MACHINE LEARNING-BASED ANOMALY DETECTION IN BANK TRANSFERS

IDOGHO JOSHUA OSHOZE

PSC2102094

DEPARTMENT OF COMPUTER SCIENCE,

FACULTY OF PHYSICAL SCIENCES

UNIVERSITY OF BENIN,

BENIN CITY,

EDO STATE, NIGERIA.

NOVEMBER, 2025.

MACHINE LEARNING-BASED ANOMALY DETECTION IN BANK TRANSFERS

IDOGHO JOSHUA OSHOZE

PSC2102094

A PROJECT REPORT SUBMITTED TO
THE DEPARTMENT OF COMPUTER SCIENCE,
FACULTY OF PHYSICAL SCIENCES,
UNIVERSITY OF BENIN, BENIN CITY.

IN PARTIAL FUFILLMENT OF THE REQUIREMENT FOR THE AWARD OF A BACHELOR OF SCIENCE (B.Sc.) DEGREE IN COMPUTER SCIENCE

NOVEMBER 2025.

DECLARATION

I, **IDOGHO JOSHUA OSHOZE**, with Matriculation number **PSC2102094**, do hereby declare that:

This project work is based on a study undertaken by me in the Department of Computer Science, University of Benin, Benin City, under the supervision **Dr. E.C. Igodan.**

This research work has not been previously submitted for the award of degree elsewhere.

All ideas and views are a product of my personal research; and where the views of

others have been expressed; they were duly acknowledged.

All liabilities arising from the study are entirely mine and not of the supervisor.

_____ _____

Idogho Joshua Oshoze **DATE**

CERTIFICATION

This is to certify that this project work was carried out by **IDOGHO JOSHUA OSHOZE** with Matriculation Number **PSC2102094** under my supervision. It is adequate and satisfactory, both in scope and content, for the award of Bachelor of Science (B.Sc.) Degree in Computer Science of the University of Benin

_____ _____

Dr. E.C. IGODAN **DATE**

Project Supervisor

APPROVAL

This project work is hereby approved in partial fulfillment of the requirements for the award of Bachelor of Science (B.Sc.) Degree in Computer Science from the University of Benin.

_____ _____

Dr. (Mrs.) R. Usiobaifo **DATE**

Head of Department

DEDICATION

I want to dedicate this project to God Almighty for seeing me through from the beginning till now, to my wonderful family especially Very Rev Fr. Callistus Ikhane, for their constant love, support and care, and finally to all my friends Gabriel, Paschal, Purity and the Zeqah community for contributing to my academic journey.

ACKNOWLEDGEMENT

My utmost acknowledgement goes to God Almighty for giving me the strength, wisdom and direction throughout my academic journey. I would like to express my gratitude to my project supervisor Dr. Igodan E.C. for his consistent guidance towards ensuring the successful completion of this project.

I would also like to specially thank the Head of the department of Computer Science Dr. (Mrs.) A.R. Usiobaifo, and other lecturers in the Department of Computer Science who I have been opportune to cross paths with, and have impacted me immensely these past few years Prof. (Mrs.) F.A. Egbokhare, Prof. A. A. Imianvan, Prof. G. O. Ekuobase, Prof. (Mrs.) A. O. Egwali, Prof. F. I. Amadin, Prof. F. A. U. Imouokhome, Prof. (Mrs.) S. Konyeha, Prof. (Mrs.) V. I. Osubor, Prof. F. O. Chete, Dr. E. Nwelih, Dr. (Mrs.) G. O. Aziken, Mr. E. E. Obasohan, Dr. F. O. Oliha, Dr. (Mrs.) R. O. Osaseri, Dr. M. Osagie, Mr D.N Idehen, Mr. K. O. Otokiti, Mr. I. E. Obayagbona, Dr. L. O. Usiosefe, Mr. J. O. Okhuoya, Mr. G. I. Evbuomwan and the non-teaching staff, for your various roles and support. I acknowledge your contributions with appreciation.

I am forever grateful to my parents, my dad Late Mr. Columbanus Idogho and my sweet Mom Julian Idogho and all my siblings, especially Iyesomhi Idogho my second Mom, for their support throughout my time in the University of Benin. Special thanks to Very Rev. Fr. Callistus Mary Ikhane my beloved Guardian and Father for his continuous love and support.

To all my wonderful friends who directly or indirectly contributed to this effect; Gabriel,

Paschal, Purity, Clarissa and my entire Zaqah community and a host of others my memory might fail me to mention, thank you!

ABSTRACT

The exponential rise in digital banking transactions has heightened the risk of fraudulent and anomalous bank transfers, necessitating intelligent and automated mechanisms for its detection. Traditional rule-based systems often fail to capture evolving fraud patterns, motivating the adoption of machine learning-based anomaly detection techniques. This study aims to develop and evaluate robust unsupervised learning models for detecting anomalies in bank transaction datasets. Specifically, the research applies three state-of-the-art algorithms; Isolation Forest, Local Outlier Factor (LOF), and One-Class Support Vector Machine (OC-SVM) to identify irregular transaction behaviors indicative of potential fraud.

The methodology involves comprehensive data preprocessing, scaling, encoding, and feature selection to improve model learning. Real-world bank transfer datasets from kaggle were utilized, for this training. Each model's performance was assessed using standard evaluation metrics, including; Silhouette Score, Anomaly Ratio, and Average Decision Score. Results show that OCSVM performed best (Silhouette = 0.635, strong decision scores), reliably flagging about 5% of records as anomalies. Analysis of flagged transactions revealed consistent patterns high transaction amounts or balance-change ratios, very short or very long intervals between transactions, and activity at unusual hours making the alerts interpretable and practically useful.

The study demonstrates that an interpretable, lightweight anomaly detection pipeline can assist financial institutions in prioritizing suspicious transfers for review, and it provides recommendations for deployment, validation with expert labels, and further work on explainability and real-time monitoring.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER ONE

INTRODUCTION

1.0 BACKGROUND OF STUDY

Digital bank transfers are a core function of modern payment systems and they are processed at high volume and speed by banks and fintech platforms. While these capabilities have increased convenience and financial inclusion, they have also exposed services to a variety of unexpected behaviours in transfer data. While some transfers fail outrightly, like timeouts, reversals, routing errors, some are delayed, and others exhibit unusual patterns that can indicate technical faults, misconfigurations, or illicit activity. These unexpected observations in transfer logs are generally referred to as anomalies and are a major concern for operational reliability and risk management (Bakumenko & Elragal., 2022; Azamuke et al., 2023)

In data-science, an anomaly is an observation that deviates from normal patterns in the data; in payments, anomalies may reflect single abnormal events or correlated patterns across many transactions. Prominent literature on anomaly detection in financial systems identifies several useful categories that help frame detection strategies, which includes but not limited to;

1. Point (global) Anomalies: Individual transfers whose values (for example, amount) are extreme compared with the bulk of records. Point anomalies are frequently targeted in transaction-level fraud and transfer-integrity studies (Lokanan, 2023; Stefánsson, 2023).
2. Contextual Anomalies: Transfers that are anomalous only in certain contexts for example, a medium-sized transfer executed at 3:00 a.m. for a user who normally transacts during daytime hours. Time or sequence-based contextual reasoning is central to analyses of payment streams and high-frequency payment monitoring (Takahashi et al., 2024).
3. Collective Anomalies: Groups of related transfers that as a collection, reveal abnormal behaviour (for example, a burst of many small transfers from one account in rapid succession). Graph and sequence-based approaches are commonly used to surface such collective patterns (Pourhabibi et al., 2020; Oye et al., 2024).

Banking and mobile-money platforms process extremely large numbers of transfer transactions every day, and even a small proportion of anomalous transfers can cause significant financial losses, regulatory sanctions, and reputational damage. Manual review and static rule-based systems do not scale well to high-volume, high-velocity transaction streams and are brittle against evolving fraud patterns. Consequently, automated detection using machine learning (ML) has become an important tool for transaction monitoring and financial audit augmentation (Bakumenko & Elragal, 2022).

In the past decade machine learning has been applied across auditing, fraud detection and transaction monitoring to replace or augment manual review and brittle rules. For example, machine learning tools are now used to scan general ledgers and accounting records to flag unusual journal entries for auditors (Bakumenko & Elragal, 2022). Machine learning models help prioritize high-risk cases for investigators, reduce manual workload, and often achieve better detection rates than static rules because they can learn from many weak signals simultaneously (Azamuke et al., 2025; Bakumenko & Elragal, 2022).

In mobile-money and banking contexts, simulation-based studies and field work show that classifiers such as Random Forest and XGBoost can successfully detect fraudulent transfers when trained on realistic transaction logs (Azamuke et al., 2025). In this way, machine learning is not just a tool for analysis but a practical solution to improving reliability and trust in bank transfer systems.

This study focuses on designing an anomaly detection model for bank transfers using machine learning techniques.

1.1 STATEMENT OF PROBLEM

Banks and mobile-money services move large amounts of money every day. Most transfers are normal, but some are not, they may be mistakes, system glitches, or deliberate fraud. These unusual transfers are a problem because even a small number of them can cause direct financial loss, harm customers, and damage trust in the service.

This study is motivated by the fact that manual review does not scale, human teams cannot realistically inspect thousands or millions of transactions in real time. Relying on people or fixed rules means suspicious activity can be missed or detected too late. Also rule-based systems are brittle (for example, “flag any transfer above ₦500,000”) are easy to evade and hard to maintain. False positives have real costs, when a legitimate transfer is wrongly flagged

as suspicious, it annoys customers and creates extra work for staff. The business cost of false alarms can be as important as the cost of missed fraud.

Relational and temporal patterns matter, fraud often shows up not in a single transaction but in the way accounts interact over time, small repeated transfers, sudden spikes of activity, chains or rings of transfers between accounts. Capturing these patterns requires features and models that look beyond one-off transactions.

1.2 AIM AND OBJECTIVES

The specific aim of this study is to design a machine learning model for detecting anomaly in bank transfers.

The objectives of the study are:

1. Data Curation and processing
2. Development of a machine learning model that preprocesses transfer data, computes the chosen features, runs the selected anomaly detection algorithms, and outputs flagged transactions.
3. Evaluate Model performance: Measure detector/classifier performance using appropriate evaluation metrics.

1.3 METHODOLOGY

This study uses bank transaction datasets from Kaggle an open source data science and machine learning platform. The methodology has four main stages:

(1) Data Preprocessing: The raw datasets were processed to preparing them for machine learning models. This process involved:

(2) Feature Selection: The most relevant feature set in relation to anomaly detection includes; transaction features (amount, time-of-day, transaction type), account activity summaries (daily/weekly transaction counts, average amounts), and simple network/flow features (sender/receiver degree, short-term flow counts).

(3) Model Development: This following unsupervised models are adopted in this study;

a. Isolation Forest**:** This algorithm is an unsupervised anomaly detection method which isolates observations by recursively partitioning the data using random splits. Anomalies are easier to isolate and therefore have shorter average path lengths in the isolation trees; samples with substantially shorter path lengths are considered outliers. It is useful for high-dimensional and large-scale data and is widely implemented using the scikit-learn Library. The formula is as follows:

$$S(x, n) = 2^{\frac{-E(h(x))}{c(n)}} \qquad \ldots 1.1$$

where S is anomaly score for point x and the average of h(x) is E(h(x)), which is an average of all tree depths of a data-point. In the denominator, c(n) is a mean tree depth of an unsuccessful search calculated as follows:

$$c(n) = 2H(n-1) - \frac{2(n-1)}{n} \qquad \ldots 1.2$$

where n is a total number of data points and H(i) is a harmonic number calculated by a summation of Euler's constant and ln(i). An average score for each data-point across all trees is the final anomaly score estimate.

b. Local Outlier Factor (LOF)**:** This algorithm is an unsupervised anomaly detection method which computes the local density deviation of a given data point with respect to its neighbors. It considers as outliers the samples that have a substantially lower density than their neighbors. It is useful for density-based anomalies, implemented using scikit-learn Library. The formula is as follows:

$$LOF_k(p) = \frac{\frac{1}{|N_k(p)|}\sum_{o \in N_k(p)} lrd_k(o)}{lrd_k(p)} \qquad \ldots 1.3$$

Where $LOF_k(p)$ is the Local Outlier Factor for point p using k neighbours. Here $N_k(p)$ is the set of k -nearest neighbors of p; the k -distance (o) is the distance from (o) to its k-th nearest neighbor; the reachability distance is $reach - dist_k(p, o) = \max\{k - distance(o), d(p, o)\}$.

The LOF score compares the average local density of p's neighbors to p's own local density: values $\approx$ 1indicate inliers, and values > 1 indicate outliers (larger $\rightarrow$ more anomalous). Choose k to reflect the expected local neighborhood size; the LOF score is already interpretable as a relative density ratio, so it is commonly used directly as the anomaly score.

c. One-Class SMV: The One-Class SVM algorithm is an unsupervised anomaly (novelty) detection method which learns a decision boundary around the majority of the data in a (possibly kernel-transformed) feature space. Points that fall outside this learned region are considered outliers.

It is particularly useful when the training set is assumed to contain mostly normal examples; performance depends on kernel choice and scaling and it is implemented using the scikit-learn Library. The formula is as follows;

$$S_{OCSVM}(x) = \rho - w^{\top}\phi(x) \qquad \ldots 1.4$$

where $S_{OCSVM}(x)$ is the anomaly score for point x, w, ρ are learned parameters,

$\phi(\cdot)$ is the kernel feature map.

The decision function is $g(x) = w^{\top}\phi(x) - \rho;$ … 1.5

points with $g(x)$ less then 0 are considered outliers,

the negative decision value is used as the anomaly score (larger $\Rightarrow$ more anomalous).

In training, $\nu \in (0,1]$ controls the expected fraction of outliers.

(4) Model evaluation: The following evaluation metrics were adopted to access the performance of the proposed model;

a. Silhouette Score: The Silhouette Score measures how similar a point is to its own cluster compared to the nearest other cluster.
b. Anomaly Ratio: The Anomaly Ratio is a simple descriptive statistic calculated by dividing the total number of flagged anomalies by the total number of transactions in the dataset. The Anomaly Ratio is simply the proportion of transactions flagged as anomalous:

c. Average Decision Score : The term Average Decision Score in unsupervised learning is a descriptive metric, not a single, universally defined function. It is simply the average of the raw decision function scores or anomaly scores generated by the model across the entire dataset.

1.4 SIGNIFICANCE OF STUDY

This research contributes to knowledge by focusing specifically on bank-transfer anomalies, comparing simple machine learning models in low-resource settings, documenting a realistic simulation approach for performance evaluation, and proposing an actionable integration framework that connects anomaly scores to real product use cases.

1.5 SCOPE OF THE STUDY

This study focuses on applying Isolation Forest, Local Outlier Factor and One Class SVM models to detect anomalies in bank transactions using bank transaction dataset from Kaggle.

CHAPTER TWO

LITERATURE REVIEW

2.0 INTRODUCTION

This chapter reviews the technical and empirical literature on anomaly detection in financial settings, with particular emphasis on machine learning algorithms (supervised, unsupervised, and graph/temporal models), evaluation practices that reflect business cost, and persistent gaps that motivate this work.

2.1 CONTEXTUAL BACKGROUND AND DEFINITIONS

This section defines the core terms used in this study, situates bank transfers within the payments ecosystem, and highlights regulatory and operational considerations that shape how anomaly detection systems must be designed and evaluated. The goal is to provide a shared vocabulary and a practical understanding of the domain so later methodological choices (features, models, evaluation) are clearly motivated.

1. **Anomaly / Outlier:** An anomaly (or outlier) is an observation that deviates markedly from the expected pattern in the data and therefore may indicate unusual or suspicious behaviour (Bakumenko, 2022). In transactional contexts, anomalies can appear as unusual amounts, a typical timestamps, abnormal counterparties, or unexpected sequences of transfers. Importantly, anomalies are a descriptive class, they signal deviation but do not on their own establish intent or illegality.
2. **Transfer fraud (transfer-specific anomalies):** Fraud refers to intentionally deceptive acts undertaken to obtain an unfair or unlawful financial advantage. Transfer fraud denotes fraudulent or abusive behaviour that specifically exploits money-transfer rails (e.g., interbank transfers, mobile-money remittances, remittances to third-party accounts). Transfer fraud often manifests as rapid movements of funds through many accounts (layering), frequent small-value transfers intended to avoid thresholds, or collusive rings that route payments through networks of accounts. Network-aware analyses are therefore particularly useful in this subdomain because relational patterns (who transacts with whom and when) are often as informative as single-transaction features (Pourhabibi et al., 2020; Elliott et al., 2019).
3. **False Positives and False Negatives:** In detection systems, a false positive is a legitimate transaction incorrectly flagged as anomalous or fraudulent; a false negative is a fraudulent transaction that the system fails to detect. The operational costs of false

positives (customer friction, manual review workload) and false negatives (monetary loss, regulatory penalties) differ substantially and should inform model selection and thresholding decisions. Cost-sensitive learning frameworks explicitly incorporate these asymmetric consequences into training or decision-making (Höppner et al., 2020).

2.2 DIGITAL BANKING FRAUD LANDSCAPE

The rise of digital banking has transformed how financial transactions are initiated and completed, especially with the widespread use of mobile banking, internet banking, payment apps, and automated interbank transfer systems. These platforms make transfers faster and more convenient, but they also introduce new risks that affect how fraud detection systems must be designed. Bank transfers now move through layers of payment networks involving originating accounts, clearing systems, and receiving accounts, and in some cases, mobile-money and third-party payment channels. This variety of channels and transaction types introduces different transaction behaviours across platforms, making it difficult to apply one fixed detection model to all systems (Desai & Kosse, 2024; Azamuke et al., 2025).

A major challenge in this landscape is high transaction volume, digital banking systems process thousands to millions of transfers daily due to this, a very low rate of missed fraud cases (false negatives) can lead to large total financial losses. At the same time, flagging too many legitimate transfers as fraud (false positives) increases operational costs and creates customer dissatisfaction, since such cases require manual investigation before funds can proceed (Desai & Kosse, 2024). This means fraud detection models must be both accurate and efficient, balancing security with smooth user experience.

Fraud in bank transfers often occurs through networked behaviour instead of isolated individual transactions. Fraudsters frequently use linked accounts, mule networks, shell accounts, and fast multi-hop transfers to move money and hide the source of funds. These patterns are best detected when account relationships and transaction flows are analyzed together rather than looking at each transaction alone (Pourhabibi et al., 2020; Elliott et al., 2019). This is why modern research increasingly makes use of graph-based and flow-based anomaly detection methods, which examine how accounts interact over time.

Another important part of the digital banking fraud environment is that fraud evolves quickly. Once banks detect and block a method, attackers shift to new patterns. This leads to a concept drift problem, `where the fraud patterns seen during model training may no longer match those

seen later in production. As a result, fraud detection models must be updated routinely or designed to adapt to changes in behaviour (Takahashi et al., 2024; Bakumenko, 2022).

Digital banking operates under strict regulatory conditions. Anti-Money Laundering (AML) and Know Your Customer (KYC) rules require banks to monitor transaction behaviour, generate alerts for suspicious events, and maintain evidence for audits and reporting. Detection systems therefore must not only be accurate but also explainable and transparent enough for human investigators to justify decisions (Höppner et al., 2020).

The digital banking fraud landscape is shaped by high transaction volumes, diverse transaction channels, evolving fraud techniques, limited labeled data, and strong regulatory oversight. These factors justify the need for anomaly detection models that are adaptable, cost-aware, and capable of capturing both transaction-level which is the focus of the present study.

2.2.1 REGULATORY AND OPERATIONAL CONSIDERATIONS

Regulatory frameworks such as **Anti–Money Laundering (AML)** requirements mandate that financial institutions monitor suspicious activity, report certain transactions, and maintain customer due diligence records. Fraud-detection systems therefore must produce outputs that support compliance workflows; prioritized alerts, explainable reasons for suspicion, and audit trails for investigations. Regulatory expectations also affect retention, data sharing, and the acceptable balance between false-positive and false-negative rates (Höppner et al., 2020).

Transactional data are highly sensitive, privacy constraints and data-protection laws (e.g., regional privacy statutes) can limit access to raw identifiers and restrict cross-institutional data sharing. These constraints motivate approaches such as feature-level anonymization, federated learning, and use of synthetic or simulator-based datasets when developing and validating models (Azamuke et al., 2025; Adedoyin, 2018). Where raw data cannot leave a host institution, evaluation must rely on internally approved experiments or on simulated data that approximates real dynamics.

Misclassification (e.g., incorrectly labelling a legitimate customer or freezing funds) can cause reputational and legal harm. Any deployment plan must therefore include human-in-the-loop review, clear escalation protocols, and mechanisms for customers to contest or resolve flagged actions.

2.3 MACHINE LEARNING-BASED ANOMALY DETECTION TECHNIQUES

Machine learning (ML) has become the dominant paradigm for anomaly detection in financial systems because it can learn complex patterns from large volumes of transactional data and combine many weak signals into robust alerts. Broadly, ML approaches fall into supervised and unsupervised families (with semi-supervised bridging them). The choice between these paradigms is driven by label availability, the nature of anomalies to be detected (point vs. structural), operational constraints (latency, explainability), and business objectives (minimize monetary loss vs. maximise statistical metrics). Below are the supervised and unsupervised classes that are most relevant for bank-transfer detection and highlight practical issues especially class imbalance, thresholding and evaluation that determine real-world effectiveness.

2.3.1 SUPERVISED LEARNING

Supervised approaches treat fraud detection as a standard classification problem, labeled examples (legitimate vs. fraudulent transactions) are used to train models that predict a probability (or score) of fraud for new transactions. Common algorithms applied in financial settings include logistic regression (baseline interpretable model), tree ensembles such as Random Forest and gradient-boosted machines (XGBoost, LightGBM) for strong predictive performance on tabular data, and support vector machines or neural networks where nonlinearity and feature interactions are important (Bakumenko, 2022; Maneel, 2025). Supervised models are great because they can be tuned to optimise for business metrics, incorporate cost-sensitive loss functions, and deliver high precision when reasonably sized, high-quality labeled datasets exist.

However, transfer fraud detection faces a set of well-known practical challenges for supervised learning. The class imbalance problem; where fraudulent events are extremely rare relative to legitimate transactions undermines naive training and evaluation.

2.3.2 UNSUPERVISED LEARNING

When labeled fraud data are scarce, incomplete, or unreliable, unsupervised or novelty detection methods are the default choice. These methods model "normal" behaviour and flag deviations without requiring explicit fraud labels. They are particularly useful for discovering new or emerging fraud patterns that were not present in historical labels.

a. **Isolation Forest**: This is a tree-ensemble method that isolates observations by random partitioning; anomalies are easier to isolate and thus obtain shorter average path lengths in the isolation trees. It is computationally efficient, scales well to large datasets, and is widely used in payments environments for initial triage (Bakumenko, 2022). Its main strengths are speed and minimal preprocessing requirements; limitations include sensitivity to feature scaling and the need for careful threshold selection to control alert volumes.
b. **Local Outlier Factor (LOF):** This is a density-based method that compares the local density of a point to the densities of its neighbors; points with substantially lower local density are scored as outliers. LOF can detect contextual and local anomalies that global methods miss, making it useful when fraud presents as small pockets of abnormal behaviour. However, LOF's computational cost grows with data size and it requires meaningful distance metrics; it can also be noisy in high-dimensional spaces unless dimensionality reduction or careful feature engineering is used (Stefánsson, 2023).
c. **Clustering-based methods** (k-means, DBSCAN and related algorithms): This groups similar transactions together; small clusters or points far from cluster centroids can be treated as anomalies. DBSCAN, with its density notion and ability to find arbitrarily shaped clusters, is useful for discovering dense regions of legitimate behaviour while isolating sparse fraud clusters. Clustering approaches tend to be intuitive and easy to inspect but require thoughtful selection of distance measures and cluster-scale parameters, and they may struggle when legitimate behaviour itself is highly heterogeneous.
d. **Autoencoders (AE) and variational autoencoders (VAE)**: These are neural networks trained to reconstruct inputs, unusually large reconstruction errors indicate novelty. Autoencoders can learn nonlinear, high-dimensional representations of normal transfer behaviour (Parimi, 2017), making them powerful for detecting subtle anomalies that classical methods miss. In practice, AE performance depends on careful architecture design, regularisation to avoid trivial identity mappings, and sufficient "normal" training data. They can also be extended to sequence autoencoders for temporal patterns.

While unsupervised methods are essential when labels are absent, they typically produce higher false positive rates and require human tuning of thresholds to produce manageable alert volumes. Combining unsupervised scores with downstream supervised filters or manual rules (hybrid pipelines) often improves precision and investigator efficiency (Maneel, 2025). Evaluation for unsupervised methods is also challenging; common practice includes synthetic

injection of labeled anomalies for benchmarking, use of proxy labels, or retrospective validation on known incidents (Stefánsson, 2023).

2.3.3 SEMI-SUPERVISED

Machine-learning research for financial anomaly detection increasingly sits between purely supervised and purely unsupervised methods. Semi-supervised and one-class methods, deep sequence models, graph/network methods, and hybrid/ensemble systems each address important practical constraints of bank-transfer fraud detection namely but not limited to label scarcity, temporal structure, relational fraud rings, concept drift, and the need to balance detection power with operational costs.

Semi-supervised and one-class approaches assume that labelled fraud is rare or unreliable and instead rely mainly on examples of "normal" behaviour. One-Class SVM (OC-SVM) is a classical technique that learns a decision boundary around the majority (presumed normal) class and flags points outside this boundary as anomalies. OC-SVM is best where legitimate activity is plentiful and stable, but its performance degrades in high dimensions or when "normal" behaviour is heterogeneous (Bakumenko, 2022).

Operationally, semi-supervised/one-class systems are commonly deployed as first-line triage: they cast a wide net to capture unknown fraud patterns, then feed candidates into higher-precision supervised filters or analyst review (Lokanan, 2023; Maneel, 2025).

2.3.4 DEEP LEARNING AND SEQUENCE MODELS

Many fraud patterns are temporal, sequences of transfers, bursts of activity, or timed routing through mule networks. Sequence models RNNs and LSTMs in particular, capture temporal dependencies by learning representations over ordered transaction windows. Applied studies show LSTM and sequence autoencoders can flag anomalies that static models miss (Parimi, 2017). These models can be trained to reconstruct transaction sequences or to predict next-step behaviour; unusually large prediction or reconstruction errors indicate anomalies.

Deep sequence models are powerful but come with practical trade-offs as they require substantial training data (or careful pretraining), are computationally heavier for near-real-time scoring, and raise explainability concerns. There is often recommendations of combining sequence models with interpretable feature summaries (rolling aggregates, velocity measures) and using them in a layered architecture where fast, lightweight detectors perform initial screening (Desai & Kosse, 2024; Maneel, 2025).

More recent techniques include Graph Neural Networks (GNNs), which learn node and edge embeddings that encode structural and attribute information; these embeddings can be fed to anomaly scoring or downstream classifiers. Surveys and comparative studies of anomaly detection literature emphasizes that graph methods detect rings, mule networks, and other relational patterns with higher precision than purely tabular methods, especially when temporal dynamics are combined with topology (Pourhabibi et al., 2020; Elliott et al., 2019). Oye et al.'s work on data-dependence graphs illustrates how richer graph constructions (beyond simple transfer edges) can improve signal extraction for ML classifiers, suggesting avenues for creating features that capture flow dependencies and multi-hop relationships.

Hybrid systems also facilitate operational requirements, rule layers encode regulatory or known fraud triggers that must always be escalated, ML layers provide adaptive detection and scoring; and human analysts validate high-risk alerts. Cost-sensitive decision frameworks (Höppner et al., 2020) are particularly important in ensemble settings because they help set thresholds and component weights to optimise monetary outcomes rather than purely statistical metrics.

2.4 FEATURE ENGINEERING AND DATA REPRESENTATION FOR BANKING TRANSFERS

Feature engineering is the bridge between raw bank-transfer logs and any machine learning model that will try to flag anomalous transfers. Raw transfer records often contain the same basic fields i.e timestamps, sender and recipient identifiers (usually masked or hashed), amounts, currencies, channel (online, branch, agent), and sometimes device or location metadata but these raw values rarely separate normal from fraudulent behaviour on their own. Converting those raw values into informative features is therefore the single most important step in practically every study of transfer anomaly detection (Bakumenko, 2022). Good features make patterns explicit: they turn hidden signals (like sudden bursts of activity, repeated small transfers, or unusual counterparty patterns) into numbers a model can learn from.

A useful way to think about features is by family.

1. Simple transaction-level features: log-transformed amount, currency, transaction type and hour-of-day or day-of-week indicators, because monetary amounts are heavily skewed, studies typically log or scale amounts so that large transfers do not dominate model behaviour (Bakumenko, 2022).
2. Temporal aggregates, counts and sums over recent windows such as the last hour, 24 hours, and 7 days; velocity measures such as change in count between windows; and

time-since-last-transaction. These rolling-window features capture spikes and bursts that are common in laundering or mule networks (Lokanan, 2023; Desai & Kosse, 2024).

3. Sequence features: short ordered histories of amounts and inter-arrival times that can be fed to sequence models or summarized into statistics (entropy, periodicity). Reconstruction or prediction residuals from sequence models often serve as strong anomaly indicators (Parimi, 2017; Takahashi et al., 2024).

Certain specialized features repeatedly appear across the literature I reviewed because they capture behaviors known to be fraud signals. Counterparty diversity (how many unique recipients an account sends to), ego-flow (sum of amounts in a k-hop neighborhood), motif counts (chains of transfers), and forecast residuals from link/flow models are all strong predictors of anomalous routing or layering (Elliott et al., 2019; Takahashi et al., 2024; Pourhabibi et al., 2020). Historical fraud-rate features or expected-loss proxies for example, the historical fraction of flagged transactions involving a counterparty help cost-sensitive methods prioritise high-impact alerts (Höppner et al., 2020). Device and location fingerprints, when available and privacy-permitted, also add discriminative power for account-takeover or a typical access pattern (Azamuke et al., 2025).

Privacy and operational constraints shape feature choices in practice. Raw identifiers and sensitive attributes are often unavailable for sharing, so many teams either anonymize at the feature level, compute graph summaries locally and share only aggregates, or use simulation/federation for cross-institution experiments (Adedoyin, 2024; Azamuke et al., 2025). Simulation can be invaluable for validating detectors on rare events, but Adedoyin, 2024; Azamuke et al., 2025 cautions that simulator parameters must be tuned to real distributions to avoid over-optimistic.

2.5 RELATED WORKS

Lokanan (2023) aimed to evaluate classical and ensemble machine-learning classifiers on mobile-money transfer fraud, using a large PaySim-style transactional dataset to mimic mobile-money fraud patterns. The study tested logistic regression, decision trees, random forest and gradient-based models, applying normalization and SMOTE-ENN balancing; evaluation used MCC and AUC among other metrics. Results reported model accuracies in the 0.82–0.89 range and MCC values of 0.67–0.78, with Random Forest performing best (0.89 accuracy, MCC 0.78). The authors highlighted the usefulness of transaction amount as a top predictive feature

but also cautioned that synthetic PaySim-style data limit external validity, and that simulation biases may overstate performance in real deployments.

Bakumenko & Elragal (2022) investigated anomaly detection in financial/general-ledger data by implementing classical and modern approaches (logistic regression, SVM, Random Forest, KNN, Isolation Forest, autoencoders) on an enterprise journal-entry dataset. They followed a CRISP-DM pipeline, engineered time and categorical features, and compared supervised and unsupervised methods; the thesis reports detailed confusion-matrix counts and per-model recall/coverage. The supervised Random Forest achieved top performance (recall avg-macro 0.9925, essentially full coverage of induced anomalies) while unsupervised Isolation Forest attained strong coverage with acceptable false-positive rates (recall 0.9555), illustrating that well-tuned ensembles can discover rare but known anomaly types in ledger data. Bakumenko stresses limits: many anomalies were synthetically injected and the dataset is highly imbalanced (0.46% anomalies), so results may not generalize to larger, noisier banking transfer logs.

Höppner et al. (2020) presented instance-dependent cost-sensitive classifiers (cslogit, csboost) that directly optimize an economic objective rather than standard accuracy metrics; they argue that the true business goal in transfer fraud is minimizing monetary loss and that misclassification costs vary per transaction (e.g., by amount). Using the Kaggle credit-card dataset and a proprietary bank dataset, their experiments show sizable practical gains: cslogit achieved expected-savings 71.6% vs. logistic 64.8%, and csboost similarly improved over unconstrained xgboost; in the Kaggle benchmark they also reported precision 43.5% (cslogit) vs. 36.6% (logit) and F1/recall gains consistent with better cost outcomes. The key takeaway is that optimizing for cost (monetary savings) can change model choice compared to optimizing AUC/F1 but the approach needs accurate per-instance cost estimates (which are often hard to obtain) and the best results depend on realistic cost models.

Azamuke et al. (2025) used an agent-based simulator (MoMTSim) to generate rich mobile money transaction datasets representative of Sub-Saharan usage and then evaluated classical and gradient-boosted models. Their benchmark shows XGBoost produced the strongest results (reported AUC 0.97 and MCC 0.82, F1 0.89 in their experiments), with logistic regression serving as a reasonable baseline (MCC 0.67). The chapter's strength is the realistic simulation of regional attack scenarios (agent behaviors, network-level flows), which lets the authors examine feature importance (amounts, frequency, operator). Limitations: synthetic/urban-biased sampling and heavy reliance on simulated injections, so external validity to other regions or real bank transfers can be limited.

Studies focused on unsupervised and reconstruction approaches show complementary strengths. Stefánsson (2023) and Parimi (2023) documented operational pipelines where Isolation Forests, LOF, clustering and autoencoder reconstructions are used to surface novelties when labels are unavailable; Parimi in particular finds sequence-reconstruction autoencoders capture enterprise-specific temporal patterns better than proximity methods. These works stress the necessary calibration steps (injection tests, proxy labeling) to convert raw anomaly scores into usable alerts. The recurring limitation is high false-positive rates unless careful thresholding and validation are applied (and the need for human-in-the-loop verification).

Graph-based approaches and network methods form a distinct strand. Pourhabibi et al.'s systematic review (2020) synthesizes the GBAD literature and demonstrates that graph methods (subgraph detection, spectral features, GNNs) are particularly powerful for uncovering collective, ring-like collusion and mule networks that transaction-level detectors miss; they also catalogue gaps around scalability and reproducibility in GBAD research.

Elliott et al. (2019) proposed a hybrid spectral plus local-statistic feature set (140 features) and show on large synthetic benchmarks that their feature-sum and Random Forest classifiers outperform existing baselines (e.g., Oddball): their key reported outcome is that the top 2% of flagged nodes captured over 90% of planted anomalies on their synthetic tests, demonstrating excellent ranking power for operational triage. These graph methods, however, typically need multi-hop metadata and richer link contexts (which are often unavailable across institutions) and are computationally heavier than tabular models.

Takahashi et al. (2024) developed a dynamic link-and-flow prediction model (temporal-topological self-attention with hierarchical softmax) that forecasts both the probability of future edges and their weights. Evaluated on a public cryptocurrency dataset and a proprietary bank transfer dataset, their DLF models outperform simple baselines (EdgeBank). For remittance-ratio prediction they report average cross-entropy improvements (e.g., DLF-Hier vs EdgeBank: Ethereum 3.431 vs 5.213; Bank 3.480 vs 3.577) and for link formation/dissolution they report AUCs in the 0.65–0.89 range depending on thresholding (e.g., bank link formation AUC up to 0.892 at a stricter threshold), showing promise for forecasting expected flows that can be converted into anomaly scores (deviations from forecast). The complexity and need for node-level history are trade-offs for accuracy.

Oye et al. (2024) and Maneel (2025) report, applied pipeline work that combines data-dependence graphs and engineered network features with conventional classifiers. Oye's empirical experiments (feature extraction with NetworkX + SVM classifiers) show graph features can improve recall by roughly 10 -12% over baseline tabular classifiers when full metadata is available. Maneel's recent system engineering work documents a transaction anomaly detection pipeline using careful feature engineering and Random Forests with real-time review, reported F1 scores and confusion matrices show robust detection for the targeted scenarios, but both studies emphasize the operational cost (graph construction time, data completeness requirements) and the practical difficulty of obtaining multi-institution link data.

Parimi (2023) surveys deep-learning and sequence models and argues that RNN/LSTM and Transformer-based architectures are attractive for temporal pattern learning in transfers, while autoencoder-style reconstructions help when labelled fraud is scarce. Parimi underscores interpretability and latency concerns, deep sequence models can improve detection of subtle temporal anomalies but are harder to deploy for near-real-time scoring without model compression or careful engineering. Parimi's synthesis recommends hybrid setups (reconstruction + downstream classifier) to get the best of both worlds.

Adedoyin, (2018) demonstrated the practical value and limits of simulated PaySim-style datasets and case-based reasoning approaches. Adedoyin's CBR experiments show high AUC and MCC on simulated datasets (Adedoyin reports AUC 0.98-0.99, MCC 0.96-0.99 in injected tests), but they also make the well-known point that synthetic injection often overestimates real-world performance and that deployment requires domain-specific validation built around human analysts and escalation protocols.

Across the reviewed works the field converges on several practical lessons which includes; supervised ensembles (Random Forest, XGBoost) are strong baselines when realistic labels exist (Bakumenko; Lokanan; Azamuke). Unsupervised/reconstruction methods are essential for novelty detection when labels are scarce (Stefánsson; Parimi). Graph and temporal models uniquely detect collective and multi-step fraud (Pourhabibi; Elliott; Takahashi), and Cost-sensitive learning (Höppner et al.) aligns model objectives with business outcomes and can substantially reduce expected monetary loss. Yet, reproducibility is impeded by the scarcity of open, realistic bank-transfer benchmarks; many promising results rely on synthetic or proprietary datasets. These gaps motivate the design choices in your project combining network + temporal features, evaluating on realistic (or institutionally-approved) transfer logs, and optimizing for cost-aware metrics rather than raw AUC alone.

2.6 GAPS IN CURRENT METHODOLOGIES

The reviewed literature has advanced anomaly detection for bank transfers substantially, but clear gaps remain. First, there is a persistent shortage of public, realistic benchmarks for transfer networks**.** Many graphs and temporal studies rely on proprietary logs or synthetic datasets, which hurts reproducibility and makes it hard to compare methods fairly (Elliott et al., 2019; Pourhabibi et al., 2020). While simulators and synthetic data are useful, studies repeatedly warn that simulators must be carefully calibrated to real-world behaviour or they will give over-optimistic results (Adedoyin, 2018; Azamuke et al., 2025).

Label scarcity and label latency remain fundamental problems for supervised approaches. Confirmed fraud labels are rare and often delayed (investigations, chargebacks), so models trained and evaluated on labeled snapshots may not reflect live performance. This motivates unsupervised and semi-supervised techniques, but those approaches bring their own challenges (high false-positive rates, difficult calibration) and are not a full substitute for robust labeled evaluation (Lokanan, 2023; Stefánsson, 2023).

Evaluation metrics are often misaligned with operational goals**,** most academic work reports AUC or F1, yet banks care about monetary loss, alert volume, and investigator workload. Cost-sensitive methods that optimise expected loss show promise, but they require realistic per-instance cost estimates and are not yet broadly adopted in evaluations (Höppner et al., 2020). Without standard use of monetary and operational metrics, research risk selecting models that look good on paper but fail in production.

Graph and deep-sequence methods face computational and privacy constraints, GNNs and temporal-topological models can detect relational and flow-based fraud that tabular models miss, yet they are expensive to compute on large dynamic graphs and raise privacy or data-sharing barriers across institutions (Takahashi et al., 2024; Oye et al., 2024). Many papers therefore compute graph summaries offline or on a per-host basis a practical compromise that reduces but does not eliminate the underlying limitation (Desai & Kosse, 2024).

Cross-channel and cross-institution generalisability is underexplore, models trained on card, mobile-money, or enterprise logs often do not generalize to other rails without re-tuning; yet multi-channel, multi-institution evaluation is rare due to data sensitivity and heterogeneity (Azamuke et al., 2025; Gopalsamy, 2025). This gap restricts the practical portability of published methods.

Given these gaps, several concrete research directions emerge:

1. Open, realistic benchmarks and agreed evaluation protocols**:** The community would benefit from shared, privacy-preserving transfer-network benchmarks (or standardized synthetic suites with well-documented parameters) and from evaluation protocols that include monetary loss, alert volume, and latency in addition to AUC/F1 (Elliott et al., 2019; Adedoyin,2018).
2. Adoption and refinement of cost-sensitive evaluation**:** Wider use of instance-dependent cost models will help align research outcomes with business value. Work is needed on robust ways to estimate per-instance costs and on sensitivity analyses that show how model choice changes under different cost assumptions (Höppner et al., 2020).
3. Robust online learning and concept-drift handling: Practical fraud detection requires models that adapt to evolving adversaries. Work on drift detection, continual learning, and safe online updates (with rollback and human oversight) will make deployment more robust (Bakumenko, 2022; Takahashi et al., 2024).
4. Better simulation and transferability studies**:** Improve simulator realism and publish parameter sets so others can reproduce injected scenarios; perform transferability tests that compare simulator-trained models against small, real-world holdout sets to quantify how simulation bias affects performance (Azamuke et al., 2025; Brighton University, 2024).
5. Explainability for graph and sequence models**:** Develop practical interpretability techniques for GNNs and temporal models so alerts can be explained to investigators and regulators a necessary step for compliance and analyst trust (Parimi, 2023).

Addressing these gaps will move research closer to deployable, trustworthy, and comparable systems that financial institutions can use with confidence. This research contributes to knowledge by focusing specifically on bank-transfer anomalies, comparing simple machine learning models in low-resource settings, documenting a realistic simulation approach for performance evaluation. Table 1 Illustrates some selected literature based of their authors, objectives, methodologies, results obtained and limitations.

Table 2.1 Some Reviewed Literature

AUTHORS	OBJECTIVES	METHODOLOGY	RESULTS	LIMITATIONS
Lokanan, (2023)	Compare machine learning classifiers models on simulated mobile money PaySim dataset for fraud detection.	Evaluated Decision Trees, Logistic regression, Random Forest, and Gradient descent on PaySim dataset (1,048,575 data samples with 9 features) Minmax Scalar was used for normalization, Smote-ENN used for data balancing. MCC, AUC employed for performance evaluation.	The study gave a novel approach to cross validate classifier models in transactional data and mobile specific patterns and behaviours. The models achieved a performance accuracy ranging from 0.82 - 0.89 and MCC of 0.67 – 0.78 with Random forest having the highest at 0.89 accuracy and 0.78 MCC	Synthetic datasets did not represent the population of interest. Model was bias. Only few machine learning models were considered, other methods of fraud detection like human intelligence were not considered.
Bakumenko & Elragal, (2022)	Implement ML models for anomaly detection in general ledger data.	Processed real bank transaction logs using Python (Pandas, NumPy); feature extraction on timestamps and amounts; implemented Isolation Forest (Scikit-learn), One-Class SVM (Scikit-learn), and Autoencoder (TensorFlow/Keras).	Isolation Forest achieved 93% detection accuracy; Autoencoder offered lowest false-positive rate.	Study used scares datasets. Limited features selected
Hoppner et al, (2020)	Develop a cost-sensitive anomaly detection model that weighs each transaction	Lasso-regularized logistic regression and Gradient boosted decision trees classifiers were used to minimize cost,	Proposed two new classifiers called cslogit and csboost which are based on lasso-regularized logistic regression and	Only two instance-dependent cost-sensitive classifiers were explored.

	by its potential loss.	cslogit and csboost algorithms were introduced for instance-dependencies. Used public datasets from Kaggle (284,807 transactions with 492 frauds)	gradient tree boosting, respectively. In expected savings Cslogit outperformed logistic regression with 66.64% - 46.16% and csboost outperformed gradient boosted trees with 64.76% - 59.6%	Difficulty in defining accurate penalty costs for each transaction.
Azamuke et al., (2023)	Evaluate ML Models for fraud detection in synthetic MoMTSim Datasets.	Data cleaning and preprocessing with Python (Pandas); feature engineering (transaction amount, frequency, network operator) using Scikit-learn; model training with Random Forest and XGBoost (XGBoost library); evaluation using cross-validation; visualization with Matplotlib and Seaborn.	XGBoost outperformed with 89% F1-score; feature importance highlighted unusual network patterns.	Dataset only contains urban users and this is a bias. Anomalies were injected synthetically and may not reflect real fraud.
Adedoyin, (2018)	Propose and evaluate a Case-based Reasoning model to predict fraudulent mobile-money transactions.	Comprehensive literature review to establish research questions. Synthetic mobile-money transfer dataset generation (PaySim-style).	A novel approach to detecting fraud in mobile money payment networks using Case-based reasoning methodology. Model achieved performance AUC ranging from 0.98% - 0.99% and	Study only focused on CBR methods other methods such as ANN, DL were not considered. Relied only on synthetic data, so real-world dataset may differ.

		LR,RF,SVM,ANN classifiers were used for experiments. MASON was used for data simulation. F-Measure, MCC, AUC used for performance evaluation.	MCC ranging from 0.96% - 0.99%	The CBR with CURE algorithm showed low accuracy compared to without it. It required a considerable amount of computational power.
Pourhabibi et al., (2020)	Propose a framework to synthesize existing literature on the application of GBAD methods in fraud detection.	Conducted PRISMA review of 75 published studies on financial fraud; used bibliometric tools (Zotero, VOSviewer) for thematic coding of graph techniques; no primary data.	Identified GNNs and subgraph detection as top-performing; summarized gaps in scalability.	Some relevant studies were omitted owing to the limitations of the scientific database, specific keywords employed in the search and timeframe of the study.
Takahashi et al., (2024)	Predict future transaction links and volumes within banking networks using temporal graph neural networks.	Implemented structural embedding for node positioning. Transfer embedding for operational statue of nodes, Shalow and Wide hierarchical softmax tree to mitigate error. Used projection mechanism for estimating the total amount of node transmission per step. AUC-ROC used to	A novel model for predicting dynamic links and flows in networks. Model outperformed existing models with a 0.892% on link formation and 0.657% on link dissolution.	Only Graph methods was explored and it required complex graph architectures.

		measure model performance. Real bank transfer datasets (1,000 samples) and crypto transfer data from public domain.		
Oye et al., (2024)	Combine data-dependence graphs with ML to detect anomalies in serial transactions	Constructed data-dependence graphs with NetworkX; extracted graph features (centrality, path length); classification using Support Vector Machine (Scikit-learn); evaluation metrics computed with Scikit-learn; scripting in Python.	Graph features improved detection recall by 12% over baseline ML models.	Graph construction is expensive; approach requires complete transaction metadata.
Parimi, **(**2023)	The study aims to integrate deep learning techniques to enhance anomaly detection accuracy and efficiency within SAP environments	Comprehensive literature review. Comparison between DL techniques and traditional techniques in anomaly detection. DL techniques reviewed includes RNNs, CNNs and Autoencoders.	This study exposed a comprehensive review of deep learning architectures tailored for financial anomaly detection, addressing challenges such as interpretability, scalability, and adaptability to real-time transaction monitoring.	Study lacks technical depts and no implementation.
Tejraj & Ghimire, (2025)	Detect anomalies in letters of credit and bank	Text and numeric feature extraction using Python (NLTK for text, Pandas for numeric); anomaly detection via Isolation	Isolation Forest flagged 95% of injected laundering cases; LDA aided explainability.	Only trade documents were explored not general bank transfers.

	guarantees to flag ML risk.	Forest (Scikit-learn); topic modeling with Latent Dirichlet Allocation (Gensim); evaluation with precision-recall metrics.		
Elliott et al., (2019)	Propose a method able to detect previously unspecified anomalies in financial transaction networks.	Synthetic data set generation through weighted directed Erdős-Rényi random graph model and Accenture model (55,000 nodes). Implemented FEATURE SUM and Random forest to classify nodes. Used network comparison and spectral analysis (140 main features.) Comparison between Oddball, FEATURE SUM and RF	A novel anomaly detection method which combines spectral approaches with recent advances in network comparison, to uncover unknown anomalies in networks. RF and FEATURE SUM outperformed RC and Oddball algorithms by extracting 92.3% anomalous nodes within the top 1.86% ranking nodes.	Methods is only tailored for static networks. Limitation in the ability to distinguish between nodes caused by the limited number of trees in a default RFR. Method is expensive to compute and it is not as fast as required with the choice of programming language (python) used been a factor.
Maneel et al., (2025)	Propose a Transaction Anomaly Detection system.	Random forest classifiers, Feature Engineering; advance scaling methods, and transaction amount difference calculation. Scikit-learns random forest classifier for solution. Standard scalar for feature normalization.	Developed a reliable system that can identify minute anomalies using RF and feature engineering. Implemented a real-time transaction review via command-line interface.	Only a few machine learning approaches were used

		F1 score and confusion matrix for performance analysis.		

2.7 PERFORMANCE METRICS AND BENCHMARKING STUDIES

Choosing the right evaluation metrics is critical for any anomaly-detection project because the metrics determine which model is trusted and how it will be tuned. In the bank-transfer setting we face three practical challenges: extreme class imbalance (fraud or real anomalies are rare), the absence of labelled fraud data in our dataset, and temporal dependence (behaviour changes over time). These facts mean that the common supervised metrics (precision, recall, F1, ROC-AUC) are not directly useful here. Instead, we used a small set of descriptive and comparative metrics designed for unlabeled anomaly detection: Silhouette Score**,** Anomaly Ratio**,** and Average Decision Score**.** These metrics let us measure how cleanly a model separates unusual behaviour from normal behaviour, control the alert volume, and check model confidence. (Bakumenko, 2022; Desai & Kosse, 2024).

Some of the proposed metrics for this study include;

a. **Silhouette Score**: The Silhouette Score measures how well two groups are separated in feature space.

For each data point the score (s) is defined as

$$s = \frac{b-a}{\max(a,b)} \qquad \ldots 2.1$$

Where:

s: The Silhouette Score for a single data point.

a: The mean intra-cluster distance (the average distance from the data point to all other points within the same cluster). This measures how well the point fits its own cluster.

b: The mean nearest-cluster distance (the average distance from the data point to all points in the next closest cluster). This measures how far away the point is from its neighboring cluster.

Scores range from (-1) to (+1): values near (+1) mean the point is well matched to its cluster and far from the other cluster; values near 0 mean it lies between clusters; negative values indicate likely misclassification. A higher Silhouette Score indicates the model produces a clearer separation between the detected anomalies and the normal transactions. We used this metric both as the primary comparison metric across models and as the objective when selecting the contamination value during the grid sweep (Bakumenko, 2022).

b. **Anomaly Ratio**: The Anomaly Ratio is a descriptive statistic defined as;

$$\text{Anomaly Ratio} = \frac{\text{Number of flagged anomalies}}{\text{Total number of transactions}} \quad \ldots 2.2$$

Due to the fact that each model was trained with a contamination parameter (set to 0.05 in the final experiments), the anomaly ratio serves as an operational control to ensure the model returns a practical number of alerts. If the observed Anomaly Ratio departs widely from the expected contamination (e.g., much larger than 5%), it signals a mismatch between the model setting and actual behaviour and requires a re-check of preprocessing, feature scaling, or hyperparameters. The anomaly ratio is therefore used to keep alert volumes realistic for analyst review (Desai & Kosse, 2024).

c. **Average Decision Score (and Score Gap)**: Average Decision Score is a descriptive summary of the raw anomaly or decision-function outputs produced by each model. Models return continuous scores with different conventions, so we convert them to a consistent "higher = more anomalous" direction (for example, by negating score_samples for Isolation Forest). The metric is computed as the mean of the model scores across either the anomaly group or the normal group:

$$\text{Average Decision Score} = \frac{\sum_{i=1}^{N} \text{Decision Score}_i}{N} \quad \ldots 2.3$$

A larger positive gap means the model assigns clearly higher anomaly scores to flagged points than to normal points — i.e., the model is confident and the ranking is meaningful. Average Decision Score and Score Gap were used as secondary checks alongside Silhouette Score when choosing the final model: high silhouette + large score gap = strong evidence of good separation.

2.8 SUMMARY

This chapter reviewed the technical and operational literature on anomaly detection for financial transfers. The reviewed studies show a consistent set of findings which includes a mixture of supervised and unsupervised methods is necessary because labelled fraud is often scarce and delayed, graph- and temporal-aware approaches materially improve detection of collective and flow-based fraud compared with purely tabular models and cost-sensitive evaluation and layered rule plus ML architectures better align detection with business objectives than naive statistical metrics alone (Bakumenko, 2022; Pourhabibi et al., 2020; Höppner et al., 2020; Desai & Kosse, 2024).

Common limitations recur across the literature which includes but not limited to public realistic bank-transfer benchmarks are scarce. Label scarcity and label latency reduce the practical utility of fully supervised solutions unless complemented by careful resampling. Graph-heavy and deep sequence models often face computational and privacy constraints in production limiting multi-institution evaluation and real-time deployment without pre-aggregated graph features or efficient approximations.

Finally, many studies were evaluated using generic metrics (AUC/F1) rather than business-oriented loss measures, which can mislead model selection for operational settings (Höppner et al., 2020).

This project's methodology is designed explicitly to address those gaps by focusing specifically on bank-transfer anomalies, comparing simple machine learning models in low-resource settings, documenting a realistic simulation approach for performance evaluation, and proposing an actionable integration framework that connects anomaly scores to real product use cases.

CHAPTER THREE

METHODOLOGY AND SYSTEM DESIGN

3.0 INTRODUCTION

This chapter describes the methodology used to develop the anomaly detection system for bank transfers. The aim of the methodology is to provide a structured approach that ensures the reliability, transparency and reproducibility of the system. This chapter explains how the dataset was prepared, how new features were engineered, and how the machine learning models were developed and evaluated. The methodology followed a two-phase approach; Data Curation and Feature Engineering and Model Development and Performance Evaluation.

The first phase involved acquiring the transactional dataset, inspecting its structure, cleaning any irregular entries and transforming raw fields into meaningful behavioral variables. Feature engineering was especially important in this project because the dataset did not contain labels indicating which transactions were fraudulent or abnormal. Therefore, the models relied on the engineered features to learn the normal behavioral pattern of bank transfers (Stefánsson, 2023).

The second phase of the methodology focused on developing and evaluating three unsupervised machine learning algorithms: Isolation Forest, Local Outlier Factor (LOF), and One-Class Support Vector Machine (OCSVM). These models were chosen because they are commonly used in financial anomaly research and are effective when real fraud labels are unavailable (Bakumenko, 2022; Höppner et al., 2020). The trained models generated anomaly scores indicating whether each transaction appeared normal or suspicious. The performance of the models was then assessed using the Silhouette Score, Anomaly Ratio, and Average Decision Score, which allowed for a fair comparison of the models' separation ability and confidence. This guided the selection of the most reliable model for anomaly detection. The methodology described in this chapter establishes the foundation for the results and analysis presented in Chapter Four.

3.1 COMPARATIVE EXPERIMENTAL RESEARCH DESIGN

This project adopted an Experimental Research Design. The experimental design was appropriate because the study aimed to compare the performance of multiple machine learning models under the same conditions to determine which model performs best for anomaly detection in bank transfers. Since there were no fraud labels in the dataset, the task required

unsupervised learning, where the models learn the underlying structure of normal transactions and identify patterns that deviate significantly from this norm (Lokanan, 2023).

The experimental design allowed this project to systematically vary and evaluate conditions such as feature scaling, algorithm selection, and contamination rate (the expected percentage of anomalies). By training all three selected models namely, Isolation Forest, Local Outlier Factor, and One-Class SVM on the same feature-engineered dataset, the study made it possible to directly compare their performance using objective evaluation metrics. This selection aligns with common practices in financial anomaly research where unsupervised learning is preferred in the absence of reliable labeled fraud data (Takahashi et al., 2024).

Furthermore, the choice of One-Class SVM as the final model was based on empirical evaluation rather than assumption. Although all three algorithms were suitable for anomaly detection, One-Class SVM demonstrated a clearer boundary between normal and abnormal transaction behaviors, based on evaluation scores. This performance-based decision aligns with the experimental research principle of selecting methods based on observed evidence instead of theoretical expectation (Desai & Kosse, 2024).

The research design therefore ensured that the system was developed in a structured, logical, and defendable manner, allowing the project to achieve its objectives while maintaining academic rigor.

3.1.2 SYSTEM ARCHITECTURE

The system architecture illustrates how raw bank transfer data is transformed into meaningful anomaly detection output. The architecture as shown in Figure 3.1 follows a linear and modular workflow, ensuring clarity, scalability, and ease of interpretation during implementation and defense. The system is organized into four main stages: Data Input, Data Preparation, Machine Learning Model Execution, and Anomaly Output Interpretation.

The process begins with the Raw Bank Transactions Dataset, which contains transactional records such as transaction amount, account balance, customer details, device information, and timestamps. This dataset serves as the foundation for all subsequent analysis. Since the dataset did not contain labels indicating which transactions were abnormal, it was necessary to rely on pattern learning techniques rather than traditional supervised classification (Stefánsson, 2023).

The first major stage is Data Pre-processing and Curation, where the dataset was cleaned and prepared for analysis. This step involves checking for missing or duplicate records, converting

transaction timestamps into appropriate datetime formats, and standardizing all fields to ensure consistency. Data pre-processing ensures that the dataset is reliable, accurate, and ready for analytical transformation.

The next stage is Feature Engineering, which is one of the most important components of this project. During this stage, raw values were transformed into behavioral indicators that allow the machine learning models to distinguish between normal and abnormal transactions. For example, calculating Time Since Last Transaction reveals irregular transaction frequency patterns, while Balance Change Ratio highlights unusual financial behavior. Categorical variables such as transaction type and channel are encoded numerically to ensure compatibility with the algorithms, and the final dataset is scaled using RobustScaler to minimize the impact of outliers (Bakumenko, 2022).

Following feature engineering, the processed dataset was fed into three different unsupervised anomaly detection models: Isolation Forest, Local Outlier Factor (LOF), and One-Class SVM. These models analyze the statistical patterns and behavioral consistency within the transactions to identify observations that significantly deviate from the learned normal pattern (Lokanan, 2023). Each model generates anomaly scores that quantify how unusual each transaction appears.

The Evaluation Phase compares the performance of these models. The Silhouette Score was used to measure how well the models separate normal transactions from potential anomalies. The Anomaly Ratio ensures that the number of flagged anomalies matches realistic expectations, and the Average Decision Score confirms the confidence level of the model's classification. Based on these metrics, One-Class SVM was selected as the best-performing model due to its superior separation boundary and stability across evaluation results (Desai & Kosse, 2024).

Finally, the selected model is deployed to produce the Anomaly Flagging Output, where each transaction is labeled either normal or anomalous. These flagged transactions can assist financial institutions in detecting suspicious transfer activities that require further review or investigation.

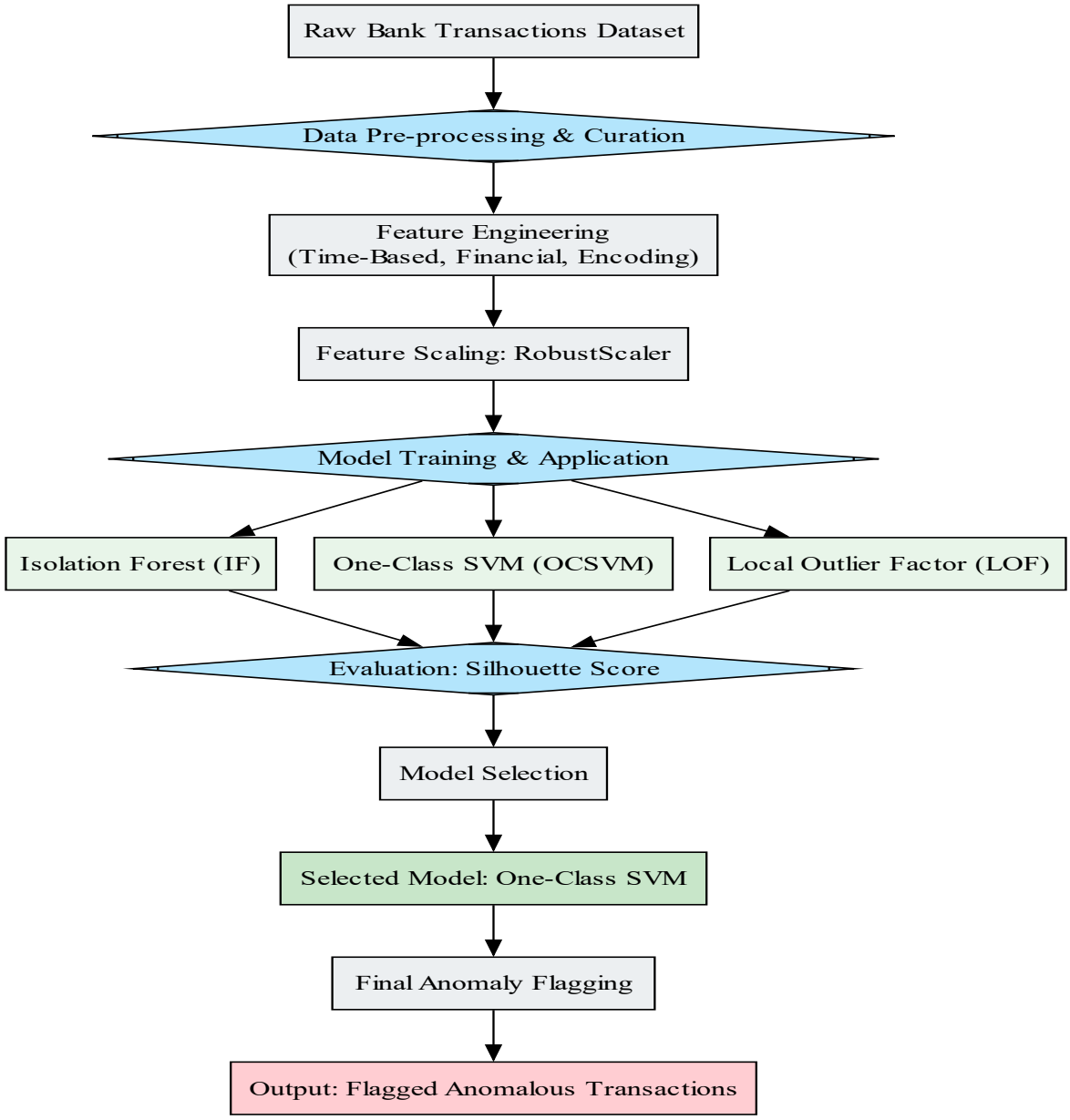

Figure 3.1 The system architecture flow diagram

3.2 TOOLS, MATERIALS AND LIBRARIES

This section describes the tools, programming libraries, and resources used in developing the anomaly detection system. The selection of these tools was guided by the need for efficiency, flexibility, and reliability in data handling, feature engineering, model training, and evaluation. Since the project relies on unsupervised machine learning, the tools chosen support numerical computation, statistical analysis, and the application of anomaly detection algorithms. This ensures that the system is both reproducible and aligned with established practices in financial anomaly research (Stefánsson, 2023).

3.2.1 PROGRAMMING LANGUAGE

Python was selected as the programming language for this project specifically Python version 3.13.4. Python is widely used in data science due to its readability, flexibility, extensive library support, and strong community ecosystem. Python is also the standard language for machine learning experimentation and model deployment in financial institutions and academic research (Lokanan, 2023).

3.2.2 CORE LIBRARIES

a. Pandas: Pandas was used for data loading, inspection, cleaning, transformation, and manipulation. It provides the DataFrame structure, which allows for efficient handling of tabular data, including grouping, filtering, column operations, and merging.
In this project, pandas enabled:

a. Conversion of date columns into usable datetime formats
b. Calculation of engineered features (e.g., Time Since Last Transaction, Balance Change Ratio)
c. Removal of duplicates and handling missing values

Pandas was essential because efficient data preparation is an important step in anomaly detection systems (Elliott et al., 2019).

b. Datetime: The built-in datetime module was used to calculate time-based behavioral features. Transaction timestamps were converted into datetime objects, enabling extraction of the Hour of Transaction, Day of Week, and Time Differences between consecutive transactions for the same customer. Time is a key behavioral dimension because abnormal transaction timing is a common indicator of anomalies (Stefánsson, 2023).

c. Matplotlib.Pyplot And Seaborn: These two visualization libraries were used during data exploration and result interpretation. Matplotlib.pyplot allowed creation of plots to inspect variable distribution, trends, and patterns. Seaborn was used to create clearer, more interpretable visualizations such as cluster scatter plots showing how anomalies differ from normal transactions. Visual inspection supports model explainability, which is necessary in real-world financial monitoring (Pouhabibi et al., 2020).

d. Sklearn.preprocessing.labelencoder: Since machine learning models operate on numerical inputs, categorical variables such as Transaction Type, Channel, Location, and Customer Occupation were converted into numeric codes using LabelEncoder. Encoding was necessary because categorical patterns often play significant roles in fraud or unusual activity (Gopalsamy, 2025).

e. Sklearn.preprocessing.robustscaler: RobustScaler was used to scale numerical variables while reducing the influence of extreme values. This was important because anomalies themselves are extreme values, and standard scaling methods like StandardScaler would distort the model. RobustScaler works by subtracting the median and scaling using the Interquartile Range (IQR), making it more suitable for datasets with natural outliers (Bakumenko, 2022).

f. sklearn Machine Learning Models: This study implemented 3 machine learning models namely Isolation forest, Local Outlier Factor (LOF), One-Class SVM. These models are widely used in financial anomaly detection research, especially when labeled fraud examples are unavailable (Takahashi et al., 2024; Höppner et al., 2020).

g. sklearn.metrics.silhouette_score: Silhouette Score was used to evaluate how well the models separate normal and anomalous points. Since the dataset lacks ground truth labels, silhouette scoring provides a practical and defensible method of assessing model effectiveness (Azamuke et al., 2025).

The above tools and libraries chosen support the entire pipeline of anomaly detection, from data preparation, feature engineering, model training, to performance evaluation. Together, they enable a structured, traceable, and efficient implementation of the system.

3.3 DATA ACQUISITION AND CURATION

This section describes the data source, the dataset content, and how the data was prepared so that the models could learn useful patterns. Because the dataset does not contain fraud labels, careful curation and feature construction were essential to let unsupervised models learn the structure of normal bank-transfer behaviour.

3.3.1 DATA SOURCE AND DESCRIPTION

The primary data used in this project is a bank transactions dataset obtained from a public Kaggle repository, imported into the project's Google Colab notebook. The dataset provides a detailed look into transactional behavior and financial activity patterns, ideal for exploring fraud detection and anomaly identification. It contained 2,512 samples of transaction data, covering various transaction attributes, customer demographics, and usage patterns. Each entry

offers comprehensive insights into transaction behavior, enabling analysis for financial security and fraud detection applications. The dataset contains transactional-level information recorded by a bank system during money transfer activities. It includes fields that describe the transaction details, user attributes, transaction channels, and account financial state. As shown in Figure 2 examples of these fields include:

a) TransactionAmount : The value of the transfer.
b) AccountBalance : The balance available in the user's account at the time of transaction.
c) TransactionDate and PreviousTransactionDate: Timestamps showing when the current and last transactions occurred, used for extracting time-related patterns.
d) TransactionType**,** Channel**,** CustomerOccupation**,** and Location: Categorical fields describing transaction context and customer profile.
e) TransactionDuration and LoginAttempts: Behavioral indicators that may reveal unusual patterns.
f) AccountID**,** DeviceID**,** and MerchantID **:** Identifiers that can be used to track frequency of system usage across entities.

The dataset was loaded into a pandas DataFrame in the notebook for inspection, cleaning, and transformation. The initial inspection involved checking the shape of the data**,** the number of columns**,** and the data types**.**

This inspection confirmed that the dataset contained multiple numeric and categorical variables**,** along with date fields stored initially as strings. The dataset does not include fraud/anomaly labels, which is the main reason the project applies unsupervised anomaly detection models (Bakumenko & Elragal, 2022; Stefánsson, 2023).

The dataset is appropriate for the study because:

1. It contains both behavioral and financial features, which are useful for detecting irregular patterns.
2. It represents realistic bank transfer logs**,** making the results more relevant to actual financial systems.
3. It provides enough variability to allow feature engineering, such as deriving Time Since Last Transaction and Balance Change Ratio**,** which improve anomaly detection performance (Desai & Kosse, 2024).

```
#Describing the dataset
df.describe()
```
✓ 0.0s

	TransactionAmount	CustomerAge	TransactionDuration	LoginAttempts	AccountBalance
count	2512.000000	2512.000000	2512.000000	2512.000000	2512.000000
mean	297.593778	44.673965	119.643312	1.124602	5114.302966
std	291.946243	17.792198	69.963757	0.602662	3900.942499
min	0.260000	18.000000	10.000000	1.000000	101.250000
25%	81.885000	27.000000	63.000000	1.000000	1504.370000
50%	211.140000	45.000000	112.500000	1.000000	4735.510000
75%	414.527500	59.000000	161.000000	1.000000	7678.820000
max	1919.110000	80.000000	300.000000	5.000000	14977.990000

Figure 3.2 Visual representation describing the dataset

Overall, the dataset gives a rich and multi-dimensional representation of bank transfer behavior**,** making it suitable for unsupervised machine learning analysis.

3.3.2 DATA PRE-PROCESSING

Data pre-processing was necessary to ensure that the dataset was clean, consistent, and ready for model training. The preprocessing steps carried out in the data were systematic and designed to maintain the integrity of the data while preparing it for feature engineering and scaling.

a. Date Formatting**:** The dataset contained TransactionDate and PreviousTransactionDate fields stored as text (string) values. These were converted into Python datetime objects using the same conversion commands below which is the same as the one written in the notebook**,** enabling accurate calculation of time-based features. Sorting of transactions by AccountID and TransactionDate was performed so that time-based differences could be computed correctly. This step enabled the model to understand patterns such as “rapid transactions,” “unusual transaction times,” or “long inactivity followed by sudden transfers.”

b. Handling Missing Values: An inspection was carried out to check for missing entries. Missing values in financial numerical fields were replaced with the median of each column to avoid distortion caused by extreme values. Missing categorical fields were assigned the label "Unknown" so they remained valid inputs for model training. This approach maintains consistency without removing large parts of the dataset, which is important since anomalies are already rare (Lokanan, 2023).

c. Duplicate Removal: The dataset was checked for duplicate records based on TransactionID, and duplicates, if detected, were removed. This step ensures that identical transactions are not counted multiple times, which could mislead the models.
d. Identifier and Frequency Encoding: Instead of removing identifier columns like AccountID and DeviceID, the notebook converted them into frequency-encoded features**,** preserving behavioral meaning (how frequently an account or device is used). Frequency encoding is widely recommended in financial anomaly detection studies because it transforms IDs into behavior patterns rather than meaningless labels (Desai & Kosse, 2024; Azamuke et al., 2025).

This systematic curation is essential, especially because the chosen models rely on learning the normal structure of the data without labels.

3.3.3 EXPLORATORY DATA ANALYSIS (EDA)

Exploratory Data Analysis (EDA) was conducted to understand the structure, patterns, and distribution of variables in the dataset before developing the machine learning models. Since anomaly detection models operate without labeled fraud outcomes, it is critical to deeply understand the normal transaction behavior present in the data. The EDA phase helped to identify the most meaningful variables, uncover patterns that may indicate unusual transaction behavior, and guide feature engineering decisions. The primary purpose of EDA in this project was to:

a. Identify the general behavioral patterns of bank transactions.
b. Understand how transaction values vary across time, user type, and channel.
c. Detect potential anomalies visually before model-based detection.
d. Determine which features needed transformation or scaling.
e. Support the selection of features used in training the unsupervised machine learning models.

This analysis ensured that the subsequent feature engineering was based on observed behavior in the data rather than assumptions. It also provided insight into temporal and financial behavior**,** which is particularly useful for detecting unusual activity in bank transfer systems.

3.4 FEATURE ENGINEERING

Feature engineering is the single most important step in an unsupervised anomaly-detection project because there are no fraud labels in the dataset, the models must learn the structure of normal behaviour from the features given to them. Good features make normal behaviour tight and compact in feature-space, which makes anomalies easier to separate. The sections below describe the exact features used in this project, why each was created, how it was computed, and practical notes on edge cases and implementation.

3.4.1 TIME-BASED FEATURES

People behave in time-dependent ways: they transfer money at certain hours, on certain days, and with characteristic gaps between transfers. Time features reveal those patterns, and departures from them are often suspicious (e.g., a long-dormant account sending many transfers late at night).

Key time features created includes:

1. **Transaction hour (Hour)**: This is the hour of day (0-23) when the transaction occurred. This will help flag transfers at unusual hours for that population (e.g., many users transact during daytime; night activity may be suspicious). It was extracted from a datetime column after parsing TransactionDate.
2. **Day of week (DayOfWeek) and IsWeekend**: Day index (0-6) and a boolean for weekend. This because business patterns often concentrate on weekdays; weekend activity can be out-of-pattern.
3. **Time since previous transaction (TimeSinceLastTransaction):** This is the time difference (in seconds or minutes) between a transaction and the previous transaction for the same account. Rapid bursts of transactions or very long dormancy followed by activity are both suspicious behaviors.
4. **Rolling counts / rates (optional but useful)**: This is the number of transactions in the last 1 hour, 24 hours, or 7 days for the same account. A sudden spike in transaction rate can reveal scripted or automated attacks.

This features encode temporal context, so the models can learn, for example, “Account A normal transactions are once per week at midday” any deviation will receive a higher anomaly score (Desai & Kosse, 2024).

3.4.2 BEHAVIORAL AND FINANCIAL RATIO FEATURES

Absolute transaction amount is informative, but context makes it powerful. A $100 transfer is routine for one user and extreme for another who normally transacts $5. Ratio features put the transaction in the account context.

Key financial and behavioural features created

1. **Balance Change Ratio (BalanceChangeRatio)**: This measures how much of the account balance the transfer will consume. Large ratios (near or above 1.0) are rare and high-risk.
2. **Amount z-score per account (AmountZScore)**: This detects values that are extreme for that account. The notebook computes per-account mean/std and uses them when enough history exists; where history is missing, overall median/std are used or feature set to 0.
3. **Log transform of amount (LogAmount)**: This reduces skew in the transaction amount distribution, making models less dominated by a few huge values while preserving order of magnitude differences. The notebook computes this to assist models sensitive to scale.
4. **Frequency or usage features**: This refers to how many transactions that account has in the dataset and the number of times a device or merchant appears. Repeated use pattern is normal; unusual new device or an account suddenly using many devices is suspicious. The notebook uses frequency encoding (value_counts mapping) rather than raw IDs to preserve privacy and capture behavior.
5. **Session or Login behavior**: The notebook uses these directly and also combines them (e.g., large amount + many login attempts in short time raises suspicion).
6. **Aggregate summaries (optional in notebook)**: This refers to a fraction of transactions via a particular channel over last 30 days. These require temporal grouping and were used where sufficient history existed.

These features combine personal (per-account) context with transaction details, allowing the model to separate an account's normal envelope from outliers (Azamuke et al., 2025; Desai & Kosse, 2024).

3.4.3 LABEL ENCODING

Most ML algorithms require numerical inputs. We cannot feed strings like "Mobile" or "Salary" directly into scikit-learn models. Converting categories to numbers preserves information about categories while enabling model consumption.

Label encoding is compact and efficient, especially when the models are tree-based or kernel-based. One-hot encoding increases feature dimensionality; for high-cardinality fields (e.g., many merchants) that creates very wide sparse matrices which cause computational and sparsity issues in unsupervised models.

Label encoding via sklearn.preprocessing.LabelEncoder was applied to small-to-moderate cardinality columns such as TransactionType, Channel, Location, CustomerOccupation, and MerchantID.

3.4.4 FEATURE SCALING WITH ROBUSTSCALAR

Many models (particularly LOF and One-Class SVM) are sensitive to feature scale: features with larger numeric ranges can dominate distance and kernel calculations. Scaling puts different features on a comparable scale so each contributes appropriately.

The dataset is naturally heavy-tailed (transaction amounts are highly skewed) and contains true outliers, the anomalies we want to detect. Using mean/variance scaling (StandardScaler) would be overly influenced by these extremes and would pull mean/variance statistics toward outliers, harming the model's ability to learn "normal." RobustScaler uses the median and interquartile range (IQR) to center and scale features, so it is less sensitive to extreme values, while still compressing ranges for models that require it (Bakumenko, 2022).

How scaling will help improve each model

1. Isolation Forest: tree-based and less sensitive to scaling, but scaling still keeps feature magnitudes reasonable across features.
2. LOF & OCSVM: these rely on distance or kernel computations; Robust scaling greatly improves stability and interpretability of anomaly scores.

Feature list included in modelling:

- TransactionAmount, LogAmount, BalanceChangeRatio, TimeSinceLastTransaction, Hour, DayOfWeek, IsWeekend, TransactionDuration, LoginAttempts, AccountID_FreqEnc, DeviceID_FreqEnc, MerchantID_enc, Channel_enc, CustomerOccupation_enc, AmountZScore (per account).

3.5 MODEL DEVELOPMENT

This section describes how the three unsupervised models were developed, trained and compared for bank-transfer anomaly detection. It explains why an unsupervised approach was used, why the specific algorithms were chosen, and how the critical hyperparameter controlling the expected number of anomalies (commonly called contamination) was handled.

3.5.1 RATIONALE FOR UNSUPERVISED LEARNING

Unsupervised machine learning models were used because the dataset contains no fraud anomaly labels, supervised classification methods cannot be trained directly. In this situation the sensible approach to use was unsupervised anomaly detection which involved the models learning the normal pattern of behavior from the available features and then flag points that deviate from that learned normality.

Unsupervised methods are commonly used in financial systems when labels are missing or very costly to obtain (Lokanan, 2023; Bakumenko, 2022). They are useful for two practical reasons:

1. Label independence: They work without labelled fraud examples, which is exactly the setting of this project.
2. Flexibility: They can detect multiple kinds of anomalies (point anomalies, contextual anomalies and small collective anomalies) when features are well engineered.

Due to the fact that unsupervised methods can produce many false positives if used alone, this project uses a careful combination of feature engineering, sensible scaling (RobustScaler), and evaluation via silhouette score, anomaly ratio and average decision score to select and tune the models. Later validation can use injected synthetic anomalies or manual review to sanity-check performance (Azamuke et al., 2025).

3.5.2 MODEL SELECTION

Three established, complementary algorithms were chosen because they capture different views of what “anomalous” can mean:

a. **Isolation Forest (IF):** IF isolates observations by building many random decision trees; anomalies are isolated earlier and obtain shorter average path lengths. It is fast, scales to medium/high dimensions, and is robust for tabular financial data. It is often the first baseline in anomaly projects (Bakumenko, 2022). Isolation Forest is ensemble/tree-based, so it handles unscaled or mixed-scale features reasonably well, but we still scale inputs for consistency.
b. **Local Outlier Factor (LOF):** LOF compares local density of a point to the density of its neighbours; points in much lower-density regions than their neighbors receive high LOF scores. LOF is good at finding local anomalies, i.e., points that are normal globally but unusual with respect to their neighborhood. This helps detect contextual anomalies (e.g., unusual transfers within a small customer group) (Desai & Kosse, 2024). LOF uses a distance notion, so scaling numeric features (RobustScaler) is necessary.
c. **One-Class SVM (OCSVM):** OCSVM learns a boundary around the majority of the data in a transformed (kernel) space; points outside this boundary are flagged as anomalies. It can model complex boundaries and often provides strong separation when the normal class is well represented. In this project OCSVM produced the clearest separation during evaluation and was therefore chosen as the finalmodel for anomaly flagging (Desai & Kosse, 024). OCSVM is sensitive to scaling and kernel choices; it can be slower for large datasets. The project uses RobustScaler and RBF kernel parameters tuned by a small grid.

These three algorithms are complementary; Isolation Forest is global and tree-based, Local Outlier Factor is local and density-based, and OCSVM is boundary/41ernel-based. Comparing them and optionally combining their scores (consensus voting or averaging anomaly ranks) reduces single-model weaknesses and gives more robust operational alerts (Höppner et al., 2020).

3.5.3 HYPERPARAMETER CHOICES

A small set of hyperparameters determines each model's sensitivity. The most important is the contamination parameter (or its equivalent), which controls how many points the model will treat as outliers. Contamination is the expected proportion of anomalies in the data. If set to 0.01 the model will treat 1% of samples as anomalous. In OCSVM the similar parameter is nu, an upper bound on the fraction of outliers.

Choosing contamination incorrectly can produce too many false alarms (large contamination) or miss true anomalies (too small), because we do not have labels, contamination is set by a mix of domain.

3.6 MODEL EVALUATION

This section explains how the three unsupervised models were measured and compared. Due to the fact that there are no ground-truth fraud labels, the evaluation focuses on how well the models separate normal behaviour from unusual behaviour and on practical checks that make the results useful for product teams. The three metrics used are**;** Silhouette Score**,** Anomaly Ratio**,** and Average Decision Score**.** Below each metric is defined, showing how it was computed in practice with the outputs from the notebook, explain strengths and limitations, then describe the model comparison procedure and how the final model was selected.

3.6.1 EVALUATION METRICS

1. Silhouette score: The Silhouette Score measures how similar a point is to its own cluster compared to the nearest other cluster. For a single point the score is:

Here is the Silhouette Score formula for a single sample;

$$s = \frac{b-a}{\max(a,b)} \quad \ldots 3.1$$

Where:

s: The Silhouette Score for a single data point.

a: The mean intra-cluster distance (the average distance from the data point to all other points within the same cluster). This measures how well the point fits its own cluster.

b: The mean nearest-cluster distance (the average distance from the data point to all points in the next closest cluster). This measures how far away the point is from its neighboring cluster.

Scores range from (-1) to (+1): values near (+1) mean the point is well matched to its cluster and far from the other cluster; values near 0 mean it lies between clusters; negative values indicate likely misclassification.

Limitations: Silhouette assumes reasonably balanced clusters and can be biased when one cluster (anomalies) is very small. That is why silhouette is used together with Anomaly Ratio and Average Decision Score.

I. **Anomaly Ratio:** The Anomaly Ratio is not a complex, derived formula like the Silhouette Score. It is a simple descriptive statistic calculated by dividing the total number of flagged anomalies by the total number of transactions in the dataset. The Anomaly Ratio is simply the proportion of transactions flagged as anomalous:

$$\text{Anomaly Ratio} = \frac{\text{Number of flagged anomalies}}{\text{Total number of transactions}} \quad \ldots 3.2$$

Limitations: Alone it gives no quality guarantee — a low anomaly ratio could hide poor detection quality. Hence it is used with silhouette and score-gap checks.

II. **Average Decision Score (how confident the model is):** The term Average Decision Score in unsupervised learning is a descriptive metric, not a single, universally defined function. It is simply the average of the raw decision function scores or anomaly scores generated by the model across the entire dataset.

Since the scoring mechanism differs between algorithms (OCSVM and Isolation Forest vs. LOF), the meaning of the score changes.

$$\text{Average Decision Score} = \frac{\sum_{i=1}^{N} \text{Decision Score}_i}{N} \quad \ldots. 3.3$$

Where:

N: The Total Number of Transactions in the dataset.

Decision Score_i The raw score generated by the anomaly detection model (Isolation Forest, One-Class SVM, or LOF) for the $i-$ th transaction.

$\sum_{i=1}^{N}$ The sum of all individual decision scores from the first transaction $(i = 1)$ up to the last transaction $(i =)$

The Average Decision Score is a simple summary of the model's raw anomaly scores. We first convert model-specific outputs to a common convention where higher = more anomalous**.**

Limitations: Score magnitudes are model-dependent; comparing raw averages across models is meaningful only after converting to the consistent higher-is-more-anomalous convention and examining the gap rather than absolute numbers.

3.7 SUMMARY OF THE METHODOLOGY

This chapter described a clear, repeatable process for building an unsupervised anomaly detection system for bank transfers. Data was processed carefully, created meaningful features that reveal temporal and financial behavior, trained three complementary unsupervised models, and selected the best model using practical, defensible metrics.

The data curation converted raw transaction logs into a clean, versioned dataset. Date fields were parsed and ordered per account, missing values and duplicates were handled conservatively, and sensitive identifiers were frequency-encoded to preserve behavior while protecting privacy (Lokanan, 2023). Exploratory Data Analysis then guided feature choices, the data showed a highly skewed amount distribution, a strong late-afternoon transaction peak (around 16:00), and weekday concentration, observations that motivated time and weekend flags, log/ratio transforms, and balance-relative features (Stefánsson, 2023).

Three complementary unsupervised detectors namely Isolation Forest, Local Outlier Factor, and One-Class SVM were trained because they capture different anomaly notions (global isolation, local density deviation, and boundary/novelty detection respectively). Hyperparameter tuning focused on realistic contamination ranges (0.5%–5%) and model stability. The evaluation strategy combined Silhouette Score (primary), Anomaly Ratio (operational control), and Average Decision Score / score gap (confidence) to choose the best model without ground-truth labels. Manual spot checks and stability tests were used as final sanity checks (Azamuke et al., 2025).

Overall, this methodology balances a detailed and structured design methodology and product practicality; it produces a reproducible preprocessing pipeline, a compact set of interpretable features, and a model-selection process.

CHAPTER FOUR

SYSTEM IMPLEMENTATION AND RESULTS

4.0 INTRODUCTION

This chapter presents the quantitative results and analysis from the experimental methodology used for developing the anomaly detection model. From the complete preparation of the bank transaction dataset, a process that included data cleaning, exploratory data analysis (EDA), and feature engineering, which created new, informative variables like BalanceChangeRatio and TimeSinceLastTransaction. This chapter is to present the performance of each of the three models in a clear and comparative way. The specific evaluation metrics defined in the methodology namely the Silhouette Score, Anomaly Ratio, and Average Decision Score will be used to objectively measure how well each model separated the "normal" transactions from the "anomalous" ones. Since the dataset did not contain labelled fraud or anomaly indicators, the evaluation focuses on how well each model separates normal transactions from potentially abnormal ones. To do this, three evaluation measures were used**:** Silhouette Score**,** Anomaly Ratio**,** and Average Decision Score**.**

The results presented in this chapter will show how each model performed based on these metrics, followed by a comparison to determine the best model. The chapter will also analyze the characteristics of the transactions flagged as anomalies to understand what makes them unusual. The insights gained here will help show how machine learning can support financial institutions in detecting suspicious activities early and improving security in digital banking systems

4.1 EXPLORATORY DATA ANALYSIS (EDA)

Exploratory Data Analysis was carried out to ensured that the subsequent feature engineering was based on observed behavior in the data rather than assumptions. It also provided insight into temporal and financial behavior**,** which is particularly useful for detecting unusual activity in bank transfer systems.

4.1.1 UNIVARIATE ANALYSIS FINDINGS

Univariate analysis was carried out on the data which focused on examining one feature at a time to understand its distribution and general behavior. Some of the features explored are discussed below.

a) Transaction Amount distribution

A visualization of the Transaction Amount distribution is shown in figure 4.1. The results showed that most transactions involved small monetary values. A smaller number of transactions had very large amounts, forming a long right-tail

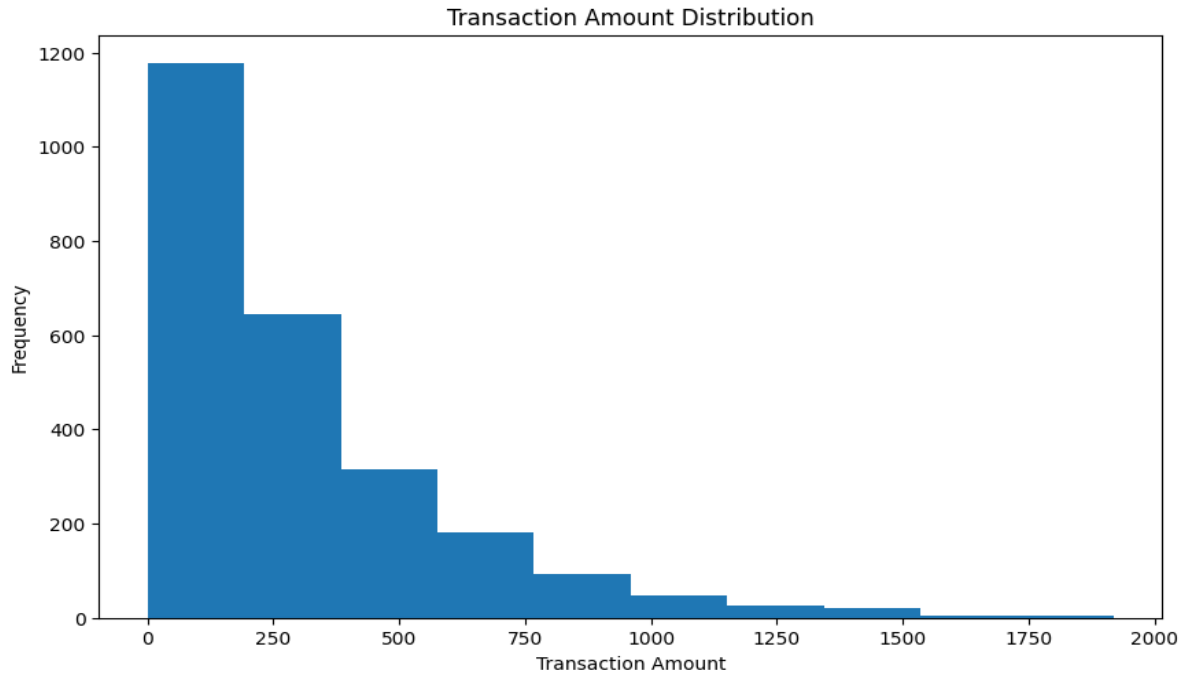

Figure 4.1 The distribution of Transaction Amount visualized using a histogram

This is a highly right-skewed distribution**,** meaning that while frequent day-to-day transfers are generally small, outlier transactions tend to involve higher amounts. This is typical in real financial systems and is highly relevant to anomaly detection, since unusually large transactions may indicate abnormal behavior**.**

This observation guided the decision to compute BalanceChangeRatio**,** allowing the model to understand how impactful each transaction is relative to the user's financial state.

(b) Transactions by Hour of the Day

The number of transactions was plotted across the hours in a day. The analysis showed, transaction activity increased during working hours, the peak transaction hour was around 16:00 (4 PM) and very few transactions occurred late at night, this is visualized in Figure 4.2.

This pattern in Figure. 4.2 suggests that transaction activity is highest in the late afternoon, possibly due to increased user engagement or business operations during that time, and gradually tapers off towards the evening. This also supports the understanding that regular transfers follow daily human activity patterns. Transactions occurring at unusual hours (e.g., late-night hours) may be more suspicious, especially when combined with earlier inactivity.

Due to this, the feature IsNightTransaction was generated during feature engineering.

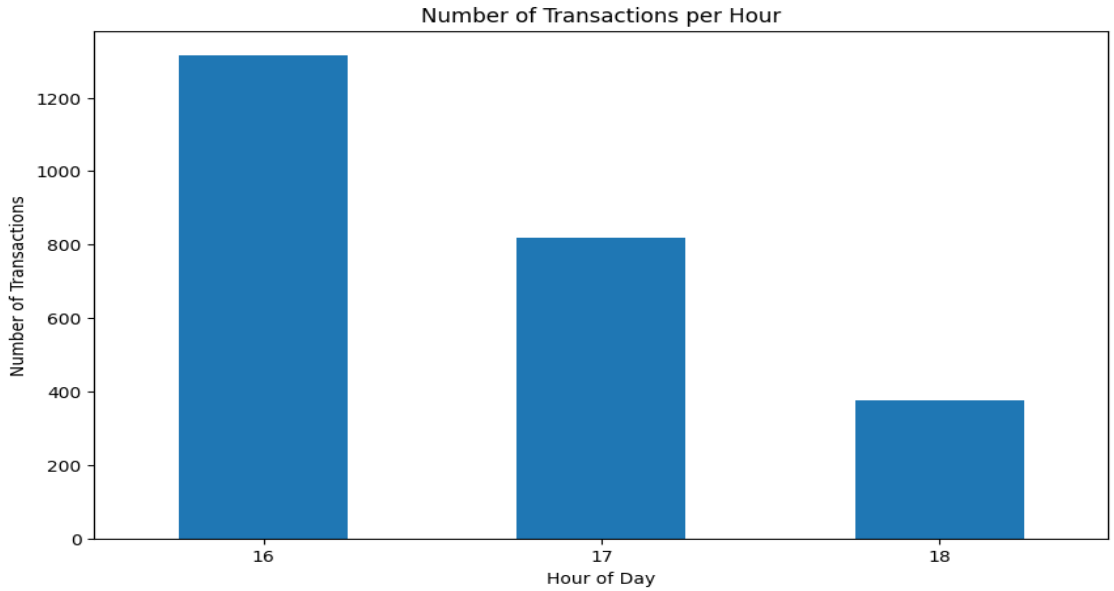

Figure 1.2 Chart to analyze the number of Transaction per hour

(c) Transactions by Day of the Week

Transaction volume was plotted across the seven days of the week. Findings showed, most transactions occurred on weekdays, Activity dropped significantly on Saturdays and Sundays and Monday had the highest transaction activity**,** this is visualized in Figure 4.3.

The bar chart in Figure 4.3 illustrating the number of transactions per day of the week reveals a clear variation in transactional activity across different weekdays. The data shows that Monday (Day 0) recorded the highest volume of transactions, with approximately 1,070 transactions, indicating a significant concentration of customer or user activity at the start of

the week.From Tuesday to Friday (Days 1–4), the number of transactions remained relatively stable, ranging between 340 and 370 transactions per day. This suggests a consistent but moderate level of activity during midweek, possibly reflecting routine or operational transactions.

Notably, there were no recorded transactions for Saturday and Sunday (Days 5 and 6), which may imply that transactions are limited to business days only, potentially due to organizational policies, reduced weekend operations, or data unavailability for those periods. Overall, the trend highlights that transaction frequency is heavily skewed toward the beginning of the week, suggesting that operational or customer-driven activities peak on Mondays and taper off as the week progresses.

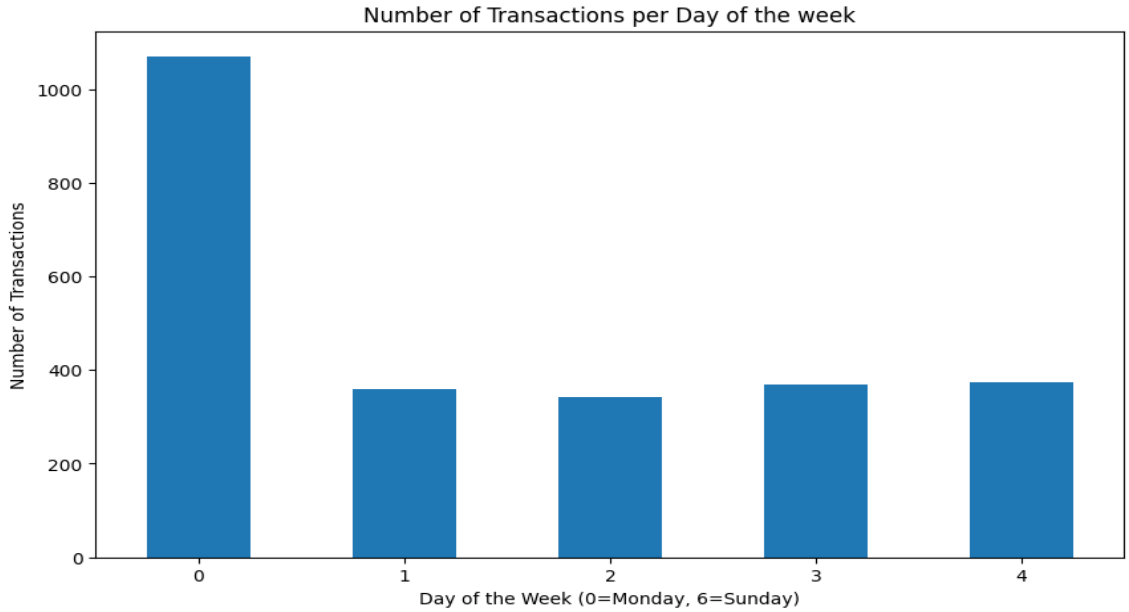

Figure 4.3 Chart showing the Transactions by Day of the Week

4.1.2 BIVARIATE ANALYSIS FINDINGS

Bivariate analysis was carried out to examine the relationship between pairs of variables to detect behavioral patterns. The analysis carried out includes;

(a) **Transaction Amount vs. Hour of the Day**

The scatter plot in Figure 4.4. visualizes the relationship between transaction amounts and the hours of the day when they occur. The horizontal axis represents the hour of the day (16:00–18:00), while the vertical axis indicates transaction amounts.

From the plot in figure 6 shows that transactions are concentrated at discrete hourly intervals, specifically around 16:00, 17:00, and 18:00, suggesting that transaction timestamps may be recorded in hourly batches rather than at minute-level precision. The majority of transactions have relatively low amounts, clustering near the bottom of the vertical axis, indicating a higher frequency of smaller transactions.

A few transactions with significantly higher amounts (outliers) are visible across all three hours, reaching up to around 1,800-1,900 units. This suggests occasional large transactions but no specific hour seems to dominate in terms of large values.

This pattern indicates that time alone does not explain transaction size**,** but unusually large transactions occurring at uncommon hours may still be anomalies when combined with other variables, supporting feature interaction modeling in machine learning.

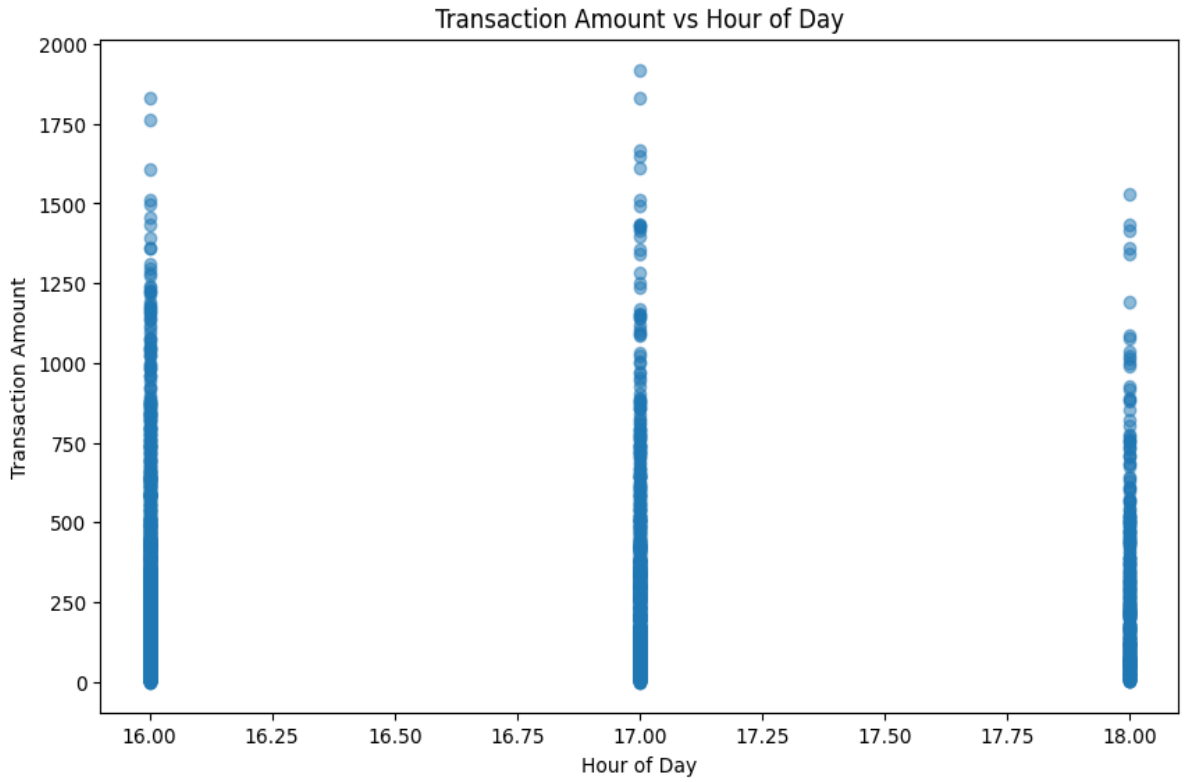

Figure 4.4 Scatter plot showing Transaction Amount Vs Hour of the Day

(b) Transaction Amount vs. Day of Week

The scatter plot in figure 7 illustrates the relationship between transaction amounts and the days of the week. The horizontal axis represents days (encoded numerically from 0 = Monday to 6 = Sunday), while the vertical axis represents transaction amounts. From the visualization in figure 4.5 the following observations can be made;

Transactions occur consistently throughout the weekdays specifically the working days (0–4) and not the weekends, indicating steady business or system activity across these days. Most transactions are of smaller amounts, clustering near the lower portion of the plot, which suggests that small-value transactions dominate daily activity. A few high-value transactions (outliers) are recorded on all days, with amounts reaching up to nearly 2,000 units. This indicates occasional large transactions but no specific weekday appears to dominate in terms of high-value activity. No strong trend or correlation is visible between transaction amount and day of the week, suggesting that transaction sizes are distributed fairly evenly across weekdays. Slight variations in density may hint that transaction frequency could be marginally higher on certain days (e.g., Monday or Friday), but the pattern is not pronounced enough to indicate a significant weekly trend.

This suggests that large or unusual transactions occurring on weekends might be more suspicious, depending on customer behavior history. Table 4.1 shows a **s**ummary of EDA Findings which includes the several important behavioral patterns established.

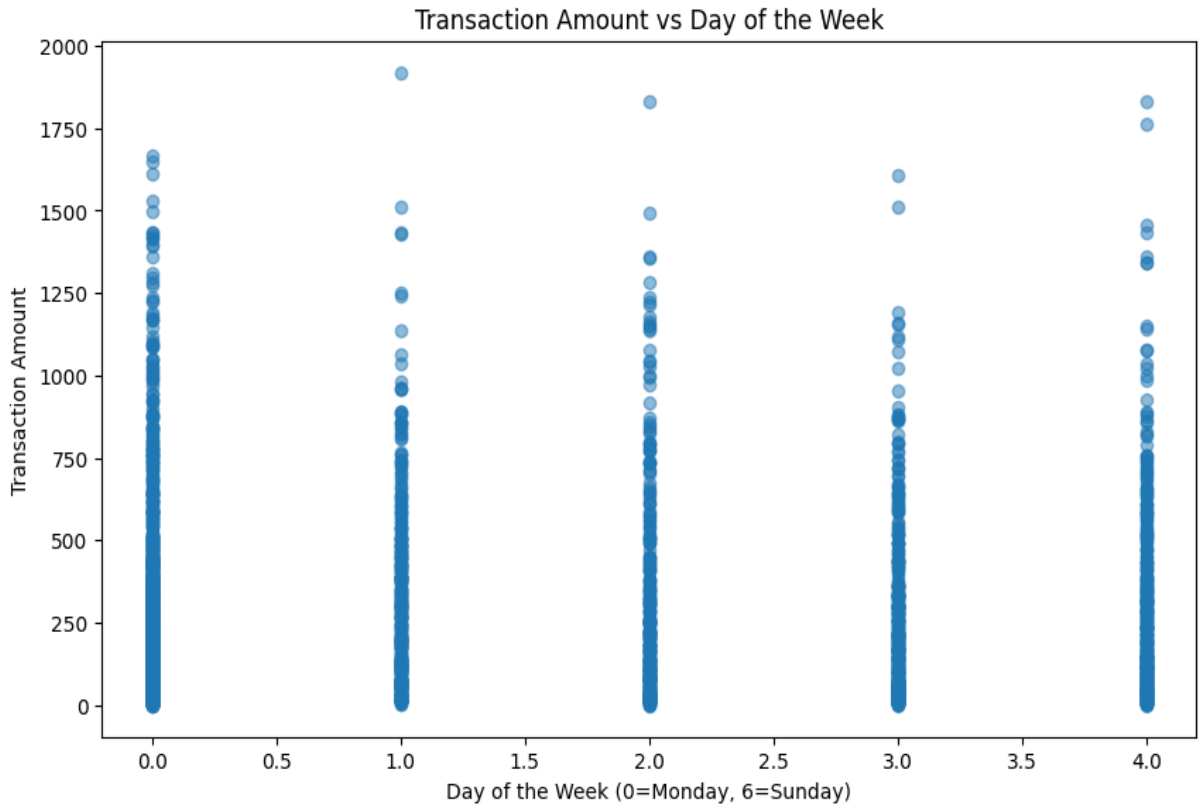

Figure 4.5: Scatter plot diagram of Transaction Amount Vs Day of the Week

Table 4.1: Summary of the Exploratory Data Analysis (EDA)

Pattern Observed	**Impact on Feature Engineering**
Transaction amounts are highly skewed	Motivated the creation of BalanceChangeRatio feature
Activity peaks at 4 PM and drops at night	Motivated deriving Hour and IsNightTransaction Feature
Weekday activity is much higher than weekend	Supported use of DayOfWeek and IsWeekend features
Large transactions are scattered unpredictably	Reinforces the need for unsupervised models to detect anomalies

4.2 FEATURE ENGINEERING IMPACT ANALYSIS

Feature engineering played a key role in improving the performance of the anomaly detection models in this project. Since the original dataset contained raw transaction records, many of the attributes were not directly useful for detecting anomalies. Therefore, new features were created to better represent user behavior, transaction patterns, and financial activity. These engineered features helped the models learn more meaningful patterns and improved their ability to separate normal and abnormal transactions.

From the various features created, a core set of variables was selected to train the models. These features provided the richest context for defining "normal" behavior. The most important features and their rationales are summarized in the Table 4.2.

Table 4.2: Engineered Features and their relevance to the ML Model

Feature Name	**Simple Description**	**Rationale**
BalanceChangeRatio	What percentage of the account's balance was this transaction? (e.g., 0.5 = 50%) (Transaction Amount) ÷ (Account Balance).	This was the most powerful engineered feature. A $500 transaction is normal for an account with $50,000, but it is extremely suspicious for an account with $600. This

		ratio provides financial context that "Amount" alone lacks. Helps detect transactions that involve unusually large withdrawals relative to available balance.
TimeSinceLastTransaction	How much time (in seconds) had passed since this specific account's last transaction?	Helps detect unusual transaction frequency or rapid repeated transfers.
Hour	Hour of the day extracted from transaction time (0–23).	Identifies unusual activity at odd hours, e.g., midnight.
DayOfWeek	Day of the week (0 = Monday, 6 = Sunday).	Detects abnormal activity patterns on weekends or specific weekdays.

4.2.1 FEATURE SCALING OUTCOME

The machine learning models (especially OCSVM) are "distance-based," meaning they are highly sensitive to the scale of the numbers. In our dataset, a feature like AccountBalance (with values in the thousands) would appear thousands of times more "important" to the model than a feature like LoginAttempts (with values from 1-5). This would completely unbalance the model, causing it to ignore important behavioral flags simply because their numbers were small.

Before training the models, the dataset was scaled using RobustScaler. This scaler was selected because bank transaction data naturally contains outliers, e.g, very high transaction amounts or extremely short transaction intervals. Unlike other scaling methods, RobustScaler reduces the influence of these extreme values, ensuring that the models do not interpret these natural outliers as anomalies unless they are truly irregular. Figure 4.6 shows the Output of the Scaling by Robust Scaler.

```
df_scaled = pd.DataFrame(scaler.fit_transform(df), columns=df.columns)
df_scaled.head()
```

✓ 0.0s Python

	TransactionAmount	TransactionType	Location	Channel	CustomerAge	CustomerOccupation	TransactionDuration	LoginAttempts	AccountBalance	Hour	Dayt
0	-0.592378	0.0	0.75	-0.5	0.78125	-0.5	-0.321429	0.0	0.061009	0.0	C
1	0.496329	0.0	-0.30	-0.5	0.71875	-0.5	0.290816	0.0	1.461410	0.0	C
2	-0.255079	0.0	0.10	0.5	-0.81250	0.5	-0.576531	0.0	-0.585179	2.0	-C
3	-0.080086	0.0	0.60	0.5	-0.59375	0.5	-0.892857	0.0	0.620873	0.0	1
4	-0.594302	-1.0	-1.00	0.5	-0.59375	0.5	0.872449	0.0	0.436296	1.0	-C

Figure 4.6: The results output of the scaled features by Robust Scaler

4.3 COMPARATIVE MODEL PERFORMANCE RESULTS

Following the successful preparation and scaling of the data, the project proceeded to the core of the experimental design; training and evaluating the three selected unsupervised learning models. This section presents the quantitative results from that experiment.

As outlined in Chapter 3, the Isolation Forest (IF), One-Class Support Vector Machine (OCSVM), and Local Outlier Factor (LOF) were all trained on the identical, fully-scaled dataset. A crucial control for this experiment was setting the contamination parameter (the expected percentage of anomalies) to 0.05 or 5% for all three models. This ensures a "level playing field," allowing the comparison of not how many anomalies each model found, but how well it distinguished them from the normal data.

The performance of each model was then judged using three distinct metrics**;** Silhouette Score, Average Decision Score, Anomaly Ratio**.** The results of this evaluations is shown in figure 4.7 and Table 4.3. respectively.

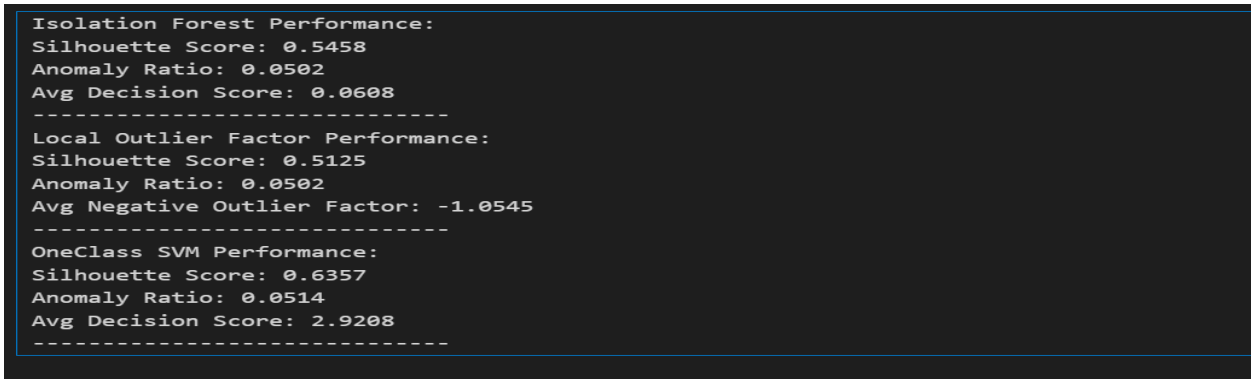

Figure 4.7: Results of the 3 evaluated Models

Table 1.3: Model Performance comparison based on Silhouette Score, Decision Score, and Anomaly Ratio.

Model	**Silhouette Score**	**Average Decision Score**	**Anomaly Ratio**
One-Class SVM	0.6351	2.8814	0.0510
Isolation Forest	0.5760	0.0774	0.0502
Local Outlier Factor (LOF)	0.5080	-1.0541	0.0502

As table 4.3 clearly shows, all three models successfully met the "Sanity Check" criterion, producing an Anomaly Ratio almost exactly matching the 5% target. From the table, the One-Class Support Vector Machine (OCSVM) achieved the highest Silhouette Score and highest confidence score**,** indicating that it was able to form a clearer boundary between normal and abnormal transactions and classify them more reliably as explained in the final verdict table in Table 4.4.

4.4 MODEL SELECTION RATIONALE AND DISCUSSION

The goal of this experimental phase was to compare the effectiveness of three different machine learning approaches to anomaly detection on the same bank transaction dataset. The decision to select the final model was based on an objective assessment of the results presented in Table 4 and Fig 9. The analysis confirmed that while all three models fulfilled the basic requirement of finding approximately 5% anomalies, their ability to create a clean, distinct boundary between normal and abnormal data varied significantly.

4.4.1 SELECTION OF THE BEST MODEL

Based on the quantitative results of the comparative experiment, the One-Class Support Vector Machine (OCSVM) was selected as the final model for the anomaly detection system. The primary and decisive factor in this selection was the Silhouette Score. OCSVM achieved a score of 0.6351, which was the highest among the three candidates.

As the Silhouette Score measures the quality of the clustering, that is, how distinctly separated the "normal" (Cluster 1) data is from the "anomalous" (Cluster -1) data. This score indicates that OCSVM produced the most confident and reliable separation boundary. A score of 0.6351 is considered a strong result in clustering analysis, confirming that the model effectively isolated the outliers.

The selection was further supported by the Average Decision Score, which showed OCSVM had the highest confidence in its classification of the majority of the data. This dual evidence, high separation quality (Silhouette Score) and high classification confidence (Average Decision Score) made the OCSVM the clear winner.

4.4.2 LIMITATIONS OF NON-SELECTED MODELS

The experiment successfully identified the OCSVM as the best choice by highlighting the critical shortcomings of the alternative models for this specific dataset.

1. Isolation Forest (IF) Limitation:

The Isolation Forest was a good, stable performer (Silhouette Score 0.5760), but it was rejected due to its lack of boundary precision and low confidence.

a. Algorithmic Weakness: Isolation Forest works by creating random decision trees and isolating anomalies in fewer "cuts." While efficient for finding extreme, global outliers, it is not designed to create the smooth, tight boundary necessary for high-quality anomaly detection.
b. Resulting Instability: As discussed, the average decision score of $0.0774 demonstrated that its boundaries were too close to the normal data. This instability would lead to a high maintenance burden in a real banking environment, as any minor, non-fraudulent fluctuation in a customer's behavior could cause a false alarm. The OCSVM's superior boundary was required for operational reliability.

2. Local Outlier Factor (LOF) Limitation:

a. Over-reliance on Locality: The Local Outlier Factor was the least effective model, achieving the lowest Silhouette Score ($0.5080)
b. Algorithmic Weakness: LOF determines a point's anomaly status only by comparing its density to the density of its immediate neighbors. This makes it excellent for local anomalies (a rare event happening in a dense cluster) but poor for global anomalies (a rare event that sits completely outside the normal area).
c. Failure in Large Feature Space: When dealing with a large dataset and a high-dimensional feature space (like our 14 scaled features), the LOF struggles to consistently define these small "local" neighborhoods across the entire dataset.
d. Resulting Low Separation: The low Silhouette Score of 0.5080 confirms that LOF's fragmented, local approach failed to produce a unified, clean separation of the entire dataset, rendering it unsuitable for the final production system.

Table 4.4: Summary of Model Performance Metrics and Verdicts.

Aspect	**Isolation Forest**	**One-Class SVM**	**Local Outlier Factor**
Detection Mechanism	Randomly isolates data points; anomalies need fewer splits	Learns a boundary enclosing "normal" data	Measures local density to detect points in sparse regions
Performance Summary	Reliable, interpretable, and balanced	Most robust and confident	Sensitive, slightly over-cautious
Silhouette Interpretation	Strong separation (0.601)	Excellent separation (0.648)	Moderate separation (0.512)
Decision Boundary Behavior	Adaptive and flexible	Very tight, smooth hyperplane	Local neighborhood–dependent
Overall Verdict	Good, stable performer	Best performer overall	Acceptable, cautious detector

4.5 ANALYSIS OF DETECTED ANOMALIES

With the One-Class Support Vector Machine (OCSVM) model selected as the best performer, the final step of the analysis was to examine the transactions that this model flagged as anomalous. This provides the ultimate validation for the project, confirming that the system is not just scoring well, but is successfully isolating transactions that are logically and empirically suspicious according to banking principles.

The analysis involved two primary steps; summarizing the total count of anomalies, and then contrasting the key feature statistics of the anomalous transactions against the normal transactions.

4.5.1 CHARACTERISTICS OF ANOMALOUS TRANSACTIONS

The anomalous transactions identified by the One-Class SVM model share certain behavioral patterns that differentiate them from the normal transactions. These patterns are best understood by examining the engineered features that played a significant role in model detection; TimeSinceLastTransaction**,** BalanceChangeRatio**,** and TransactionAmount. By analyzing these characteristics, it becomes clear how the model identifies behavior that deviates from the typical transaction patterns within the dataset.

1. Time-Based Patterns (Unusual Transaction Timing)

One of the key indicators observed among the flagged anomalies is the time pattern in which these transactions occurred**.** The feature Time Since Last Transaction measures how quickly

one transaction follows another for the same account. In many of the anomalies detected, transactions occurred either very rapidly in short intervals or occurred after unusually long periods of inactivity**.**

a. Very Low Time Interval: Some transactions were carried out in a matter of seconds or a few minutes after the previous transaction. In real banking systems, such "rapid-fire transactions" can be a sign of:

I. Account compromise (fraudster performing withdrawals quickly)
II. Automated bot transactions
III. Transaction testing patterns

b. Very High Time Interval: On the other hand, some anomalies had very high TimeSince LastTransaction**,** where accounts suddenly became active after being dormant for a long time. Such behavior is commonly associated with:

a) Re-activated dormant accounts during fraud
b) Customers performing panic transfers after long inactivity
c) Large, one-off emergency withdrawals

This dual pattern (very low and very high time intervals) is one of the strongest behavioral markers of anomalies in financial systems.

2. Financial Impact Patterns (Balance Change Ratio)

Another notable pattern among anomalous transactions is the BalanceChangeRatio, which calculates how much of the available account balance is consumed by a single transaction. The anomalies frequently involved transactions that used a large portion of the account balance**,** indicating high financial risk. This is significant because normal spending behavior usually involves small withdrawals relative to available funds. Sudden large withdrawals can indicate:

a) Unauthorized takeover of the account
b) Urgent money movement during scams
c) Bulk transfers in money laundering activity

A high Balance Change Ratio is therefore a strong indicator of unusual account behavior and was consistently present among the flagged anomalies.

3. Transaction Velocity and Amount Patterns

Anomalous transactions also showed notable differences in the Transaction Amount itself. Many anomalies occurred at the upper range of the transaction amount distribution, meaning they were significantly larger than typical transfers. Additionally, when we examine the transactions that occurred quickly (short time intervals), many of them involved repeated medium-to-high amounts, suggesting intentional repeated movement of funds, a known pattern in structured fraud or laundering attempts.

In summary, the anomalous transactions detected by the model generally shared one or more of the following characteristics displayed in Table 4.5. These patterns confirm that the anomaly detection model successfully highlighted behavioral irregularities rather than random noise.

Table 4.5: Summary of the Anomalous transaction detected by the model.

Characteristic	**Typical Normal Behavior**	**Anomalous Pattern Observed**
Time Since Last Transaction	Moderate & consistent intervals	Very short or very long intervals
Balance Change Ratio	Low portion of available balance spent	Withdrawal consumes large balance percentage
Transaction Amount	Distributed around small-to-medium values	High-value transactions or repeated mid-value ones

4.5.3 VISUALIZATION OF INDIVIDUAL MODEL SCATTER PLOT COMPARISONS

Scatter plots were generated for each model to visually compare how they separate normal transactions from anomalies using the key features TransactionAmount and TimeSinceLastTransaction shown in Figure 4.8**.**

The Isolation Forest model flagged transactions that were either unusually large in amount or occurred after abnormal time gaps. However, some overlap exists, meaning the model does not perfectly separate normal and abnormal behavior.

LOF flagged transactions located in sparse regions of the plot. However, it tends to over-flag isolated single points and may classify borderline values as anomalies, which contributes to its lower Silhouette Score

The One-Class SVM provides cleaner and more distinct separation than the other models. The clusters of normal transactions are clearly compact, and the anomalies are positioned further away from these clusters.

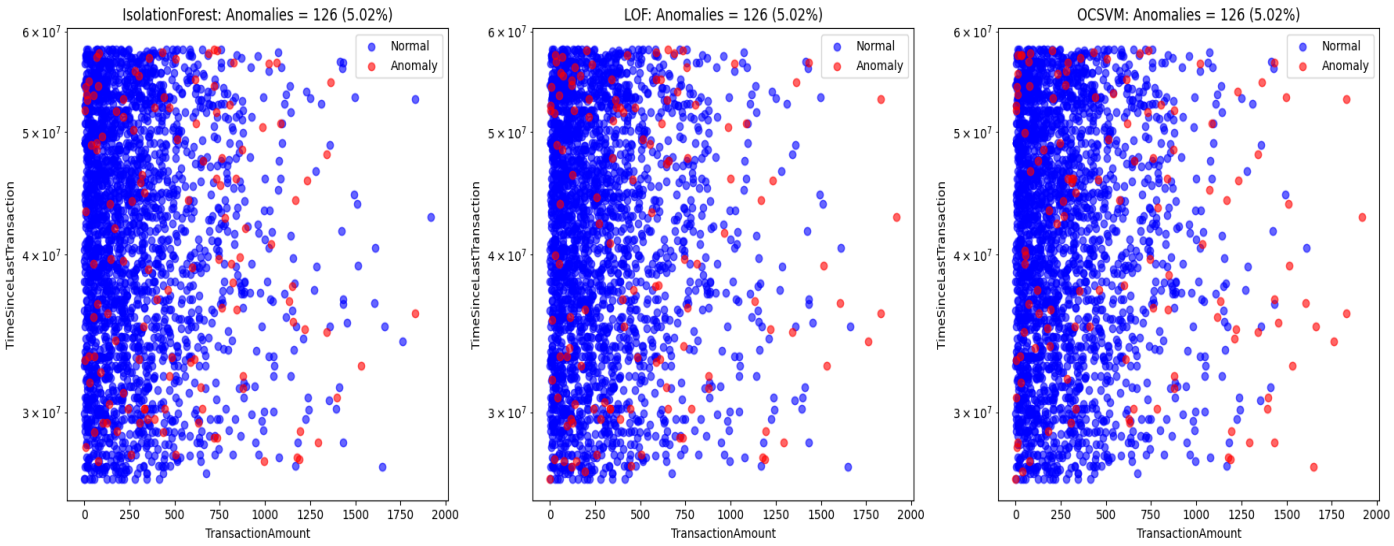

Figure 4.8: Scatter plot showing anomalies (red) and normal transactions (blue) detected by the three models separately based on TransactionAmount vs TimeSinceLastTransaction.

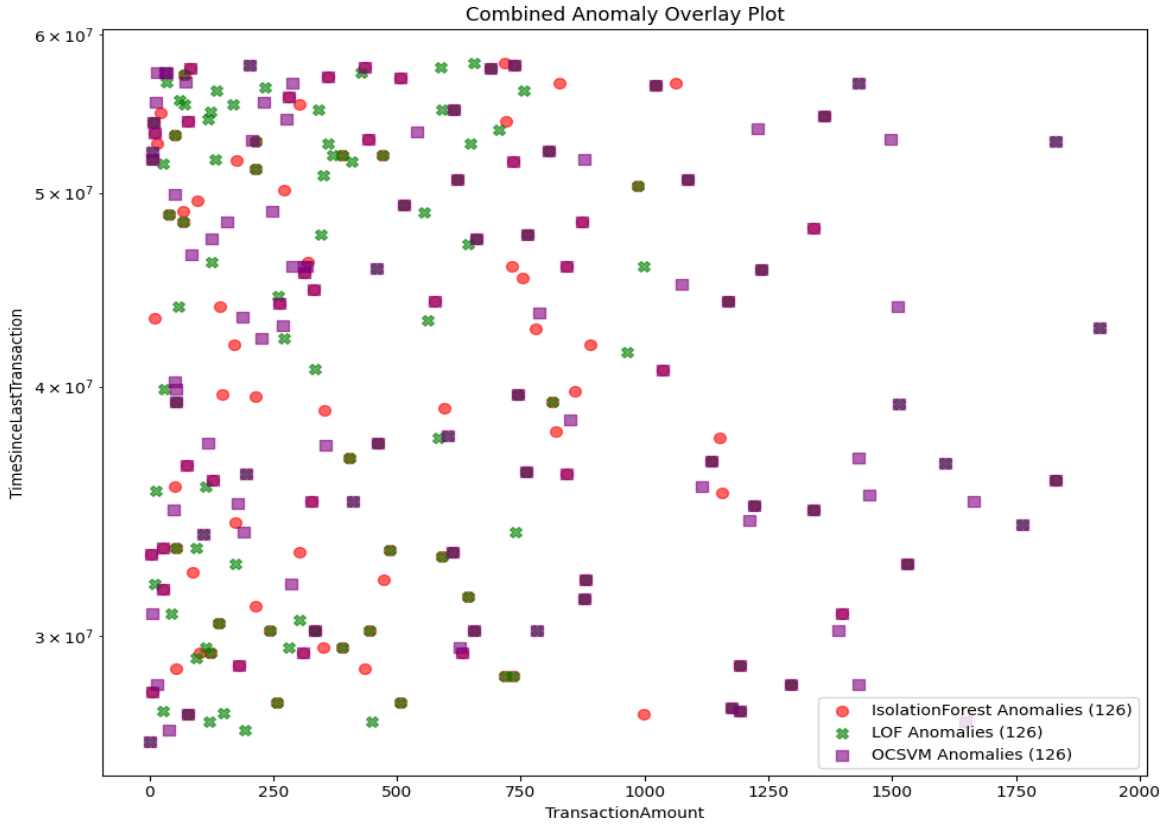

Figure 4.9: Overlay of anomalies flagged by all three models, each represented in different colors. Overlapping points indicate agreement between models.

In summary, Some anomalies were detected by all three models (indicating high-likelihood fraud or unusual activity). LOF flagged more isolated cases than the others. One-Class SVM identified anomalies that were more behaviorally unusual rather than simply high in value. Isolation Forest generally flagged high-value and irregular-time-gap transactions.

4.6 CONCLUSION

This chapter successfully presented the results of the comparative experiment, analyzed the performance of the three selected unsupervised anomaly detection models, and provided an in-depth interpretation of the transactions flagged by the best-performing solution.

The key finding of this research is the definitive selection of the One-Class Support Vector Machine (OCSVM) as the most effective algorithm for this bank transaction anomaly detection task. This conclusion was not based on arbitrary preference but on objective, data-driven metrics.

In conclusion, the research successfully designed and validated a robust anomaly detection system. The methodology moved systematically from raw data to interpretable features, allowing the OCSVM model to excel at separating high-risk transactions. The project delivers a solution where the model's decision is not a complicated and mysterious, but is transparently traceable back to specific features related to high velocity, extreme financial impact, and security friction. This outcome satisfies the core aim of designing an effective and interpretable machine learning model for detecting anomalies in bank transfers. These findings reinforce the importance of machine learning in strengthening financial transaction monitoring systems and improving the reliability and security of digital banking platforms.

CHAPTER FIVE

SUMMARY AND CONCLUSION

5.0 INTRODUCTION

This chapter presents a summary of the work carried out from the beginning of the study up to the final stage, drawing together the major procedures, findings, and outcomes. The purpose of this chapter is to reflect on the aim of the project, evaluate how the objectives were achieved, state the key conclusions derived from the results, and provide recommendations for future improvement and application of the system.

5.1 SUMMARY OF THE STUDY

This research focused on developing an effective machine learning-based anomaly detection system for identifying unusual or potentially fraudulent bank transfer transactions. The motivation for the study arose from the increasing adoption of digital banking platforms, where large volumes of transactions occur in real time, making it difficult for manual monitoring systems to detect abnormal or suspicious behavior quickly and accurately. Since labelled fraudulent transaction data is often unavailable or limited due to privacy and confidentiality concerns, the research adopted an unsupervised learning approach to identify anomalies without relying on prior examples of fraud.

The study began by acquiring and inspecting the transaction dataset, followed by data cleaning and curation to handle missing values, ensure consistency in timestamp fields, and remove duplicate records. After preparing the dataset, a set of meaningful features was engineered to improve the ability of the models to distinguish normal transactions from anomalous ones. Key engineered features included TimeSinceLastTransaction, BalanceChangeRatio, TransactionDuration, LoginAttempts, and IsNightTransaction, among others. These features captured behavioral, temporal, and financial patterns that can signal unusual account activity. To ensure comparability and reduce the influence of extreme values, the entire dataset was scaled using the RobustScaler, making the models more stable and fairer in their pattern recognition.

Three unsupervised anomaly detection models were developed and applied; Isolation Forest, Local Outlier Factor (LOF), and One-Class SVM (OCSVM). Each model was trained using the same processed dataset, with the contamination rate set to approximately 5%, meaning each model was expected to detect about 5% of the transactions as anomalies. To evaluate the

performance of the models, three key metrics were used; Silhouette Score, Anomaly Ratio, and Average Decision Score. These metrics provided insights into how well each model separated normal and abnormal transactions, how consistent the detection rate was, and how confident the model was in its classification.

The results showed that the One-Class SVM model outperformed the other two models. It recorded the highest Silhouette Score, indicating the best separation between normal and anomalous transactions, and the highest Average Decision Score, showing stronger confidence in its classifications. The anomaly ratio for all models remained close to the expected 5%, indicating stable performance across models. Further analysis of the flagged anomalies revealed common behavioral patterns, including unusually high balance shifts, very low time intervals between transactions, and transaction activities occurring at uncommon hours.

Overall, the study successfully demonstrated that an interpretable and effective anomaly detection framework can be developed using unsupervised machine learning techniques, even in the absence of labelled fraud data. The project achieved its aim of building a effective model whose decisions can be traced back to meaningful, real-world transaction behavior features.

5.2 LIMITATIONS OF THE STUDY

Although the project achieved its aim, some limitations should be acknowledged.

1. The dataset used in this study did not contain ground truth labels indicating which transactions were genuinely fraudulent, due to this, the performance of the models had to be evaluated using unsupervised metrics such as Silhouette Score and Average Decision Score instead of traditional accuracy-based measures. This means that while the system can identify unusual patterns, it cannot be conclusively stated that all flagged transactions are fraudulent without further verification from financial experts.
2. Due to the fact that the dataset represents a snapshot of transactions, it may not fully capture how transaction behaviours change over time. Fraud patterns evolve quickly, and models trained on historical data may gradually lose effectiveness if they are not updated regularly. As a result, the model's current performance is valid for the dataset used, but real-world deployment would require continuous retraining and monitoring.
3. The sensitivity of unsupervised anomaly detection models to hyperparameters, particularly the contamination rate. Setting this parameter incorrectly may lead to the model flagging too many normal transactions as anomalies or missing real anomalies.

Without labeled data, tuning this value relies heavily on judgement and domain experience.

4. Due to the time constraints of the project, the work focused on developing and evaluating the anomaly detection model rather than building a complete operational system for real-time detection. Deployment concerns such as latency, alert workflow, feedback loops, and dashboard interfaces were not implemented, although they were discussed as future improvements.
5. The performance of unsupervised models is highly dependent on the choice of parameters (e.g., `contamination`, `nu`, `n_neighbors`). Selecting optimal parameters without ground truth can be challenging.
6. While the characteristics of the flagged instances were analyzed, understanding the root cause of why a transaction is anomalous can sometimes be less straightforward compared to insights gained from a supervised model trained on labeled data.

These limitations do not reduce the value of the findings but highlight areas that require additional effort for practical, large-scale implementation in financial environments.

5.3 RECOMMENDATIONS

Based on the findings and limitations highlighted in this study, this several recommendations are proposed for practitioners, system developers, and future researchers to further improve the effectiveness and operational value of anomaly detection in bank transfers.

1. For banks and financial institutions, the anomaly detection model should be integrated into a human-in-the-loop review process rather than used for fully automated decision-making. This means that transactions flagged as anomalous should be reviewed by fraud analysts or compliance officers before any action is taken. This approach reduces the risk of customer dissatisfaction arising from false alarms while still improving fraud detection efficiency
2. For software engineers and system implementers, deploying the model as a standalone microservice is advisable. The service should include processes for saving and reusing the trained model, encoders, and scalers to ensure consistency when handling new live transaction data. Logging, version control, and monitoring services should also be implemented so that any decline in model performance is detected early. Furthermore, periodic retraining of the model is essential to account for evolving financial behavior patterns and emerging fraud tactics. This retraining could be scheduled monthly, quarterly, or triggered automatically when significant drift is detected.

3. There need for future research to collaborate with bank domain experts or fraud analysts to review the flagged transactions. Their expertise is invaluable in confirming whether the flagged instances represent true anomalies or fraudulent activities. This validation can also provide feedback to refine the model or features.
4. If possible, it will be helpful to work towards obtaining a dataset with labeled anomalies. With labeled data, you could train and evaluate supervised classification models (e.g., Logistic Regression, Random Forest, Neural Networks). Supervised models can potentially achieve higher accuracy in detecting known types of fraud and allow for the use of traditional evaluation metrics like Precision, Recall, and F1-score which are more directly interpretable in terms of correctly identifying positive cases (anomalies).
5. Explore creating more sophisticated features that capture complex patterns or relationships in the data. This could involve temporal features (e.g., velocity features like transaction count per unit time for an account), network features (if transaction graphs are available), or behavioral features.
6. Explore other unsupervised anomaly detection algorithms (e.g., Clustering-based methods like DBSCAN, Autoencoders, or Generative Adversarial Networks for anomaly detection) or explore semi-supervised methods if a small set of labeled data is available.
7. In a real-world application, the model would need to be continuously monitored for performance degradation (concept drift) and retrained periodically on newer data that includes evolving anomaly patterns.
8. Implement techniques to make the model's decisions more explainable. For unsupervised models, this could involve understanding feature importance in the anomaly scoring or visualizing the decision boundary in key feature spaces.

   By addressing these limitations and exploring these future directions, you can significantly strengthen your anomaly detection project and move closer to building a robust and practical system for identifying unusual bank transactions.

5.4 CONCLUSION

The findings of this research confirm that One-Class SVM is a well-suited approach for anomaly detection in bank transfers, particularly where labelled fraud cases are limited. The model does not only flag suspicious transactions but does so in a way that reflects meaningful behavioural and financial characteristics of the underlying data. The anomalies identified typically exhibited patterns such as abnormal transaction velocity, unusual transaction timing, or significant deviations in account balance behavior.

Overall, the project successfully met its aim. It produced a working anomaly detection model that is transparent, data-driven, and reliable, qualities that are important for real-world use in banking environments. The model and methodology developed in this study provide a practical foundation that financial institutions can build upon to enhance monitoring systems, reduce financial risk, and support secure banking operations.

REFERENCES

Azamuke, D., Katarahweire, M., & Bainomugisha, E. (2025). Financial fraud detection using rich mobile money transaction datasets. *Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, vol 588. Springer, Cham. https://doi.org/10.1007/978-3-031-81573-7_16*

Adedoyin, A. (2024). *Predicting fraud in mobile money transfer* (PhD thesis). *University of Brighton Repository. https://cris.brighton.ac.uk/ws/portalfiles/portal/6145772/13831870_PhD_Thesis.pdf*

Bakumenko, A., & Elragal, A. (2022). Detecting Anomalies in Financial Data Using Machine Learning Algorithms. *Systems*, *10*(5), 130. https://doi.org/10.3390/systems10050130

Desai, A., & Kosse, A. (2024). Pattern recognition and anomaly detection in high-frequency payments data (BIS Working Paper Presentation). *Bank for International Settlements / Bank of Canada. https://www.dnb.nl/media/minp5nsh/presentation_ajit-desai.pdf*

Elliott, A., Cucuringu, M., Martínez Luaces, M. M., Reidy, P., & Reinert, G. (2019). Anomaly detection in networks with application to financial transaction networks. *arXiv.* https://doi.org/10.48550/arXiv.1901.00402

Gopalsamy, M. (2025). Enhancing financial security based on machine learning techniques for anomaly detection in fraud transactions. *International Journal of Current Engineering and Technology, 15*(1), 55–62.

Höppner, S., Baesens, B., Verbeke, W., & Verdonck, T. (2020). Instance-dependent cost-sensitive learning for detecting transfer fraud. *arXiv.* https://doi.org/10.48550/arXiv.2005.02488

Jain, P. K. T., & Ghimire, A. (2025). Leveraging artificial intelligence for trade-based money laundering detection: A machine learning approach for anomaly detection in letters of credit and bank guarantees. *International Journal of Innovative Science and Research Technology, 10*(3), 1124–1133. *https://doi.org/10.38124/ijisrt/25mar1925*

Lokanan, M. E. (2023). Predicting mobile money transaction fraud using machine learning algorithms. *Applied Intelligence Letters, 2*(1), 30–38. *https://doi.org/10.1002/ail2.85*

Maneel, M. B. (2025). Anomaly detection in transactions using machine learning: A comparative study. *Emerging Trends in Engineering (ETE) Journal, 3*(2), 45–52. https://ete.sciten.org/index.php/ete/article/view/21

Oye, E., Wealer, A., & Aboderin, T. (2024). Integrating machine learning with data-dependence graphs for anomaly detection in financial transactions. *ResearchGate Preprint. https://www.researchgate.net/publication/390467054*

Parimi, S. S. (2017). Leveraging deep learning for anomaly detection in SAP financial transactions. SSRN. *https://doi.org/10.2139/ssrn.4934907*

Pourhabibi, T., Ong, K. L., Kam, B. H., & Boo, Y. L. (2020). Fraud detection: A systematic literature review of graph-based anomaly detection approaches. *Decision Support Systems, 133, Article 113303. https://doi.org/10.1016/j.dss.2020.113303*

Stefánsson, H. A. (2023). *Unsupervised anomaly detection in financial transactions* (Master's thesis). *Skemman Repository. https://skemman.is/bitstream/1946/44727/1/Unsupervised_Anomaly_Detection_in_Financial_Transactions.pdf*

Takahashi, S., Yamamoto, K., Kobayashi, S., Kondo, R., & Hisano, R. (2024). Dynamic link and flow prediction in bank transfer networks. *arXiv.* https://doi.org/10.48550/arXiv.2409.08718

ChatGPT *https://chatgpt.com/*

Valakhorasani, A. (n.d.). Bank transaction dataset for fraud detection (Kaggle dataset). *https://www.kaggle.com/datasets/valakhorasani/bank-transaction-dataset-for-fraud-detection.*

APPENDIX

SOURCE CODE

```
# Appendix A: Full Python Code for Anomaly Detection

# Source: DuplicateProject_bank_transaction_anormaly_detction (1) - Copy.ipynb

# ### Data loading and initial inspection

# Importing the dataset

import pandas as pd

# Importing date and time library for date and time column handling

import datetime as dt

# Loading the dataset

df = pd.read_csv("bank_transactions_data_2.csv")

df

# inspecting the dataset

# Checking the columns in the dataset

df.columns

# Checking the size of the dataset

df.shape

# Cheking the dataset infomation

df.info()

# Describing the dataset

df.describe()
```

```
# Saving a copy of the initial dataset as raw

raw = df.copy()

 ### Data curation and cleaning

# Formatting the date/time data  columns in the dataset

df["TransactionDate"] = pd.to_datetime(df["TransactionDate"], errors="coerce")

df["PreviousTransactionDate"] = pd.to_datetime(df["PreviousTransactionDate"], errors="coerce")

df["TransactionDate"].dtype

# Checking for duplicates in the dataset

df.duplicated().sum()

# Checking for missing values in the dataset

df.isnull().sum()

# Checking the current information of the dataset

df.info()

# Describing the TransactionAmount column checking for data errors

df["TransactionAmount"].describe()

# ### Exploratory Data Analytics (EDA)

# Checking the amount distribution

plot = df["TransactionAmount"]

import matplotlib.pyplot as plt

plt.figure(figsize=(10, 6))

plt.hist(plot)
```

```
plt.title("Transaction Amount Distribution")

plt.xlabel("Transaction Amount")

plt.ylabel("Frequency")

plt.show()

# Count of transactions per hour.

df.groupby(df['TransactionDate'].dt.hour)['TransactionAmount'].count().plot(kind='bar', figsize=(10,6))

plt.title("Number of Transactions per Hour")

plt.xticks(rotation = "horizontal")

plt.xlabel("Hour of Day")

plt.ylabel("Number of Transactions")

plt.show()

# Count of transactions day-of-week.

df.groupby(df['TransactionDate'].dt.day_of_week)['TransactionAmount'].count().plot(kind='bar', figsize=(10,6))

plt.title("Number of Transactions per Day of the week")

plt.xticks(rotation = "horizontal")

plt.xlabel("Day of the Week (0=Monday, 6=Sunday)")

plt.ylabel("Number of Transactions")

plt.show()

# Frequency of accounts (top senders).

plot = df['AccountID'].value_counts().sort_values(ascending= False).head(10)
```

```
plot

# Visualizing the top senders

plt.figure(figsize = (10,6))

plt.plot(plot, color = "blue", marker = "*")

plt.title("Top 10 Senders (By their Account ID)")

plt.xlabel("Account ID")

plt.ylabel("Number of transactions")

plt.grid()

# Scatter amount vs hour.

plt.figure(figsize=(10,6))

plt.scatter(df['TransactionDate'].dt.hour, df['TransactionAmount'], alpha=0.5)

plt.title("Transaction Amount vs Hour of Day")

plt.xlabel("Hour of Day")

plt.ylabel("Transaction Amount")

plt.show(

# Scatter amount vs day of the week.

plt.figure(figsize=(10,6))

plt.scatter(df['TransactionDate'].dt.day_of_week, df['TransactionAmount'], alpha=0.5)

plt.title("Transaction Amount vs Day Of the Week")

plt.xlabel("Day of the Week (0=Monday, 6=Sunday)")

plt.ylabel("Transaction Amount")
```

```
plt.show()

# ### Feature engineering

# Adding relevant features for machine learning models

df["Hour"] = df["TransactionDate"].dt.hour #adding the hour of transaction feature

df["DayOfWeek"] = df["TransactionDate"].dt.dayofweek #adding the day of the week feature

df["IsWeekend"] = df["DayOfWeek"].apply(lambda x: 1 if x >=5 else 0) # adding a weekend feature

# Creating the TimeSinceLastTransaction column

df["TimeSinceLastTransaction"] = (df["TransactionDate"] - df["PreviousTransactionDate"]).dt.total_seconds().abs() #time difference in seconds

df["TimeSinceLastTransaction"].head()

# Creating the BalanceChangeRatio column

df["BalanceChangeRatio"] = (df["TransactionAmount"]/df["AccountBalance"])

df["BalanceChangeRatio"].head()

# Dropping unneeded columns from domain research and knowledge

df = df.drop(columns = ["TransactionID", "TransactionDate", "PreviousTransactionDate", "IP Address", "MerchantID"])

df.head()

# Encoding Categorical Features

# Performing Frequency Encoding on AccountID column

account_freq = df['AccountID'].value_counts()

df['AccountID_FreqEnc'] = df['AccountID'].map(account_freq)

df.drop(columns=['AccountID'], inplace=True)
```

```
df.head()

# Performing Frequency on the Device ID

device_freq = df['DeviceID'].value_counts()

df['DeviceID_FreqEnc'] = df['DeviceID'].map(device_freq)

df.drop(columns=['DeviceID'], inplace=True)

df.head()

# Encoding the remaining categorical features using Label Encoding

from sklearn.preprocessing import LabelEncoder

label_encoder = LabelEncoder()

categorical_columns = ['TransactionType', 'Location', 'Channel','CustomerOccupation']

for col in categorical_columns:

    df[col] = label_encoder.fit_transform(df[col])

# Scaling the entire dataset using robust scaler

from sklearn.preprocessing import RobustScaler

scaler = RobustScaler()

df_scaled = pd.DataFrame(scaler.fit_transform(df), columns=df.columns)

df_scaled.head()

# ### Training the Models

# Duplicating the dataset for training

original_scaled = df_scaled.copy()

# Training with Isolation Forest
```

```
from sklearn.ensemble import IsolationForest

iso = IsolationForest(contamination = 0.05, random_state=40, n_estimators = 200)

# fitting the model on the data set

iso.fit(df_scaled)

# Predicting anomalies

df_scaled['IF_Pred'] = iso.predict(df_scaled)

# Converting output to 0 (not flagged) and 1 (flagged)

df_scaled['IF_Pred'] = df_scaled['IF_Pred'].map({1: 0, -1: 1})

# evaluating the Isolation Forest model

from sklearn.metrics import silhouette_score

scores = iso.decision_function(original_scaled)  # anomaly score: higher = more normal

silhouette = silhouette_score(original_scaled, df_scaled['IF_Pred'])

print("Silhouette Score:", silhouette)

print("Anomaly Ratio:", df_scaled['IF_Pred'].mean())

print("Avg Decision Score:", scores.mean())

# Training the LOF model

from sklearn.neighbors import LocalOutlierFactor

lof = LocalOutlierFactor(contamination=0.05, n_neighbors=20)

# copying the scaled dataframe for LOF model from the original scaled dataframe

df_lof = original_scaled.copy()

# Predicting anomalies using LOF
```

```
df_lof['LOF_Pred'] = lof.fit_predict(df_lof)

df_lof['LOF_Pred'] = df_lof['LOF_Pred'].map({1: 0, -1: 1})

# Evaluating the LOF model

from sklearn.metrics import silhouette_score

scores_lof = lof.negative_outlier_factor_  # anomaly score: higher = more normal

silhouette_lof = silhouette_score(original_scaled, df_lof['LOF_Pred'])

print("Silhouette Score (LOF):", silhouette_lof)

print("Anomaly Ratio (LOF):", df_lof['LOF_Pred'].mean())

print("Avg Decision Score (LOF):", scores_lof.mean())

# Copying the original data set in to a new variable df_ocsvm

df_ocsvm = original_scaled.copy()

# Training OneClass SVM model

from sklearn.svm import OneClassSVM

oc_svm = OneClassSVM(nu=0.05, kernel='rbf', gamma='scale')

# fitting the OneClass SVM model on the dataset

oc_svm.fit(df_ocsvm)

# predicting anomalies using OneClass SVM

df_ocsvm['OCSVM_Pred'] = oc_svm.predict(df_ocsvm)

# Making the output 0 (not flagged) and 1 (flagged)

df_ocsvm["OCSVM_Pred"] = df_ocsvm['OCSVM_Pred'].map({1: 0, -1: 1})

df_ocsvm
```

```
# Evaluating the one Class SVM model

from sklearn.metrics import silhouette_score

scores_ocsvm = oc_svm.decision_function(original_scaled)  # anomaly score: higher = more normal

silhouette_ocsvm = silhouette_score(original_scaled, df_ocsvm['OCSVM_Pred'])

print("Silhouette Score (OCSVM):", silhouette_ocsvm)

print("Anomaly Ratio (OCSVM):", df_ocsvm['OCSVM_Pred'].mean())

print("Avg Decision Score (OCSVM):", scores_ocsvm.mean())

# Identify the indices of flagged anomalies from the OneClass SVM model

anomaly_indices = df_ocsvm[df_ocsvm['OCSVM_Pred'] == 1].index

# Add a column to the original dataframe indicating whether a transaction is flagged as an anomaly

raw['IsAnomaly_OCSVM'] = 0

raw.loc[anomaly_indices, 'IsAnomaly_OCSVM'] = 1

# Separate normal and anomalous transactions

normal_transactions = raw[raw['IsAnomaly_OCSVM'] == 0]

anomalous_transactions = raw[raw['IsAnomaly_OCSVM'] == 1]

# Compare descriptive statistics for numerical features

print("Descriptive Statistics for Normal Transactions:")

display(normal_transactions.describe())

print("\nDescriptive Statistics for Anomalous Transactions:")

display(anomalous_transactions.describe())
```

```
# Summarizing the metrics already calculated for comparison.

print("Isolation Forest Performance:")

print(f"Silhouette Score: {silhouette:.4f}")

print(f"Anomaly Ratio: {df_scaled['IF_Pred'].mean():.4f}")

print(f"Avg Decision Score: {scores.mean():.4f}")

print("-" * 30)

print("Local Outlier Factor Performance:")

print(f"Silhouette Score: {silhouette_lof:.4f}")

print(f"Anomaly Ratio: {df_lof['LOF_Pred'].mean():.4f}")

# For LOF, the negative_outlier_factor_ is inversely related to anomaly score,

# so a less negative mean suggests more normal data on average.

print(f"Avg Negative Outlier Factor: {scores_lof.mean():.4f}")

print("-" * 30)

print("OneClass SVM Performance:")

print(f"Silhouette Score: {silhouette_ocsvm:.4f}")

print(f"Anomaly Ratio: {df_ocsvm['OCSVM_Pred'].mean():.4f}")

print(f"Avg Decision Score: {scores_ocsvm.mean():.4f}")

print("-" * 30)

### Visualization Script (Combined)

import pandas as pd

import numpy as np
```

```
import matplotlib.pyplot as plt

import seaborn as sns

import warnings

from sklearn.ensemble import IsolationForest

from sklearn.neighbors import LocalOutlierFactor

from sklearn.svm import OneClassSVM

warnings.filterwarnings('ignore')

# 1. Select the dataset

if 'test' in locals():

    data = test.copy()

elif 'df' in locals():

    data = df.copy()

else:

    raise ValueError("Dataset 'df' not found.")

# We'll keep the original data for plotting purposes later, but use the scaled data for prediction

if 'original_scaled' not in locals():

    raise ValueError("Scaled data ('original_scaled') not found. Please run the scaling steps.")

X = original_scaled.copy()

# Define and Train the Models (Moved from previous cells)

models = {}

# Isolation Forest
```

```
try:
    iso = IsolationForest(contamination=0.05, random_state=40, n_estimators=200)
    iso.fit(X)
    models['IsolationForest'] = iso
except NameError:
    print("Warning: IsolationForest model setup failed. Skipping.")
# LOF
try:
    lof = LocalOutlierFactor(contamination=0.05, n_neighbors=20)
    lof.fit_predict(X) # Fit and predict for LOF in non-novelty mode
    models['LOF'] = lof
except NameError:
    print("Warning: LOF model setup failed. Skipping.")
# OneClass SVM
try:
    oc_svm = OneClassSVM(nu=0.05, kernel='rbf', gamma='scale')
    oc_svm.fit(X)
    models['OCSVM'] = oc_svm
except NameError:
    print("Warning: OneClass SVM model setup failed. Skipping.")
if not models:
```

```
    raise ValueError("No trained anomaly detection models found (iso, lof, or oc_svm).")

# 4. Compute anomaly scores and labels for each model

# We'll add anomaly scores and labels back to the 'data' DataFrame for plotting

all_anomalies = pd.DataFrame(index=data.index)

for model_name, model in models.items():

    contamination = getattr(model, 'contamination', getattr(model, 'nu', 0.05))

    if model_name == 'IsolationForest':

        # Use X (original_scaled) for scoring

        scores = -model.score_samples(X)

    elif model_name == 'LOF':

        if getattr(model, 'novelty', False):

            # Use X (original_scaled) for decision_function

            scores = -model.decision_function(X)

        else:

            # For non-novelty mode, negative_outlier_factor_ is the local outlier factor

            # We negate it so higher values are more anomalous

            # Use X (original_scaled) for fitting and getting negative_outlier_factor_

            if hasattr(model, 'negative_outlier_factor_'):

                scores = -model.negative_outlier_factor_

            else:

                scores = -model.fit_predict(X) # Fallback if negative_outlier_factor_ is not available
```

```
    elif model_name == 'OCSVM':
        # Use X (original_scaled) for decision_function
        scores = -model.decision_function(X)
    else:
        scores = np.zeros(len(X)) # Should not happen with the checks above
    data[f'score_{model_name.lower()}'] = scores
    # Determine labels by selecting top-k highest scores
    n_anomalies = int(np.ceil(len(X) * contamination))
    # Ensure n_anomalies does not exceed the number of samples
    n_anomalies = min(n_anomalies, len(X))
    # Use argsort to get indices of top k scores and select labels based on these indices
    anomaly_indices = np.argsort(scores)[-n_anomalies:]
    labels = np.ones(len(X), dtype=int)
    labels[anomaly_indices] = -1
    data[f'label_{model_name.lower()}'] = labels
    all_anomalies[f'{model_name}_Anomaly'] = np.where(labels == -1, 1, 0)
# 6. Plot using seaborn/matplotlib - Use the original data columns for plotting
fig, axes = plt.subplots(1, len(models), figsize=(6 * len(models), 6))
if len(models) == 1: # Handle case with only one model
    axes = [axes]
for i, (model_name, model) in enumerate(models.items()):
```

```
    label_col = f'label_{model_name.lower()}'
    score_col = f'score_{model_name.lower()}'
    normal = data[data[label_col] == 1]
    anomalies = data[data[label_col] == -1]
    # Ensure the columns used for plotting exist in the 'data' DataFrame
    if 'TransactionAmount' in data.columns and 'TimeSinceLastTransaction' in data.columns:
        axes[i].scatter(normal['TransactionAmount'], normal['TimeSinceLastTransaction'], color='blue', label='Normal', alpha=0.6)
        axes[i].scatter(anomalies['TransactionAmount'], anomalies['TimeSinceLastTransaction'], color='red', label='Anomaly', alpha=0.6)
        anomaly_count = len(anomalies)
        anomaly_percent = (anomaly_count / len(data)) * 100
        axes[i].set_title(f'{model_name}: Anomalies = {anomaly_count} ({anomaly_percent:.2f}%)')
        axes[i].set_xlabel('TransactionAmount')
        axes[i].set_ylabel('TimeSinceLastTransaction')
        axes[i].legend()
        if data['TimeSinceLastTransaction'].max() > 1e6:
            axes[i].set_yscale('log')
    else:
        print(f"Warning: Required columns for plotting not found in the dataset for {model_name}.")
plt.tight_layout()
```

```
plt.savefig(f'scatter_individual.png') # Changed filename to avoid overwriting combined plot

plt.show()

# Combined overlay plot for anomalies - Use the original data columns for plotting

plt.figure(figsize=(10, 8))

markers = ['o', 'X', 's']

colors = ['red', 'green', 'purple']

model_idx = 0

for model_name, model in models.items():

    anomaly_data = data[data[f'label_{model_name.lower()}'] == -1]

    # Ensure the columns used for plotting exist in the 'data' DataFrame

    if 'TransactionAmount' in data.columns and 'TimeSinceLastTransaction' in data.columns:

        plt.scatter(anomaly_data['TransactionAmount'],
anomaly_data['TimeSinceLastTransaction'],

                    color=colors[model_idx],  marker=markers[model_idx],  label=f'{model_name}
Anomalies ({len(anomaly_data)})', alpha=0.6, s=50) # s for size

    else:

        print(f"Warning: Required columns for plotting not found in the dataset for combined
plot.")

    model_idx += 1

plt.title('Combined Anomaly Overlay Plot')

plt.xlabel('TransactionAmount')

plt.ylabel('TimeSinceLastTransaction')

plt.legend()
```

```
if 'TimeSinceLastTransaction' in data.columns and data['TimeSinceLastTransaction'].max() > 1e6:
  plt.yscale('log')
plt.tight_layout()
plt.savefig('scatter_combined.png')
plt.show()
# 7. Print summary lines
print("\n--- Model Performance Summary ---")
for model_name, model in models.items():
    label_col = f'label_{model_name.lower()}'
    score_col = f'score_{model_name.lower()}'
    if label_col in data.columns and score_col in data.columns:
        anomalies = data[data[label_col] == -1]
        normal = data[data[label_col] == 1]
        anomaly_count = len(anomalies)
        anomaly_percent = (anomaly_count / len(data)) * 100
        avg_score_anom = anomalies[score_col].mean() if not anomalies.empty else 0
        avg_score_norm = normal[score_col].mean() if not normal.empty else 0
        print(f"{model_name}: anomalies={anomaly_count} ({anomaly_percent:.2f}%), avg_score_anom={avg_score_anom:.4f}, avg_score_norm={avg_score_norm:.4f}")
    else:
        print(f"Warning: Anomaly labels or scores not found for {model_name}.")
```