size()[0]):

    # determine reconstruction loss - individual transactions

    reconstruction_loss_transaction[i] = loss_function(reconstruction[i], data[i]).item()

    if(i % 100000 == 0):

        ### print conversion summary

        now = datetime.utcnow().strftime("%Y%m%d-%H:%M:%S")

        print('[LOG {}] collected individual reconstruction loss of: {:06}/{:06}
transactions'.format(now, i, reconstruction.size()[0]))

# prepare plot

fig = plt.figure()

```

```

ax = fig.add_subplot(111)

# assign unique id to transactions

plot_data = np.column_stack((np.arange(len(reconstruction_loss_transaction)),
reconstruction_loss_transaction))

# obtain regular transactions as well as global and local anomalies

regular_data = plot_data[label == 'regular']

global_outliers = plot_data[label == 'global']

local_outliers = plot_data[label == 'local']

# plot reconstruction error scatter plot

ax.scatter(regular_data[:, 0], regular_data[:, 1], c='C0', alpha=0.4, marker="o",
label='regular') # plot regular transactions

ax.scatter(global_outliers[:, 0], global_outliers[:, 1], c='C1', marker="^",
label='global') # plot global outliers

ax.scatter(local_outliers[:, 0], local_outliers[:, 1], c='C2', marker="^",
label='local') # plot local outliers

# add plot legend of transaction classes

ax.legend(loc='best')

```

## 11. Evaluating SGD Algorithm

```

# append labels to original dataset

ori_dataset['label'] = label

# inspect transactions exhibiting a reconstruction error >= 0.1

ori_dataset[reconstruction_loss_transaction >= 0.1].head()

# inspect transactions exhibiting a reconstruction error < 0.1 and >= 0.05

ori_dataset[(reconstruction_loss_transaction >= 0.05) &
(reconstruction_loss_transaction <= 0.09)]

import pandas as pd

from sklearn.model_selection import train_test_split

```

```

from sklearn.linear_model import SGDClassifier # Using SGDClassifier

from sklearn.metrics import accuracy_score, classification_report,
confusion_matrix

import matplotlib.pyplot as plt

import seaborn as sns

from sklearn.preprocessing import OneHotEncoder

import numpy as np

# Load the dataset

ori_subset_transformed = pd.read_csv('Anomaly_dataset.csv')

label = ori_subset_transformed['label']

ori_subset_transformed = ori_subset_transformed.drop('label', axis=1)

# Create a OneHotEncoder object

encoder = OneHotEncoder(sparse_output=False, handle_unknown='ignore')

# Fit the encoder on the categorical features and transform them

categorical_features =
ori_subset_transformed.select_dtypes(include=['object']).columns

encoded_features =
encoder.fit_transform(ori_subset_transformed[categorical_features])

# Create a new DataFrame with encoded features

encoded_df = pd.DataFrame(encoded_features,
columns=encoder.get_feature_names_out(categorical_features))

```

## 12. Concatenate the encoded features with the numerical features

```

numerical_features =
ori_subset_transformed.select_dtypes(exclude=['object']).columns

X = pd.concat([ori_subset_transformed[numerical_features], encoded_df],
axis=1)

# Split data into training and testing sets

X_train, X_test, y_train, y_test = train_test_split(X, label, test_size=0.2,

```

```

random_state=42)

# Create the SGDClassifier (equivalent to linear SVM)

svm_classifier = SGDClassifier(loss='hinge', alpha=0.0001, max_iter=1000,
random_state=42)

# Partial fitting (in batches)

chunk_size = 10000 # Adjust chunk size as needed

for i in range(0, X_train.shape[0], chunk_size):

    svm_classifier.partial_fit(X_train[i:i + chunk_size], y_train[i:i + chunk_size],
classes=np.unique(y_train))

# Make predictions on the test set

y_pred = svm_classifier.predict(X_test)


# Evaluate the classifier

accuracy = accuracy_score(y_test, y_pred)

print(f"Accuracy: {accuracy}")

print(classification_report(y_test, y_pred))


# Visualize the results using a confusion matrix

cm = confusion_matrix(y_test, y_pred)

plt.figure(figsize=(8, 6))

sns.heatmap(cm, annot=True, fmt="d", cmap="Blues",

            xticklabels=['global', 'local', 'regular'],

            yticklabels=['global', 'local', 'regular'])

plt.xlabel("Predicted Label")

plt.ylabel("True Label")

plt.title("Confusion Matrix")

```



```

plt.show()

# Visualize decision boundaries (if data has 2 features)
if X.shape[1] == 2:
    x_min, x_max = X.iloc[:, 0].min() - 1, X.iloc[:, 0].max() + 1
    y_min, y_max = X.iloc[:, 1].min() - 1, X.iloc[:, 1].max() + 1
    xx, yy = np.meshgrid(np.arange(x_min, x_max, 0.1),
                          np.arange(y_min, y_max, 0.1))

    Z = svm_classifier.predict(np.c_[xx.ravel(), yy.ravel()])
    Z = Z.reshape(xx.shape)

    plt.contourf(xx, yy, Z, alpha=0.8, cmap=plt.cm.RdYlBu)
    plt.scatter(X.iloc[:, 0], X.iloc[:, 1], c=label, cmap=plt.cm.brg)
    plt.xlabel("Feature 1")
    plt.ylabel("Feature 2")
    plt.title("SVM Decision Boundaries")
    plt.show()
else:
    print("Data has more than 2 features. Dimensionality reduction is needed for
    visualization.")

```

## REFERENCES

- [1] ACFE, "Report to the Nations on Occupational Fraud and Abuse", The 2016 Global Fraud Study, Association of Certified Fraud Examiners (ACFE), 2016.
- [2] J. T. Wells, "Corporate Fraud Handbook: Prevention and Detection", John Wiley & Sons, 2017.
- [3] PwC, "Pulling Fraud Out of the Shadows", The Global Economic Crime and Fraud Survey 2018, PricewaterhouseCoopers LLP, 2018.
- [4] N. Srivastava, G. E. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting", The Journal of Machine Learning Research, 15(1), 1929-1958, 2014.
- [5] B. Xu, N. Wang, T. Chen, and M. Li, "Empirical Evaluation of Rectified Activations in Convolution Network", ICML Deep Learning Workshop, pages 1–5, 2015.
- [6] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization", International Conference on Learning Representations (ICLR). 2015.
- [7] Research on Time Series Anomaly Detection Algorithm and Application 2019 IEEE 4th Advanced Information Technology, Electronic and Automation Control Conference (IAEAC).
- [8] 2022 19th International SoC Design Conference (ISOCC) Performance Comparison of Soiling Detection Using Anomaly Detection Methodology.
- [9] DeepWindow: An Efficient Method for Online Network Traffic Anomaly Detection 2019 IEEE 21st International Conference on High Performance Computing and Communications; IEEE 17th International Conference on Smart City; IEEE 5th International Conference on Data Science and Systems (HPCC/SmartCity/DSS)
- [10] Nupur Sawarkar, Pranay Yenagandula, Devang Shetye and Prof. Shruti Agrawal, "Expense Manager: An Expense Tracking Application using Image Processing", *International Research Journal of Engineering and Technology (IRJET)*, vol. 09, no. 04, pp. 2395-0056, Apr 2022.