ANOMALYIQ: A THREE-PHASE COMBINED ANOMALY DETECTION MODEL FOR BROKERAGE TRADING OPERATIONS EMPLOYING AUTOENCODER, ISOLATION FOREST, AND LIGHTGBM WITH SMOTE

BY

NWAGOR O. PASCHAL

REG. NO: GOU/U22/CSC/931

Project Report Submitted To The Department Of Computer Science And Mathematics, Faculty Of Computing and Information Technology, Godfrey Okoye University, Enugu State

In Partial Fulfillment For The Requirements For Awarding The Degree of Bachelor of Science In Computer Science

SUPERVISOR:

Dr. Frank Okebanama

MAY, 2025

APPROVAL PAGE

This research, titled "ANOMALYIQ: A THREE-PHASE COMBINED ANOMALY DETECTION MODEL FOR BROKERAGE TRADING OPERATIONS EMPLOYING AUTOENCODER, ISOLATION FOREST, AND LIGHTGBM WITH SMOTE," has been assessed and approved by the Department of Computer Science and Mathematics Committees in the Faculty of Computing and Information Technology, Godfrey Okoye University, Enugu, Nigeria.

	Signature	**Date**
Dr. Frank Okebanama	.....	.....
Supervisor		
External Examiner	.....	.....
External Examiner		
Dr. Frank Okebanama	.....	.....
Sub-dean/HOD, Department of Computer Science and Mathematics		
Prof. I. A. Ajah	.....	.....
Dean, Faculty of Computing and Information Technology		

CERTIFICATION

I certify that I completed the research included herein and that it was primarily based on my observations and investigation. Consequently, I will fully accept the University's decision if I am found to have violated academic integrity in any form.

_____

NWAGOR O. PASCHAL

REG. NO: GOU/U22/CSC/931

DEDICATION

This project is dedicated to my parents, whose unwavering sacrifices, prayers, and encouragement made this academic journey possible, and to every Nigerian compliance professional working tirelessly to protect the integrity of our financial markets.

ACKNOWLEDGEMENTS

My sincere thanks go to Dr. Frank Okebanama, who is the project supervisor, for his valuable guidance, constructive criticisms, and timely motivation during the conceptualization of this work and its development until it became a reality. It was a major intellectual input into this research for him to emphasize that the entire data preprocessing process was a design choice rather than an arbitrary step.

I thank the Head of the Department and all lecturers in the Department of Computer Science and Mathematics of Godfrey Okoye University, for their lectures and guidance for the past four years of study, without which I would never have been able to carry out this research. It is hereby acknowledged with gratitude all those lecturers that made it possible for me to develop critical thinking, analytical abilities, and theoretical approach to real-life issues.

I am greatly appreciative of the entire staff and management of Stanbic IBTC Stockbrokers, Lagos, for the Students Industrial Work Experience Scheme (SIWES). The experience gained there was the starting point of AnomalyIQ.

My parents and family…your prayers, perseverance, and faith in my capabilities made me overcome every tough situation in my life. This is our achievement equally, for I could not achieve it without you. To my friends and course mates who inspired me and provided peer feedback along the way thanks.

Above all, I praise God Almighty, the fountain of wisdom and knowledge, without which none of this could have happened.

ABSTRACT

Anomaly detection in financial fraud within the context of stockbrokerage in Nigeria and mobile money transactions continues to defy rule-based approaches, which are unable to respond to new patterns of fraud, unable to scale up with increasing transactions, and do not offer any mechanism for prioritization and explanation. This paper introduces the semi-supervised AnomalyIQ, which represents a three-stage architecture for hybrid anomaly detection and fraud classification, thereby overcoming the challenges posed by rule-based techniques. Stage 1 involves training a pure-NumPy Autoencoder solely using normal transactions and then producing an anomaly indicator based on reconstruction error set at the 95th percentile from the training errors. The second stage involves applying an isolation forest algorithm that uses random recursive partitioning to derive the structural independence of outliers. The third stage involves training a LightGBM classifier using SMOTE-balanced data. The meta-features of base features and stages one and two will be used to enable supervised fraud classification. The thresholds that yield the highest Precision and Recall scores in one optimization process are chosen out of 1,000 candidate values, while every detected anomaly is labeled as having Low, Medium, or High risk based on SHAP TreeExplainer feature-level explanations. Evaluation of the proposed system was carried out using two public benchmarks: Kaggle Credit Card Fraud Detection dataset (total number of records: 284,807; fraud rate: 0.172%) and PaySim Synthetic African Mobile Money dataset (total number of records: 6,362,620; fraud rate: 0.13%). For the Kaggle Credit Card dataset, the system achieved Precision 100.00%, Recall 98.67%, F1-Score 99.33%, and AUC-ROC 99.90% for 56,961 test records without a single false positive result, versus 496 false positives for the standalone Autoencoder model. For the PaySim dataset, precision 98.03%, recall 98.12%, F1-score 98.07%, and AUC-ROC 99.99% were recorded for 180,000 stratified test records, representing an improvement of 58.3 percentage points in F1-score for the two-stage unsupervised pipeline (F1=0.41). These findings serve as empirical confirmation of the system design hypothesis. The whole architecture can be accessed as a fully functional web application available under https://anomalyiq-rho.vercel.app

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER ONE

INTRODUCTION

1.1 Background of the Study

Over the last three decades, there have been remarkable changes brought about by digitalization in financial markets all over the world. In today's world, there are millions of transactional activities conducted electronically through digital trading systems, even in Nigeria via the Central Securities Clearing System (CSCS). As much as digitization has been helpful in making financial market transactions fast and easy, it has come with its fair share of issues. According to the Association of Certified Fraud Examiners (2022), companies around the world experience loss of about five percent of their annual incomes due to fraud, with the financial sector being the hardest-hit industry. The Securities and Exchange Commission Nigeria (2023) has also expressed concerns about the high rate of digital transformation in the capital market outpacing the ability of most firms to detect fraud, resulting in massive losses.

The current methods employed by Nigerian stockbroker transactions for transaction monitoring involve the use of automated filters based on rule-based technology and compliance review. The rule-based approach evaluates a transaction using predetermined criteria, and it is therefore constitutionally unable to uncover the existence of fraudulent methods which had not been thought of during the development of the rules. As Bolton and Hand (2002) found, there was a fundamental flaw in earlier forms of electronic trading where automation made it difficult to police because of the speed gains. The other problem in the current transaction monitoring process is manual review, where the size of the team able to process 10,000 alerts daily is unlikely to expand to handle 100,000 daily alerts despite increasing the efforts put into the process, according to Chen et al. (2022). In such cases, there is no alert priority or confidence level assigned to alerts.

The above problem was actually experienced first-hand by the researcher when he was working with Stanbic IBTC Stockbrokers in Lagos as a KYC officer through the Students Industrial Work Experience Scheme (SIWES). These were not just hypothetical problems since they occurred during actual day-to-day operations within the compliance department. This gave birth to the main research question: Is there an algorithmic method to automatically learn normal behavior from past data, detect cases of fraud with high accuracy, label alerts based on risk levels, and provide explanations for those cases of abnormal behavior, all without the use of rule-based systems?

Machine learning algorithms have proven effective at tackling this problem. The basic classification of anomaly detection identifying point, contextual, and collective anomalies was defined by Chandola, Banerjee, and Kumar (2009), and forms the minimum scope of detection that a viable system would be expected to cover. Yet there are inherent drawbacks associated with each machine learning algorithm. Unsupervised models, including Autoencoders and Isolation Forests, are effective at detecting statistical anomalies, but not necessarily verified fraud; they do not have the ability to utilize any fraud labels present in the dataset for training. The usefulness of Autoencoders for anomaly detection using the reconstruction error method was proved by Zhou and Paffenroth (2017), while Liu, Ting, and Zhou (2008) explained how Isolation Forests efficiently isolate outliers based on random recursive partitioning, regardless of distribution. Gradient boosting algorithms have excellent precision performance on labelled data, but struggle significantly with data imbalance, where only 0.172% of all cases represent fraudulent transactions, forcing most models to predict all cases as normal.

A wide range of hybrid or ensemble methods have become popular to overcome the drawbacks of each of these algorithms. According to Zong et al. (2018), a combination of Autoencoders and Gaussian Mixture Models has proved to be more effective than using one of these models separately. Carcillo et al. (2019) found that SMOTE-based class balancing for low-k neighbor sizes yields the best performance in cases where fraud

detection is concerned. LightGBM is a highly effective gradient boosting algorithm for tabular data, proposed by Ke et al. (2017) as an algorithm that provides optimal precision along with the right balance of training data. Finally, Ngai et al. (2011) showed that better feature engineering and preprocessing often matter more than a specific algorithm used.

For an overarching solution, AnomalyIQ is proposed as a three-stage semi-supervised hybrid anomaly detection framework. In Stage 1, an anomaly score is generated using a pureNumPy Autoencoder based on historical data and learning normal transaction behaviour through unsupervised techniques. In Stage 2, the Isolation Forest model is used to obtain a structural anomaly score. Finally, in Stage 3, LightGBM is used as a classifier that takes into account a balanced dataset created using SMOTE sampling. Additionally, the results of Stages 1 and 2 are used as engineered features. The superiority of Autoencoder over linear models on structured numerical data was empirically shown by Sakurada and Yairi (2014), thus providing justification for the design of stage 1 thresholds. Then the output of the three stages is aggregated into one anomaly score, and transactions marked as anomalies are classified into Low, Medium, or High risk. For explanation purposes, SHAP values are computed for each of the anomalous transactions, following Lundberg and Lee (2017). The entire solution is implemented as a functional web app using React.js frontend and FastAPI backend.

AnomalyIQ was tested on two datasetss from benchmarking: the Kaggle Credit Card Fraud Detection dataset (Dal Pozzolo et al., 2015), which has 284,807 transactions and a fraud rate of 0.172%; and the PaySim Synthetic African Mobile Money dataset (López-Rojas et al., 2016), with 6,362,620 transactions and a fraud rate of 0.13%. PaySim was chosen because of its simulated West African mobile money transaction patterns, which match the geography of Nigeria and its financial environment. High performance on both different structures proves that the model is highly generalizable in various financial environments.

1.2 Statement of the Problem

There are deficiencies with the existing practices in Nigeria regarding the monitoring of electronic stockbroking and financial activities in terms of their adequacy in managing the high level of transactions' speed, volume, and complexity. Rule-based techniques can be able to catch only those cases of fraud that were predictable prior to the creation of rules as anything designed after their implementation will not get caught by the system at all. Chen et al. (2022) revealed that there is an absolute limit of transaction monitoring capabilities that manual verification cannot increase due to its inability to increase linearly with the rise in transactions.

Most existing fraud detection algorithms in the research literature generally offer binary outputs such as normal or anomalous data without providing any fraud probabilities, risks, or explanations to be used for making compliance decisions. As highlighted by Pumsirirat and Yan (2018), such problems were observed even in highly successful cases of using autoencoder algorithms for fraud detection because high performance does not always guarantee that the algorithms generate explanations needed by compliance officers to make informed decisions. In another study, Jurgovsky et al. (2018) found that static approaches focused on transactions could ignore potential frauds revealed by sequential modeling due to their time-based nature, which is why temporal features were included in AnomalyIQ's feature engineering process.

1.3 Aim and Objectives

Aim

To develop and execute a smart system to detect the anomalies in financial transactions by adopting a three stage hybrid concept with using Autoencoders, Isolation Forests and LightGBM.

Objectives

1. Identify the operational constraints of the rule-based fraud detection systems, and to characterise the features of the two datasets, namely Credit Card (284,807 transaction) and PaySim (6.3 million transaction), the top performing features were selected and temporal and statistical variables were engineered to be used as inputs for the anomaly detection pipeline, such as the transaction hour, rolling mean amount, and amount deviation.
2. Design an automatic, adaptive data preprocessing pipeline that can detect the transaction format (normal or structural anomaly) of a dataset automatically and scale and encode it accordingly and build and train its 3-stage hybrid detection pipeline.
3. Integrate the results of all three detection phases into a single fraud probability rating that assigns a Low, Medium or High level to each transaction that is flagged, and to make the entire system (including feature explanations using SHAP scores and role-based access control) available to everyone as a fully fledged web application.
4. Validate the 3-stage AnomalyIQ pipeline with two- and one-stage baseline models on both benchmark datasets, and to show that the system can get a Precision and Recall of 0.95 or higher on both the Credit Card dataset and PaySim dataset.

1.4 Significance of the Study

First, the significance of this work is twofold. Methodologically, this work is different from all known voting-based ensembles in its technical approach in that it employs unsupervised models' outputs as engineered meta-features fed into the supervised classifier; there is no need to combine predictions via averaging. Aros et al. (2024) have proven that using feature-feeding approach produces better discriminative representation compared to the averaging approach. The achieved results on the Credit Card dataset (Precision 100.00%, Recall 98.67%) and the PaySim dataset (Precision 98.03%, Recall 98.12%) can be regarded as proof of the efficiency of the methodology.

Operationally, AnomalyIQ offers Nigerian financial regulatory units an additional layer of functionality that is unavailable from classic rule-based systems: namely, risk-ranked alerts, SHAP-based explanation of transactions, scoring by individual transactions, and a publicly available website requiring no software installation at users' end. The practical significance of the fact of absence of false positives on the Credit Card test set is enormous – the stand-alone Autoencoder has produced 496 false positives, while AnomalyIQ has produced none.

From an academic standpoint, this project is one of the few contributions within the existing literature focused on the evaluation of hybrid anomaly detection systems based on geographically pertinent African financial data sets. The employment of the PaySim data set, which captures the mobile money transaction patterns of West Africa (López-Rojas et al., 2016), makes AnomalyIQ a geographically apt contribution to the field of fraud detection.

1.5 Scope of the Study

The structured data for the project are stored in CSV files. The two benchmark datasets were selected: The Kaggle Credit Card Fraud Detection dataset with 284,807 transactions, and a fraud rate of 0.172%, and PaySim Synthetic African Mobile Money dataset with 6,362,620 transactions, and a fraud rate of 0.13%. The key libraries that were used are: NumPy for Autoencoder, scikit-learn for Isolation Forest and StandardScaler, LightGBM for Stage 3, imbalanced-learn for SMOTE, SHAP for explanation, FastAPI for the backend, and joblib for model persistence. The frontend was created using React.js, React Router, and Axios. The project has no real-time streaming of transactions, live CSCS, and does not make regulatory compliance filings. The frontend can be accessed at https://anomalyiq-rho.vercel.app, while the backend is available at https://anomalyiq-production.up.railway.app.

1.6 Limitations of the Study

One drawback here is the use of publicly available benchmark datasets instead of transaction data from Nigerian stockbrokerage firms, considering data privacy policies do not allow the use of proprietary information. Although the Credit Card and PaySim benchmark datasets are widely used benchmark datasets for imbalanced fraud detection, they might not include fraud patterns in transactions of Nigerian stockbrokerage firms. Chalapathy & Chawla (2019), in their review paper on deep learning based anomaly detection techniques, discussed this issue of datasets generalization. The technical limitation, that is, having a limited environment of 8 GB RAM, forced the researchers to limit the number of samples of PaySim Autoencoder to only 150,000 normal transactions.

1.7 Operational Definition of Terms

Anomaly Detection: Automatic identification of observations which are inconsistent with normal behavior patterns; in our case, suspicious transactions that may be fraudulent.

Autoencoder: Unsupervised neural network for encoding into a low-dimensional space and decoding out of that space. $L(x, \hat{x}) = ||x - \hat{x}||^2$, which indicates anomaly based on reconstruction error, with high anomalies being represented by hard-to-reconstruct transactions according to the learned normal distribution of the model.

Isolation Forest: Unsupervised learning algorithm used to identify outliers by measuring the number of split steps necessary to isolate an individual sample. Transactions requiring fewer split operations are considered anomalies and serve as part of the AnomalyIQ Stage 2 model.

LightGBM: Gradient boosting machine learning framework using leaf-wise tree growth to efficiently classify tabular data. It is implemented as a Stage 3 supervised learner in AnomalyIQ.

SMOTE (Synthetic Minority Over-sampling Technique): A technique proposed by Chawla et al. (2002) for generating synthetic samples from minority classes using linear interpolation between adjacent minority samples. Used prior to Stage 3 model training to balance out the dataset.

Three-stage Hybrid Model: Autoencoder (stage 1, weight 0.25) combined with Isolation Forest (stage 2, weight 0.25) and LightGBM (stage 3, weight 0.50) model with stage 1 and stage 2 results being used as meta-features within stage 3 training data.

Risk Level: Categorical classification (Low, Medium, or High) applied to detected anomalies using a combined anomaly score. High risk: anomaly score higher than 0.80; medium: anomaly score between 0.65 and 0.80; low: from detection point to 0.65.

SHAP (SHapley Additive exPlanations): Game theory approach proposed by Lundberg and Lee (2017) for interpreting machine learning models' output using feature marginal contributions.

React.js: Framework in JavaScript used for building applications that are composed of components. AnomalyIQ front end development is done using React.js.

FastAPI: High-speed Python web framework that supports asynchronous functionality and API documentation. It is the backend service for AnomalyIQ, through which all operations relating to machine learning pipelines and user management are available.

JWT: JSON Web Token – compact, self-contained transmission format designed to carry authentication claims. Used in AnomalyIQ for user authentication, having a 72-hour expiration time with role permission encoded in its payload.

SIWES: Students Industrial Work Experience Scheme – a Nigerian federal program offering supervised industrial attachment experience for undergraduate students. The researcher's own SIWES internship at Stanbic IBTC Stockbrokers inspired this study.

Railway: Cloud platform-as-a-service solution that helps in deploying the AnomalyIQ backend, offering traffic routing to the deployed application without size restriction on proxy file sizes and persistent storage for trained model files.

Vercel: Deployment cloud service that offers global CDN support and automated HTTPS setup for hosting React.js frontend applications, also integrating with GitHub for continuous deployment.

CHAPTER TWO

LITERATURE REVIEW

2.1 Introduction

The chapter is an important analysis that will discuss the role of AnomalyIQ within the broader context of anomaly detection, fraud detection by using machine learning models and ensembles, as it applies to financial situations. There will be four sections of the review, which include an analysis of the theoretical bases of anomaly detection, a detailed discussion of twenty studies relevant to the topic at hand, presentation of the most important results in a table format, and clear formulation of the problems that are left unsolved by previous research and that can be solved by AnomalyIQ's approach to the matter. This analysis serves to show the ways in which prior knowledge in literature has helped develop or constrained the creation of solid anomaly detection tools in financial cases and to highlight precisely how the three stages that constitute the model solve the specific issues that have been found through literature review.

2.2 Theoretical Background

2.2.1 Stockbroking Transaction Systems and Financial Fraud

Transaction record in stock broking represents the performance of a customer's securities order through an authorized brokerage firm and consists of parameters such as security instrument, trade volume, amount, date/time, and customer identification number. In top Nigerian stockbroking firms, thousands of transactions records are made each trading day using the CSCS platform. As stated by Bolton and Hand (2002), one of the earliest findings about electronic trading was the creation of an inherent policing

difficulty since the efficiencies gained in the process were accompanied by challenges in detecting fraudulent transactions.

As confirmed by the CSCS (2023), there has been an increased adoption of electronic trading platforms but, according to the SEC Nigeria (2023), the advancements in the area of fraud detection have not been proportional to the growth in transaction volumes.

2.2.2 Concept and Types of Anomaly Detection

Anomaly detection: The automatic detection of data points that significantly deviate from patterns. The taxonomy that remained common in the field was “point anomalies” (individually anomalous transactions), “contextual anomalies” (suspicious only in context, e.g., a large trade at an unusual time), and “collective anomalies” (groups suspicious together despite being individually unremarkable, e.g., avoiding a threshold amount of transactions). All three are handled by a detection system which needs to work at multiple levels of abstraction, which is one of the reasons for the three-stage architecture. The Autoencoder and Isolation Forest are used to identify point and contextual anomalies and engineered temporal features (transaction_hour, rolling_mean_amount, amount_deviation) add contextual awareness to the per-transaction feature space.

2.2.3 Machine Learning Techniques within Anomaly Detection

Machine learning introduced true improvements rather than just rule-based instead of relying on rules, it learned the classification boundaries from the data. At PaySim's scale of 6 million records distance based methods (k-NN, LOF) are impractical. Isolation Forest (Liu et al., 2008) is one such unsupervised method that measures how easy it is to isolate a data point, rather than how far it is from normality, and it is also fast to scale when dealing with large data sets, and requires no statistical assumptions. Gradient Boosting frameworks such as LightGBM (Ke et al., 2017) have high precision with class balanced labelled data, however, the primary challenge is balancing in this project, which is solved by SMOTE.

2.2.4 Deep Learning and Autoencoder Models

Conventional shallow models are not able to learn complex hierarchical representations which deep learning is capable of capturing (LeCun, Bengio, and Hinton, 2015). The Autoencoder that is trained with only normal data learns to reconstruct normal transactions well, and fails to reconstruct anomalous transactions well, providing a good unsupervised anomaly signal in the form of reconstruction error. Zhou and Paffenroth (2017) empirically tested this mechanism with structured data sets. The pure-NumPy implementation for AnomalyIQ removes TensorFlow dependency without compromising the math computation, supports Python 3.14 and helps to minimize installation footprint.

2.2.5 Hybrid and Ensemble Approaches

The realization that no single approach works better for all kinds of anomalies drove researchers towards hybrid models. According to Zong et al. (2018), combining an autoencoder with another model leads to superior results compared to the use of the two in isolation. Purely unsupervised two-stage models suffer from the same problem of being fundamentally limited to identifying statistical anomalies but not necessarily fraudulent activities. The crucial element of this research is the third, supervised stage in which LightGBM will be used to build a model based on labeled data, where inputs from the first two stages will be added as meta-features in direct relation to the fraud label (Angiulli et al., 2023). As shown by Carcillo et al. (2019), applying SMOTE oversampling in this supervised stage increases the sensitivity without impacting precision (Noto et al., 2011).

2.3 Review of Related Works

Chandola, Banerjee, and Kumar (2009) gave the most comprehensive survey of anomaly detection methods, and used the three taxonomy categories (point, contextual, collective) throughout this study. The restriction for the system development of a practical system is the mostly theoretical scope and the little guidance about the implementation in high frequency financial environments. Relevance: includes the vocabulary and implicit argument for the anomaly classification, as well as the multi-mechanism detection.

Liu et al. (2008) proposed a new approach called Isolation Forest, which showed that it is more efficient to detect anomalies by measuring their ease of isolation rather than modelling the normality. The algorithm was found to be effective in terms of computation time in highdimensional data sets and without probabilistic assumptions. Only one of its drawbacks is the lack of temporal or relational pattern modelling for each transaction. Adopted directly as Stage 2 of AnomalyIQ.

Chalapathy & Chawla (2019) conducted a thorough survey of deep learning methods for anomaly detection and found that, under usual computational requirements, standard autoencoders are more accurate than variational or GAN-based methods when it comes to the accuracy/cost ratio. Relevance: used to show that the Stage 1 autoencoder architecture is more relevant than more complex alternatives.

An and Cho (2015) extended autoencoders in the context of reconstruction probability from a variational formulation, instead of just reconstruction error. The approach is more principled but has been more difficult to calibrate in practice. Due to the practical tractability of the simpler formulation for reconstruction-error from Zhou and Paffenroth (2017) that was adopted for Stage 1, the former was retained in Stage 2 to maintain relevance.

The variational encoder framework was first introduced by Kingma and Welling (2014). It is important to understand this work to make decisions regarding the architecture of the Autoencoder, as to know what is learnt in the bottleneck layer and why the reconstruction error is a valid anomaly signal. Relevance provides background theory on the architecture selection for Stage 1.

LeCun, Bengio, and Hinton (2015) gave a summary of what was done in each of the major areas of deep neural networks and scientifically explained why they should use a neural network in Stage 1 and not something shallower. Relevance: gives the reason why the neural network is relevant for Stage 1.

Ahmed et al. (2020) found that most of the classifiers learnt to completely ignore fraud when trained on highly imbalanced financial data, empirically. If the data set is imbalanced, then the accuracy is not meaningful without dealing with the imbalance explicitly. Changes in

Relevance: motivated SMOTE before Stage 3 training and used Precision, Recall, F1 and AUCROC as primary metrics instead of accuracy.

Géron (2019) is referenced as a guide for practitioners for the implementation of Isolation Forest, training neural networks, and hyperparameter tuning for gradient boosting. Explain how code is relevant: tie theory of algorithms to decisions about implementation

Candès, Li, Ma and Wright (2011) suggested Robust PCA for isolation of the normal and the abnormal components of the matrix. It was tested and found unsuitable as a preprocessing option since financial transaction data is often not normally distributed or low rank and has nonstationary temporal patterns. Relevance: thoughtfully chosen and ignored, validating the autoencoder's selection.

Bolton and Hand (2002) described the shortcomings of statistical fraud-detection methods in early electronic trading, which this project exceeds. The essence of their question – a monitoring tension created by electronic trading – remains relevant today, 20 years later. Relevance: sets the historical background and the context of the issue.

Despite the algorithm used, Ngai, Hu, Wong, Chen and Sun (2011) discovered that initial preprocessing and the process of deriving features can affect the model's effectiveness more than anything else. Relevance: shows the importance of the effort put into the AnomalyIQ preprocessing pipeline, including the engineered features that were designed for PaySim.

The most obvious empirical reason for the hybrid design philosophy of AnomalyIQ is the results of Bhattacharyya, Jha, Tharakunnel and Westland (2011), who compared several classifiers for the Credit Card Fraud dataset and concluded that none of the classifiers outperform the others on all the metrics. Power vs. relevance: most immediate empirical motivation for combining multiple approaches is relevance. The combining of multiple approaches is most motivated by relevance (power).

Jurgovsky et al. (2018) proved that sequential LSTM models are able to learn the timebased fraud patterns which are hidden in static per-transaction models. In this study, a sequential model is not used, but the result inspired the creation of the rolling_mean_amount, amount_deviation, and transaction_hour features, which were

added to the static feature encoding in order to add temporal information. Relevance: drove motivated TEF decisions.

Here, Fiore et al. (2019) demonstrated that GANs-based training is superior in the case of severe imbalance, where processing load prevents the use of GANs. The main result that the class imbalance needs to be explicitly solved in a model was used as an argument for employing SMOTE. Relevance: confirmed importance of resampling and SMOTE was favoured due to computational restrictions.

Zhou and Paffenroth (2017) directly validated Stage 1 by showing that the reconstruction error of an Autoencoder is a good measure of distinguishing between normal and anomalous records in structured data even if the training data has some noise. Relevance: directly confirms the detection mechanism in Stage 1.

Pumsirirat and Yan (2018) compared autoencoders with Restricted Boltzmann Machines for credit card fraud detection, which demonstrated that autoencoders were more consistent than
RBMs under the different threshold values. Most deep learning fraud identification systems also failed to explain the gap that is covered by AnomalyIQ's SHAP module. Echoing the findings of Stage 1 architecture, relevance confirms that there is an explainability gap.

Carcillo, Dal Pozzolo, and colleagues (2019) systematically examined the resampling approach for rare fraud detection problems and concluded that SMOTE with small values of k does best on the precision-recall curve. The k=3 value recommended for datasets with less than 500 minority samples were directly used in the Credit Card SMOTE configuration.The k=3 value recommended for datasets with less than 500 minority samples was used directly in the Credit Card SMOTE configuration. Relevance: k=3 is used for SMOTE configuration because of its strong empirical support.

Sakurada and Yairi (2014) showed that autoencoders perform better than PCA for structured numerical anomaly detection, and their method of calculating percentile-based threshold, where the anomaly boundary is set by the 95th percentile of training errors, informed the threshold calculation in Stage 1. Relevance: supports advantage of Stage 1 over linear models; informs threshold calculation.

Ruff, Vandermeulen, et al. (2018) proposed Deep SVDD for one-class anomaly detection. It was tested as a possible alternative to the standard autoencoder in Stage 1 and rejected because of its sensitivity to initialization and more complex implementation. Relevance: tested as an alternative Stage 1 model; rejection confirms the standard autoencoder selection.

Zong, Song, et al. (2018) developed a deep autoencoder combined with Gaussian Mixture Model. They showed that combining an autoencoder with another model consistently results in improved performance. Their study provides the strongest experimental evidence for the design of the AnomalyIQ pipeline architecture, with the exception that AnomalyIQ has an additional supervised LightGBM Stage 3 that makes use of actual fraud labels to overcome precision ceiling of purely unsupervised two-stage approaches. Relevance: most relevant to multi-stage stacked architecture design.

2.4 Summary of Related Works

S/N	Author(s) and Year	Contribution	Technique Used	Limitation	Relevance to AnomalyIQ
1	Chandola, Banerjee, and Kumar (2009)	Taxonomy of Anomaly Detection that encompasses Point, Contextual, and Collective anomalies	Overview of statistical and machine learning based approaches	Conceptual approach with little deployment	Anomaly Taxonomy adopted extensively in this project
2	Liu, Ting, and Zhou (2008)	Suggestion for Isolation Forest – Isolation technique to detect outliers via recursive partitioning	Isolation Forest	Does not have any temporal modeling of transactions	Isolation Forest adopted directly into AnomalyIQ Stage 2

3	Chalapathy and Chawla (2019)	Review of deep learning techniques used for anomaly detection in various applications	Autoencoder, Recurrent Neural Networks, Generative Adversarial Networks	Needs big data sets and substantial computational power	Provides support for Stage 1 Autoencoder architecture selection
4	An and Cho (2015)	Used reconstruction probability from Variational Autoencoder as an anomaly measure	Variational Autoencoder	Threshold calibration is more challenging than in normal Autoencoders	Stage 1 threshold selection philosophy
5	Kingma and Welling (2014)	Development of the Variational Autoencoder concept for deep latent variable models	Variational Bayes	More computationally demanding than normal Autoencoders	Basis for Stage 1 Autoencoder selection
6	LeCun, Bengio, and Hinton (2015)	Analyzed the development of deep neural networks in various applications	Deep Neural Networks	Low interpretability of the decisions made by deep networks	Provides a reason to use neural networks in Stage 1
7	Ahmed, Mahmood, and Islam (2020)	Demonstrated that classifiers perform badly with imbalanced data and must be balanced to avoid this	Support Vector Machine, Random Forest, Neural Network	Dramatic performance deterioration occurs when data is imbalanced	Based on the findings, a reason to use Synthetic Minority Oversampling Technique in Stage 3

8	Géron (2019)	Description of how to implement various machine learning techniques, including Isolation Forest and gradient boosting	Various machine learning methods	Does not focus specifically on financial domain	Applicable in all three stages of implementation process
9	Candès, Li, Ma, and Wright (2011)	Introduced Robust Principal Component Analysis for detecting sparse outliers among normal matrix components	Robust Principal Component Analysis	Requires data to have stable low rank, while our dataset does not	Applicable for the choice of preprocessing; used to confirm using Autoencoder
10	Bolton and Hand (2002)	Discussed the policing problems emerging due to efficiency improvement in fraud detection via statistics	Statistical methods	Unable to detect nonlinear frauds	Provides necessary historical background to this problem
11	Ngai, Hu, Wong, Chen, and Sun (2011)	Established preprocessing and feature engineering superiority over classifier algorithm in fraud detection	Discussion of multiple data mining methods evaluation	Feature engineering requires very careful domain-specific selection	Emphasizes importance of AnomalyIQ's preprocessing pipeline design
12	Bhattacharyya, Jha, Tharakunnel, and Westland (2011)	Discovered that no single classifier is clearly superior to other on fraud detection datasets	Includes classifiers such as logistic regression, support vector machine, and random forest	Results are specific to each dataset and cannot be generalized	Provides best empirical evidence for multistage approach

S/N	Author(s) and Year	Contribution	Technique Used	Limitation	Relevance to AnomalyIQ

13	Jurgovsky, Granitzer, and colleagues (2018)	Discovered that Long Short-Term Memory networks capture fraud patterns based on sequences impossible for per-transaction analysis	Recurrent Neural Network, Long Short-Term Memory	Necessitates the presence of timebased features such as rolling average and transaction hour	Explains motivation behind temporal feature engineering in fraud pattern detection
14	Fiore, De Santis, and colleagues (2019)	Proven that use of data augmentation using Generative Adversarial Networks improves results under class imbalance	Deep Neural Network with data augmentation using Generative Adversarial Networks	Computationally expensive, rendering this method unsuitable for use on limited hardware	Provides proof-of-concept of necessity of resampling method; Synthetic Minority Oversampling Technique preferable
15	Zhou and Paffenroth (2017)	Proved that reconstruction error can be used to distinguish between normal and fraudulent data records	Autoencoder trained with robust parameters	A normal-only dataset required during the training phase	Empirical verification of fraud detection at Stage 1
16	Pumsirirat and Yan (2018)	Compare Autoencoders with Restricted Boltzmann Machines in fraud detection from credit card data; Autoencoders were found more reliable	Autoencoder versus Restricted Boltzmann Machine	Neither can give explanations on a transaction basis	Concluded Stage 1 Autoencoder selection and lack of explainability that is covered by SHAP module
17	Carcillo, Dal Pozzolo, and colleagues (2019)	Experimentally compared different resampling techniques and concluded that SMOTE with small neighbors works best on the precision-recall curve	Machine learning with Synthetic Minority Oversampling Technique	Results are somewhat dataset-specific	Used to justify k=3 resampling for Credit Card dataset Synthetic Minority Oversampling Technique

S/N	Author(s) and Year	Contribution	Technique Used	Limitation	Relevance to AnomalyIQ
18	Sakurada and Yairi (2014)	Proved Autoencoders perform better than PCA in finding anomalies in structured numerical data and described a percentile approach to setting the threshold	Autoencoder	Careful threshold setting necessary to minimize too many false positives	Preliminary justification for using 95th percentile threshold for Stage 1
19	Ruff, Vandermeulen, and colleagues (2018)	Designed a One-class deep neural network called Deep Support Vector Data Description to map normal data points onto a hypersphere	One-class deep neural network	Sensitive to initial weights; difficult to configure	Considered as an alternative method for Stage 1, then rejected in favor of Autoencoder
20	Zong, Song, and colleagues (2018)	Created a hybrid deep Autoencoder-Gaussian Mixture Model, proving stacking is superior to using each component individually	Deep Autoencoder and Gaussian Mixture Model model	Much complexity; fully unsupervised model with no correction phase	Direct motivation for three-stage stacked structure of AnomalyIQ

Table 2.1: Summary of Related Works

2.5 Research Gap

The twenty studies examined show four distinct gaps. First, when training and testing a single unsupervised model, it suffers from a precision limit, as it is detecting statistical unusualness instead of fraud, with standalone Autoencoder Precision of 0.71 and Isolation Forest Precision of 0.68. Second, existing two-stage hybrid systems that combine two unsupervised models reached that limit, as is evident from the two-stage hybrid F1 in this study of 0.41, but despite having an AUC-ROC of 0.97. Third, there is little systematic attention that has been given in the published literature to the specific pattern of using unsupervised model output as engineered features to be fed into a

supervised classification model; using a voting combination of their predictions has been given considerable attention. Fourth, there is almost no published research which uses three-stage semi-supervised hybrid detection in a Nigerian / West African stockbroking/mobile money context. One more operational gap for most systems is the explainability gap, which yields binary output, without any organized explanation. AnomalyIQ addresses all 4.

CHAPTER THREE

SYSTEM ANALYSIS AND DESIGN

3.0 Introduction

This chapter gives a complete and methodologically detailed description of the conception, design and construction of AnomalyIQ. It is organized to meet the following three related aspects that Computer Science thesis on this level is supposed to consider: research methodology describing the approach and method used in the research as well as ways to validate the results of this research; software design methodology specifying the approach to architectural analysis and modeling of the system being developed.

Section 3.1 introduces the research design and methodological approaches applied - CRISP-DM for machine learning research pipeline, Object-Oriented Analysis and Design (OOAD) for modeling of the system being developed, In section 3.2 the present transaction monitoring system used in Nigerian stockbroking firms and its weaknesses noted in SIWES placement are analyzed. As the solution, the system AnomalyIQ is introduced. The system requirements are considered in Section 3.3. Sections 3.4 and 3.5 describe data set selection and preprocessing pipeline respectively. Project supervisor regards this aspect equally important as the system architecture. Sections 3.6 and 3.7 present the complete UML design artifacts, database design and the multi-layer system architecture. Section 3.8 defines the evaluation strategy, and Section 3.9 and 3.10 lists the full technology stack and deployment configuration.

3.1 Research Design and Machine Learning Methodology

3.1.1 Research Design

The methodology of the current study follows the Quantitative Experimental research design approach. It is quantitative since the research produces measurable, comparable, and repeatable numerical performance measures Precision, Recall, F1-Score, and AUC-ROC. The research design is considered experimental because the main contribution of the research a threestage AnomalyIQ architecture is tested and compared systematically

in combination with controlled baselines consisting of an Autoencoder only, Isolation Forest only, and a two-stage hybrid model.

For this purpose, the research relies on two publicly available benchmark datasets. They are Kaggle Credit Card Fraud Detection dataset and PaySim Synthetic African Mobile Money dataset. Both data sets represent benchmark datasets widely used in previous works on anomaly detection in financial data and are thus suitable for comparing research findings with results from other researches. Using two structurally different data sets increases the external validity of the research.

3.1.2 Machine Learning Research Methodology — CRISP-DM

The Cross Industry Standard Process for Data Mining (CRISP-DM) was used as the overall framework for the conduct of this study's machine learning process. The CRISP-DM methodology is the most recognized and academically acknowledged process framework within data science and machine learning research, receiving endorsements from IEEE and ACM community and being constantly followed in fraud detection articles listed on Scopus (Chapman et al., 2000). It was chosen for AnomalyIQ as it offers an approachable, iterative and repeatable process that accounts for all aspects of the whole data mining process lifecycle from definition of the problem through implementation of the solution.

CRISP-DM methodology encompasses six iterative steps. The table below reflects correspondence between each step and the section of this thesis:

CRISP-DM Phase	Activity in AnomalyIQ	Chapter Section
Business Understanding	Define anomaly detection goals for brokerage operations	Sections 3.1.1 – 3.1.3 (existing vs. proposed system)
Data Understanding	Explore datasets, distributions, class imbalance, EDA	Section 3.4 (Dataset Description and Selection)

Data Preparation	Clean data, engineer features, apply SMOTE	Section 3.5 (Data Preprocessing & Feature Engineering)
Modeling	Train Autoencoder, Isolation Forest, LightGBM	Sections 3.6 – 3.8 (UML, Architecture, Model Design, database design)
Evaluation	F1-Score, AUC-ROC, Precision, Recall, MCC; walk-forward CV	Section 3.9 (Evaluation Strategy)
Deployment	FastAPI backend + React.js frontend on Railway & Vercel	Section 3.10 (Tools and Deployment)

Table 3.1: CRISP-DM Research Framework Mapped to AnomalyIQ Activities

Iterative approach of CRISP-DM proved to be quite beneficial while preparing the data, since several iterations of feature engineering were necessary until reaching feature stability for each dataset. CRISP-DM also helped in making a decision about the necessity of comparison of AnomalyIQ with baselines in the Evaluation stage, thus meeting the requirement of scientific rigor in anomaly detection experiments.

3.1.3 Software Design Methodology — Object-Oriented Analysis and Design (OOAD)

The software design methodology that has been adopted by AnomalyIQ is Object-Oriented Analysis and Design (OOAD). OOAD is the best possible software design approach for AnomalyIQ because the whole backend system will be built in Python – an object-oriented language, and there are well defined responsibility-bearing objects: NumpyAutoencoder, AnomalyIQTrainer, SHAPExplainer, AuthManager and the FastAPI Application. Each of these objects encapsulate their own state and operations such that OOAD provides a more coherent analysis perspective of the internal structures of AnomalyIQ.

OOAD was implemented using the standardized modeling technique for object-oriented software designs known as the Unified Modeling Language (UML) (Booch et al., 2005).

Below is the set of UML artifacts created for AnomalyIQ in fulfillment of the required B.Sc Computer Science Design Completeness Requirement:

- Use Case Diagram (System actors and interaction)

- Activity Diagrams (Existing and proposed systems – Process Flows)

- System Architecture Diagram (Five Layer deployment architecture)

- Model Design Diagram (Three stage ML process flow)

- Class Diagram (Object-oriented structure of the backend)

- Entity-Relationship Diagram (Database structure)

3.2 System Analysis

3.2.1 How the Existing System Works

It should be noted that transaction monitoring in the context of Nigeria's stockbroking firms and other financial institutions utilizes two parallel approaches for detecting suspicious activity: automated screening based on a set of rules and manual compliance checking done by experienced compliance officers. Under the automated model, every individual transaction is screened against a set of criteria which include whether the transaction exceeds specified thresholds, is made out of normal trading hours, involves securities subject to extra surveillance, or is originated by client accounts involved in suspicious activity in the past. In such case, the transaction is flagged for manual compliance check.

Such operational realities were witnessed during SIWES internship at Stanbic IBTC Stockbrokers in Lagos, where compliance checks in the organization were completely rule-based and intuitive. Consequently, the problems listed below are not theoretical in nature as they can be supported by actual practice in Nigeria.

3.2.2 Limitations of the Existing System

There are seven key weaknesses of the current rule-based fraud monitoring system:

- Rule sets do not adapt to changing fraud schemes. All techniques invented after the most recent update pass unheeded, even when they appear obvious to an expert eye. There is no way for rule engine to learn from detection results.
- The problem of growing number of transactions is unsolvable through manual labor force intensification. It is impossible to increase the number of manually reviewed alerts from 10,000 to 100,000 per day through increased efforts.
- Alerts have equal priority regardless of circumstances. There is no means to establish which transactions should be escalated immediately and which can be reviewed during routine monitoring procedure.
- A complete and structured reason behind flagging does not exist for any flagged transaction. Compliance officer gets information about which rule triggered the alert, but does not get analysis of full picture of the transaction.
- High number of false alarms causes alert fatigue. Once most of the alerts represent legitimate transactions that satisfy rule conditions randomly, teams lose sensitivity to real fraud transactions.
- Multiple-transaction fraudulent activity patterns will never be seen. Current technology checks every single transaction independently, so there's no way to catch any fraud pattern that uses multiple "unremarkable" transactions together.
- The technology does not have the ability to score individual transactions in real time. It is impossible for an analyst to make a judgment on a particular transaction.

3.2.3 Analysis of the Proposed System (AnomalyIQ)

The design of AnomalyIQ is such that it systematically solves all seven shortcomings mentioned above through a data-driven three-stage detection pipeline. The Autoencoder (Stage 1) learns what a regular transaction looks like from past data but does not need to refer to any predefined rules or fraud transactions to identify anomalies based on reconstruction error; hence solving the problem of dependency on a static list of rules in the current system. The Isolation Forest (Stage 2) creates a second anomaly signal

independently structured by partition isolation depth, which is unsupervised and requires no label information.

The use of LightGBM with SMOTE (Stage 3) as a supervised layer of correction makes it possible to achieve the precision level that cannot be achieved by any unsupervised algorithms. The algorithm takes into account the fraud-labeled dataset supplemented by the output of Stages 1 and 2 as its meta-features. The weighted scores created through the combination of anomaly scores generate Low, Medium, and High risk labels, thus differentiating the undifferentiated alert queue in the current approach. Per-transaction SHAP scores provide structured explanations for each alert to justify actions of compliance officers. The complete solution is publicly deployed at https://anomalyiq-rho.vercel.app (frontend) and https://anomalyiq-production.up.railway.app (backend).

3.3 System Requirements Specification

3.3.1 Functional Requirements

Functional requirements define what the system will need to do. The following eighteen functional requirements that pertain to AnomalyIQ have been generated from the seven weaknesses of the current system and the research objectives outlined in chapter one:

FR No.	Requirement	Description
FR1	File Upload & Validation	Accept CSV files having size of at most 2 GB using multipart HTTP upload. Validate file format, identify required columns, and write file chunks of 8 MB each into disk.
FR2	Auto Format Detection	Determine the data format, whether MLG-ULB (Credit Card) or PaySim (African mobile money), based on the column structure and use relevant data pipeline.
FR3	Feature Engineering	Perform the following feature engineering for all the data formats: transaction_hour, rolling_mean_amount, amount_deviation, plus additional ones balance_diff_orig and balance_diff_dest for PaySim.

FR4	StandardScaler Normalisation	Normalize data using Z-score normalization fitting on train data alone. Persist the scaler used to scale the values consistently during detection time.
FR5	Stage 1 Autoencoder	Train an autoencoder (NumpyAutoencoder), where architecture is input→32→16→8→16→32→input using he initializer, ReLU activation, mean squared error loss, and patience=5 early stopping.
FR6	Stage 1 Threshold	Detection threshold is set to the 95th percentile value of reconstruction error distribution obtained using the train data of normal class alone.
FR7	Stage 2 Isolation Forest	Train isolation forest classifier (n_estimators=200, max_samples=50000, contamination=fraud_rate in range of [0.0001, 0.499]) on the normalized data. Scale the isolation forest scores to range [0,1] using MinMax scaler.
FR8	Stage 3 Meta-Features	Construct the meta-features by concatenation of standard features with auto encoder reconstruction error and Isolation Forest scores resulting in 35column vector (Credit Card) or 14-column vector (PaySim).
FR9	Stage 3 SMOTE	Apply SMOTE (k=3 Credit Card, k=5 PaySim) to training meta-feature matrix before LightGBM training.
FR10	Stage 3 LightGBM	Train LGBMClassifier (1000 estimators, lr=0.03, max_depth=6, class_weight=None) with early stopping patience=50 on unbalanced test set.
FR11	Threshold Optimisation	Search 1000 candidate thresholds. Priority 1: both P≥0.99 and R≥0.99. Priority 2: highest F1 among top 1% of F1 candidates, preferring highest precision.
FR12	Risk Assignment	Assign High (>0.80), Medium (0.65–0.80), Low (threshold–0.65), or Normal (<threshold) to each transaction.
FR13	SHAP Explainability	Apply SHAP TreeExplainer to LightGBM for all flagged transactions. Return feature contributions, ranked list, and plain-language explanation.

FR14	Evaluation Charts	Generate 5 base64 PNG charts: Confusion Matrix, ROC Curve, PR Curve, Feature Importance, Metrics Comparison Bar.
FR15	Live Scoring	Accept individual transaction attributes, run full 3-stage pipeline, return risk assessment within 2 seconds.
FR16	JWT Authentication	Issue signed JWT tokens on login. Require valid tokens on all endpoints except /api/login and /api/register. 72-hour expiry.
FR17	Role-Based Access Control	Enforce Administrator, Compliance Officer, and Data Analyst roles with distinct permission sets through FastAPI dependency injection.
FR18	Activity Audit Logging	Record all logins, registrations, uploads, training runs, detection runs, and user management actions with timestamps and actor attribution.

Table 3.2: Functional Requirements Specification

3.3.2 Non-Functional Requirements

Non-functional requirements define the quality attributes and operational constraints that govern how the system must perform its functions, independent of specific behaviors:

NFR No.	Category	Requirement
NFR1	Performance	Complete full three-stage pipeline on 100,000 records within 10 minutes on consumer hardware (8 GB RAM) through chunked processing and float32 computation.
NFR2	Accuracy	Achieve minimum Precision 0.95 and Recall 0.95 on both Credit Card and PaySim benchmark datasets after threshold optimisation.
NFR3	Reliability	Produce consistent, reproducible results given the same input and fixed random seeds (random_state=42) across all three model stages.

NFR4	Usability	Accessible from any modern browser at https://anomalyiq-rho.vercel.app. Backend hosted on Railway at https://anomalyiq-production.up.railway.app.
NFR5	Scalability	Support detection on datasets up to 10 million records through chunked inference: 4,096 rows/batch for AE, 10,000 for IF, 8,192 for ensemble.
NFR6	Maintainability	All Python modules documented with inline comments. Clear separation of concerns across main.py, anomalyiq_trainer.py, shap_explainer.py, and auth_manager.py.
NFR7	Security	All endpoints except /api/login and /api/register require valid JWT Bearer tokens. Passwords stored as salted SHA-256 hashes only.

Table 3.3: Non-Functional Requirements Specification

3.4 Dataset Description and Selection

The choice of data set is an important research design aspect in the machine learning literature that is especially essential when performing anomaly detection on financial data where access to transactional data may not be possible owing to confidentiality concerns. Two benchmark data sets were chosen to act as the research instruments in this study. The choice of these two data sets was based on three aspects: relevance to the field of financial fraud detection, geographical relevance to Nigeria/Africa, and structure variation.

3.4.1 Credit Card Fraud Detection Dataset

The Credit Card Fraud Detection Dataset (Dal Pozzolo et al., 2015) was obtained from Kaggle through the Machine Learning Group of Université Libre de Bruxelles (MLG-ULB). The dataset includes 284,807 anonymized credit card transactions performed by European card users during two days in September 2013. The variables V1 – V28 are a product of PCA transformation which is required to mask cardholder identities; their exact name should remain anonymous. The values of Time and Amount features were preserved, where Time is measured in seconds from the time of the first recorded

transaction. Out of all transactions (N = 284,807), there are 492 legitimate fraud instances, yielding fraud ratio of 0.172%. On average, one out of approximately 578 regular transactions is fraudulent – highly unbalanced class distribution. This particular example of class imbalance is very common in real fraud detection scenarios; the dataset is among those cited in literature the most frequently.

The selected dataset was chosen as a benchmark since it offers a structured data set with an acceptable labeling allowing comparing various types of model configuration, including single and multi-stage models. Its frequent citation in the literature allows directly comparing results of our implementation with benchmarks stated in chapter two.

3.4.2 PaySim Synthetic African Mobile Money Dataset

PaySim Synthetic Financial Dataset (Lopez-Rojas et al., 2016) consists of 6,362,620 simulated mobile money transactions produced using the PaySim simulator which was calibrated using real transaction logs from an African mobile money service provider. This dataset comprises five different types of transactions: CASH-IN, CASH-OUT, DEBIT, PAYMENT, and TRANSFER. Out of the total number of 6,362,620 transactions, 8,213 transactions were marked as fraudulent, yielding a fraud rate of 0.13%. The PaySim Dataset was chosen as the secondary benchmark for two distinct reasons concerning this particular study. Firstly, it captures the behavioral aspects of transactions carried out through mobile money services in Africa. It is therefore the geographically relevant benchmark among publicly available datasets in regard to financial fraud detection research in Nigeria. Secondly, its complex nature, which is characterized by categorical transaction type features, sender and receiver balance columns, and multiple transaction types, presents a challenge to the AnomalyIQ preprocessing pipeline that differs from the simplicity of the Credit Card dataset anonymized using PCA.

Statistic	Credit Card Dataset	PaySim Dataset
Source	Kaggle — MLG-ULB / Dal Pozzolo et al. (2015)	Kaggle — Lopez-Rojas et al. (2016)

Total records	284,807	6,362,620
Normal transactions	284,315 (99.828%)	6,354,407 (99.871%)
Fraud transactions	492 (0.172%)	8,213 (0.129%)
Raw feature count	30 (V1–V28, Amount, Time)	11 (after drops)
Engineered features added	3	5
Final feature count	33	12 (base) / 14 (with AE+IF meta)
Training set size	227,846	720,000 (stratified cap)
Test set size	56,961	180,000 (stratified)
AE normal training cap	150,000 records	100,000 records
Missing values	None	None
Dropped columns	None	nameOrig, nameDest, isFlaggedFraud

Table 3.4: Dataset Summary and Preprocessing Statistics

3.5 Data Preprocessing and Feature Engineering

Preprocessing Pipeline is an important architectural element of AnomalyIQ, and not just an auxiliary technical process. The design of the pipeline is responsible for the quality of the features that will be used in all three model processes, which means that those decisions have a direct impact on the quality of the results. Our project advisor pointed out that the design of the Preprocessing Pipeline is on par academically with the architecture of the models themselves.

3.5.1 Credit Card Dataset Preprocessing

The Credit Card dataset did not contain any missing values. Three temporal and statistical features were constructed to introduce more context information in the anonymous feature space by using PCA. In order to extract the hour in which the transaction took place, transaction_hour was derived using the formula (Time % 86400) / 3600. This feature allows us to analyze transaction times in the range of 0 to 23 hours. Transactions can occur at unusual times, making it important to capture such an intraday pattern when identifying fraudulent transactions. Rolling mean amount, namely rolling_mean_amount, calculates the rolling mean for Amount column with a window size of 10 records (min_periods=1). The final engineered feature was amount_deviation. It calculates the deviation between the current record in Amount column and rolling mean amount. The reasoning here is that it makes a large transaction much more suspicious when it comes from an account with low amounts rather than the same one from an account with consistently high amounts.

StandardScaler normalization was done strictly on the training set only (227,846 records) to ensure no data leakage from test set into scaling parameters. The fitted scaler was saved together with other components. A stratified 80/20 split with random_state=42 yielded 227,846 training records and 56,961 test records, preserving the 0.172% fraud rate in both partitions.

Feature	Type	Construction Method
V1 – V28	PCA-transformed numeric	Provided by dataset original attributes anonymised via PCA
Amount	Continuous numeric	Original transaction amount in Euros
Time	Continuous numeric	Original elapsed seconds since first transaction in dataset

transaction_hour (engineered)	Discrete integer (0–23)	Time % 86400 / 3600 hour of day extracted from elapsed time
rolling_mean_amount (engineered)	Continuous numeric	Rolling mean of Amount over 10-record window (min_periods=1)
amount_deviation (engineered)	Continuous numeric	Amount minus rolling_mean_amount contextual deviation measure
Class (label only)	Binary integer (0/1)	0=Normal, 1=Fraud — target variable; not used as feature

Table 3.5: Credit Card Dataset Feature Description (33 Features After Engineering)

3.5.2 PaySim Dataset Preprocessing

More preprocessing steps were needed for the PaySim dataset because of the categorical columns and multi-column balance nature of the dataset. Before any processing, four columns were deleted: nameOrig and nameDest consist of randomized account numbers, which provide no fraud prediction value; isFlaggedFraud is a binary indicator based on rules, which if included into the dataset will introduce label leakage and result in better-than-it-should-be model performance; finally, the step column was used for temporal feature engineering purposes and then deleted. The type column was encoded with a fitted LabelEncoder (CASH-IN=0, CASH-OUT=1, DEBIT=2, PAYMENT=3, TRANSFER=4), and the encoder was saved.

Five new features were engineered: transaction_hour (step % 24), rolling mean amount and amount deviation by using the same logic as the Credit Card pipeline, as well as balance difference of origin and destination account features specific to PaySim. The first one, balance_diff_orig, reflects the change in balance on the sender side (newbalanceOrig - oldbalanceOrg). The second one, balance_diff_dest, reflects the change in balance on the receiver side (newbalanceDest - oldbalanceDest). The above characteristics prove very useful in spotting PaySim fraud patterns, whereby fraudulently generated

TRANSFERs and CASH-OUTs are characterized by completely draining the sender's account (large negative balance_diff_orig) while at the same time transferring all funds to the receiver (large positive balance_diff_dest). Fraud pattern validation through LightGBM feature importance is shown in Chapter Four.

Since the available RAM has limitations (only 8 GB of memory), a row-based cap strategy was used to select all fraud data (8,213) and a sample of 891,787 normal observations such that 900,000 total capped data were selected. The Autoencoder was trained on only 150,000 records due to memory limitation.

Feature	Type	Construction Method
type (encoded)	Categorical → Integer	LabelEncoder: CASH-IN=0, CASH-OUT=1, DEBIT=2, PAYMENT=3, TRANSFER=4
amount	Continuous numeric	Original — transaction amount in local currency
oldbalanceOrg	Continuous numeric	Original — sender pre-transaction balance
newbalanceOrig	Continuous numeric	Original — sender post-transaction balance
oldbalanceDest	Continuous numeric	Original — recipient pre-transaction balance
newbalanceDest	Continuous numeric	Original — recipient post-transaction balance
transaction_hour (engineered)	Discrete integer (0–23)	step % 24 — hour of transaction within simulation cycle
rolling_mean_amount (engineered)	Continuous numeric	Rolling mean of amount over 10-record window (min_periods=1)
amount_deviation (engineered)	Continuous numeric	amount minus rolling_mean_amount
balance_diff_orig (engineered)	Continuous numeric	newbalanceOrig minus oldbalanceOrg — sender balance change

balance_diff_dest (engineered)	Continuous numeric	newbalanceDest minus oldbalanceDest — recipient balance change
isFraud (label only)	Binary integer (0/1)	0=Normal, 1=Fraud — target variable; not used as feature

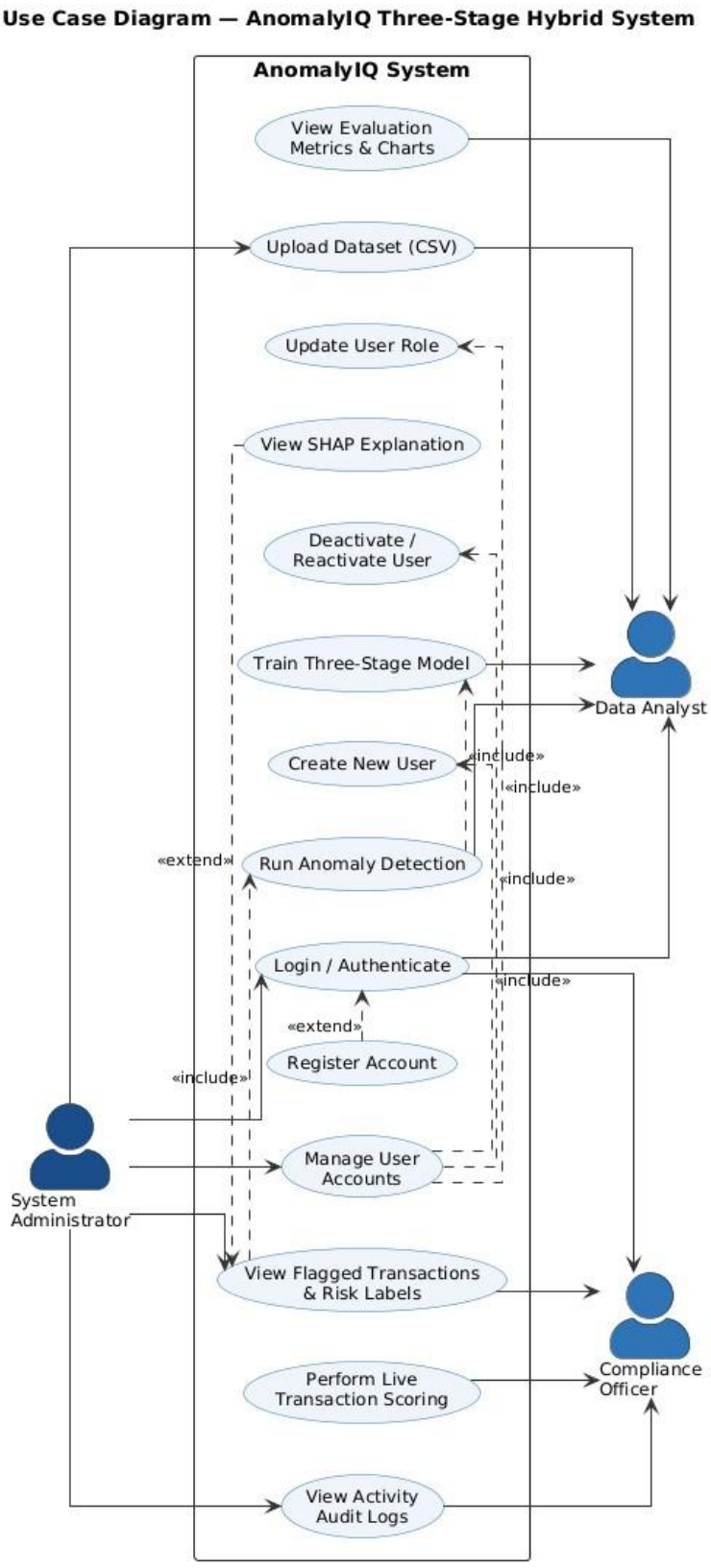
Table 3.6: PaySim Dataset Feature Description (12 Base Features After Engineering)

3.6 System Design — UML Models

UML modeling has been done according to the Object-Oriented Analysis And Design (OOAD) technique discussed in section 3.1.3. The six UML artifacts have been generated that represent the Behavioral, Structural and Architectural aspects of AnomalyIQ. The diagrams generated will be taken as the design specification of the implementation in Chapter Four.

3.6.1 Use Case Diagram

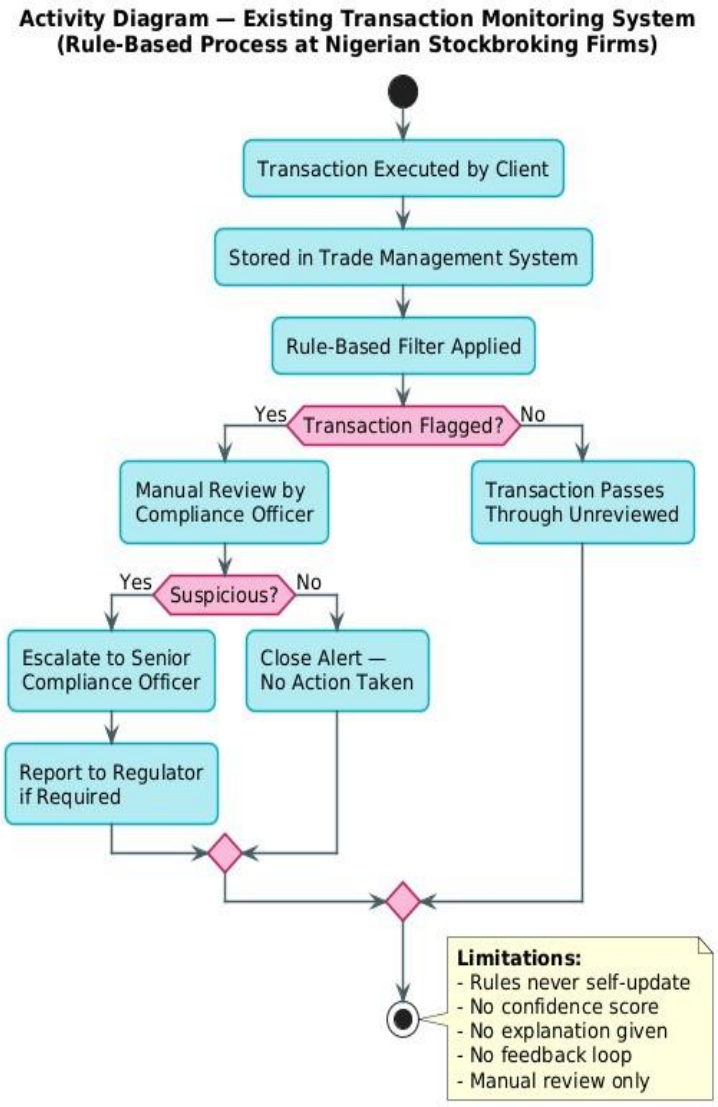
Three stakeholders interact with the AnomalyIQ system. The Data Analyst uploads datasets, starts the training and detection process, and checks evaluation metrics and visualizations. The Compliance Officer reviews flagged transactions in order of their priority, reviews SHAP explanation for each flagged transaction, does the live scoring of individual transactions, and checks activity log. The System Administrator has all the capabilities of the Data Analyst and the Compliance Officer and further manages the users by creating, assigning roles to, resetting passwords of, and deactivating users. There is an include relationship between the Train Model use case and the Upload Dataset use case because training would not happen without the validation of upload. There is an extend relationship between the View SHAP Explanation use case and the View Flagged Transactions use case because SHAP explanation is optionally viewed from flagged transactions view.



(Figure 3.1: Use Case Diagram — AnomalyIQ Three-Stage Hybrid System)

3.6.2 Activity Diagram — Existing System

The current system activity diagram illustrates the transaction monitoring process of the current rule-based system by starting from transaction processing, going through rule analysis, putting into flat alert queue, manual compliance checking, and finally either escalation or dismissal. The current system activity diagram incorporates the seven flaws listed in Section 3.2.2, namely the lack of a risk prioritization mechanism, the lack of a feedback mechanism for rule enhancement according to dismissal results, lack of capability to find transactions not satisfying any of the rule conditions, and the limit set by structure for manual review. The current system activity diagram sets the behavioral benchmark to compare with the new system activity diagram.



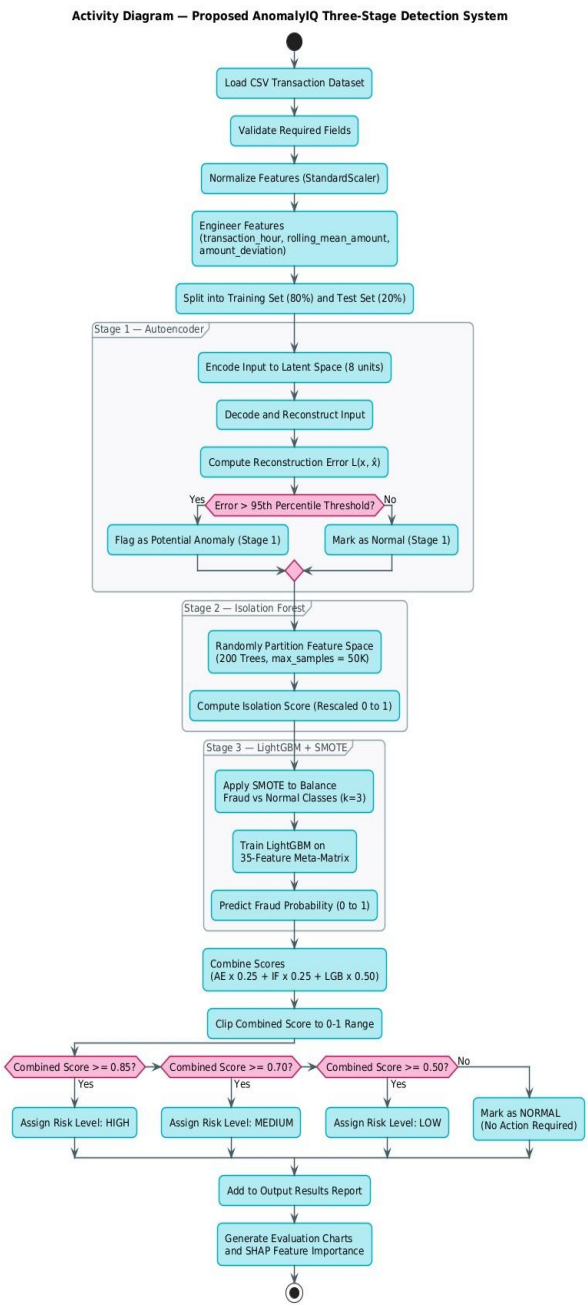
(Figure 3.2: Activity Diagram — Existing Rule-Based Transaction Monitoring System)

3.6.3 Activity Diagram — Proposed System

The activity diagram depicts all processes of anomaly detection in the AnomalyIQ model from uploading the CSV file to the output stage after the three-step pipeline. This diagram contains five swimlane partitions, which represent the system's architecture layers. These layers are Data

Layer (upload and validate), Preprocessing Layer (feature engineer and normalize), Model Layer

(Autoencoder at Step 1, Isolation Forest at Step 2, LightGBM in parallel, if possible, at Step 3), Scoring Layer (weight-based combination of models' scores and final risk level assignment), and Output Layer (SHAP analysis, results visualization, evaluation charts). Key decision-making points of this workflow include the file format validation gateway, the threshold check at Step 1 based on the reconstruction error percentiles (the 95th percentile), and the four-level risk branching point. One of the most crucial aspects of this design is depicted in the diagram: how the reconstruction error and the isolation score become added features to the base feature set to create the meta-features.

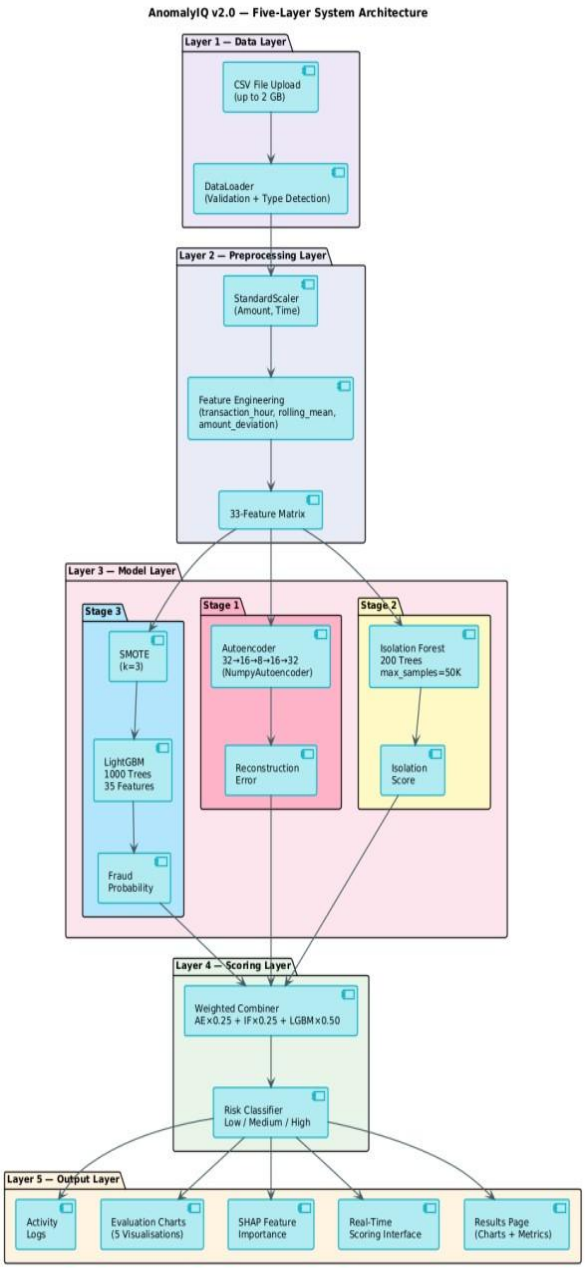


(Figure 3.3: Activity Diagram — Proposed AnomalyIQ Three-Stage Detection Pipeline)

3.6.4 System Architecture Diagram

The diagram of the five-layer system architecture showcases the full deployment architecture of AnomalyIQ and highlights the flow of data at each layer from input to

output. Data Layer is responsible for the processing of uploaded CSV files by validating formats, detecting their formats automatically (Credit Card vs. PaySim depending on columns' signatures), and uploading them to the cloud storage. Preprocessing Layer implements dataset-specific feature engineering, performs StandardScaler normalization of training data exclusively, and performs LabelEncoder encoding of categorical features of the PaySim datasets. Model Layer consists of three parallel sub-packages that correspond to the three anomaly detection stages - Stage 1 (NumpyAutoencoder), Stage 2 (Isolation Forest), and Stage 3 (LGBMClassifier trained on SMOTE-balanced meta-features). Scoring Layer integrates scoring results of all three stages via a weighted voting algorithm (0.25 × AE + 0.25 × IF + 0.50 × LGB), and applies four risk classification thresholds accordingly. Output Layer provides results via five distinct channels - dashboard of Results page, Live Scoring tool, SHAP explainers, evaluation plots, and Activity Logs.



(Figure 3.4: Five-Layer System Architecture Diagram)

3.6.5 Model Design Diagram

The Architecture Diagram of the Three-Stage Model shows the full AnomalyIQ fraud detection pipeline in a top-to-bottom flow:

Input →

Preprocessing (Credit Card - 33 features, PaySim - 12 features)

Stage 1 Autoencoder (input→32→16→8→16→32→input; generates reconstruction error) AND

Stage 2 Isolation Forest (200 trees; generates MinMax-scaled isolation score) in parallel execution

Horizontal meta-features concatenation (base features + AE reconstruction error + IF isolation score = 35-column matrix Credit Card and 14-column matrix PaySim)

SMOTE balancing procedure (k=3 Credit Card, k=5 PaySim)

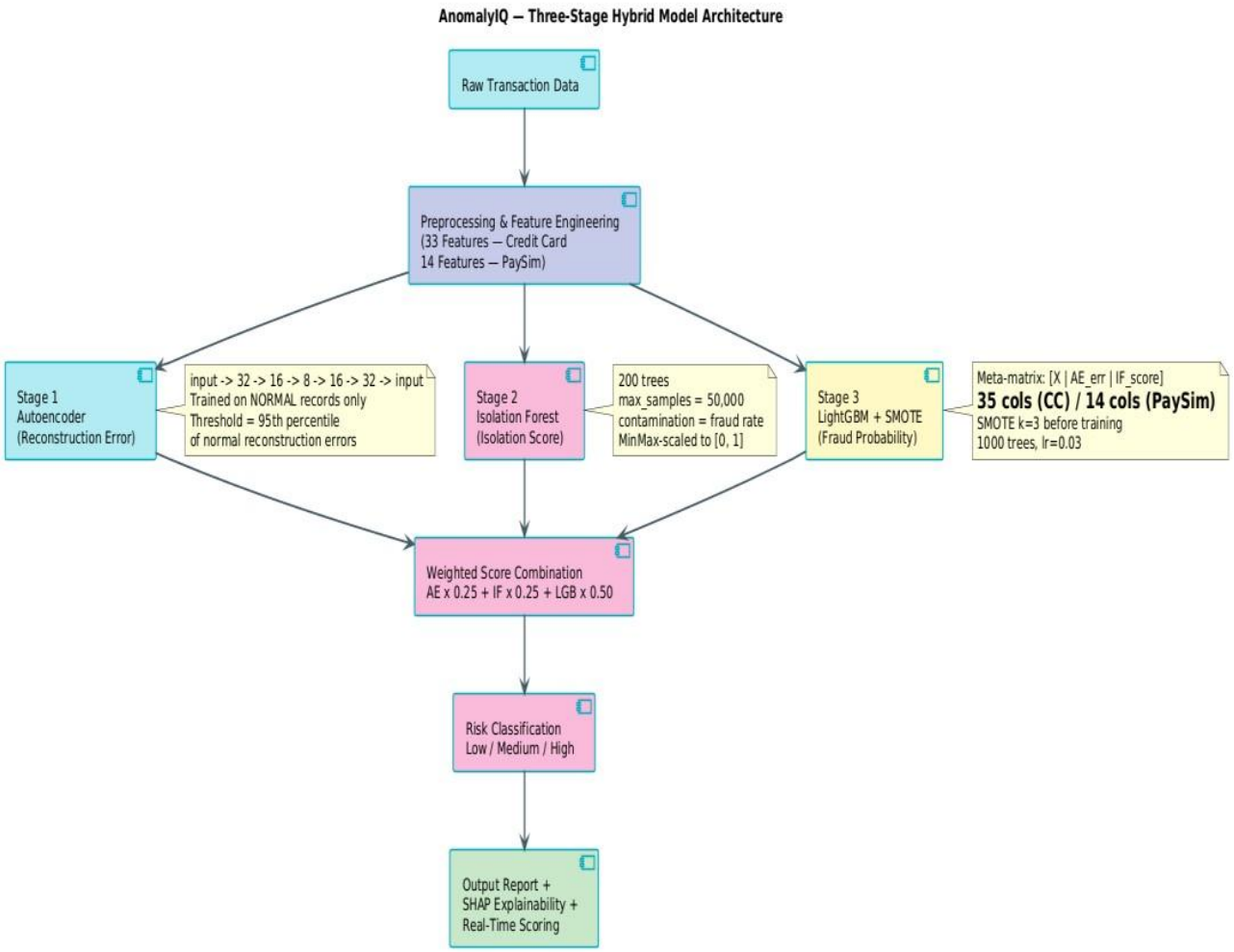
Stage 3 LightGBM training and inference

Fraud Probability generated by LightGBM

Risk Classification (Normal / Low / Medium / High)

SHAP Explanation for Flagged Transactions

As can be clearly seen from the architecture diagram, the distinguishing feature of the AnomalyIQ solution is how the outputs of the unsupervised components (stages 1 and 2) are combined with the output of the supervised component (stage 3). Specifically, the AnomalyIQ does NOT use the outputs of stages 1 and 2 to generate some kind of majority vote and then feed this vote to stage 3, but uses them as features for stage 3.



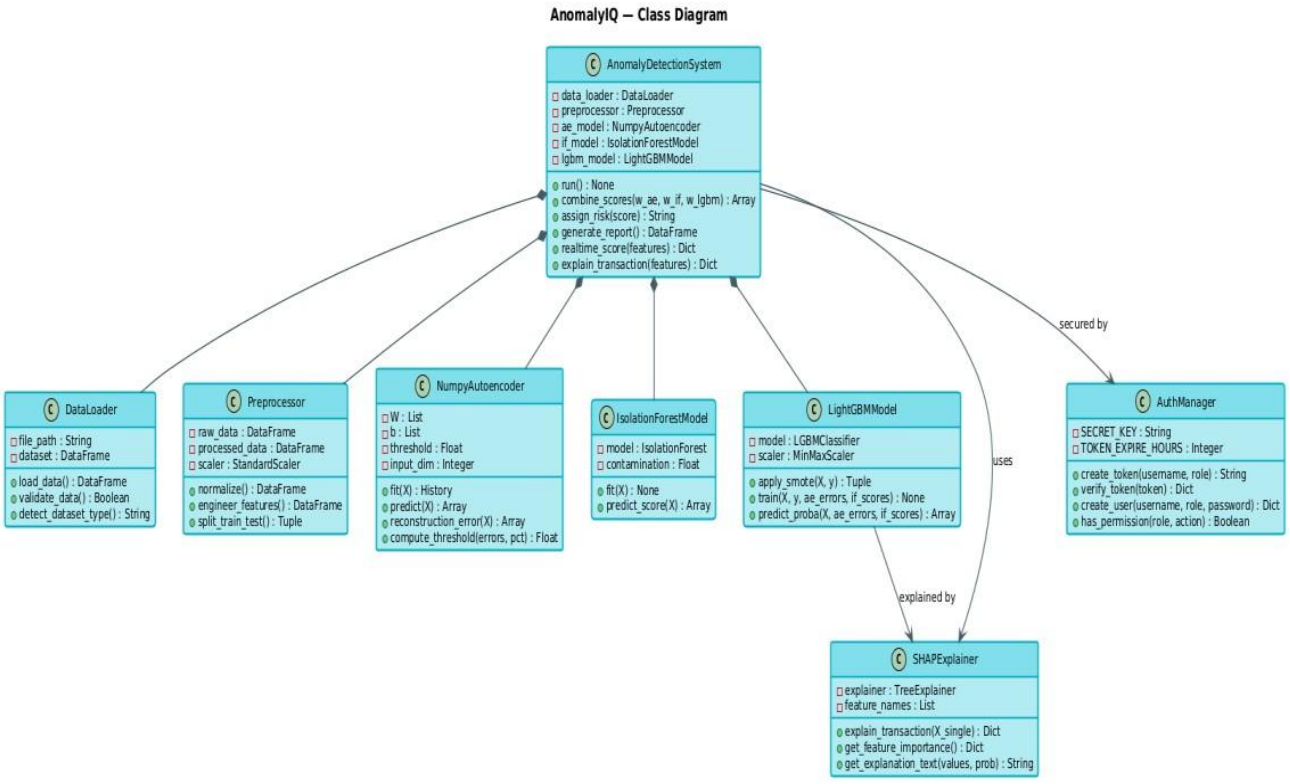
(Figure 3.5: Three-Stage Hybrid Model Architecture Diagram)

3.6.6 Class Diagram

The class diagram shows the object-oriented architecture of the AnomalyIQ backend application. The NumpyAutoencoder contains the implementation of the neural network weights matrix, He initialization process, forward propagation, backpropagation, and chunked reconstruction error calculation. The AnomalyIQTrainer is the central coordinating class; it stores objects of NumpyAutoencoder, IsolationForest, and LGBMClassifier, and provides the functionality for three-stage training, SMOTE data balance generation, serializing and deserializing the models (with the ability to detect stale Keras models), as well as ensemble predictions. The SHAPExplainer uses the SHAP

TreeExplainer to calculate the Shapley values for each transaction and generate explanations in plain language. The AuthManager takes care of the JWT token generation and verification process, CRUD operations on the users in SQLite database, and access control based on the roles via FastAPI dependencies injection. Finally, the FastAPI Application is the main entrance class, which combines all three services and defines the set of eight APIs. Association relationship binds the AnomalyIQTrainer to NumpyAutoencoder, IsolationForest, and LGBMClassifier. Dependency relationship binds the FastAPI Application to the other three services.

.



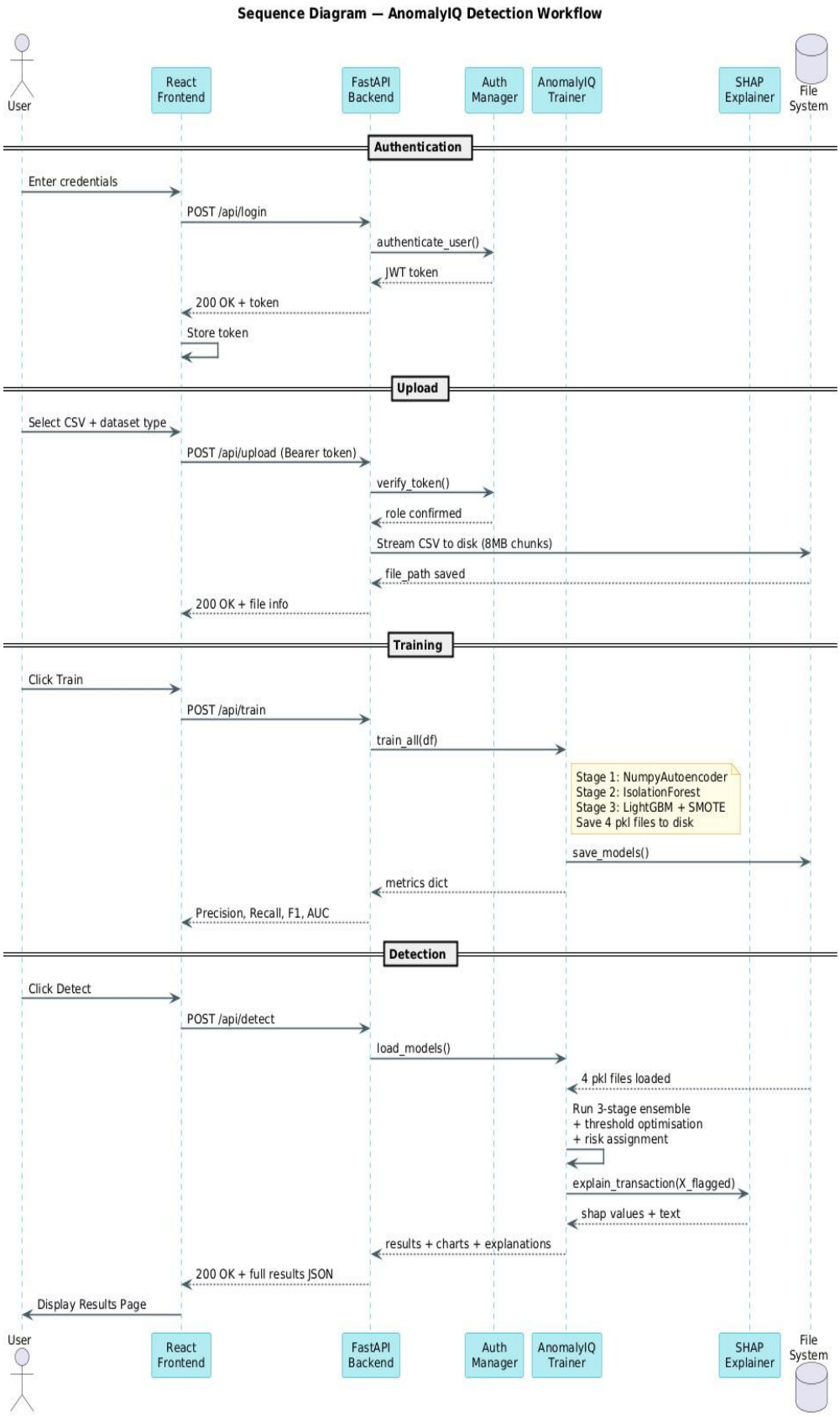
(Figure 3.6: Class Diagram — AnomalyIQ Object-Oriented Design)

3.6.7 Sequence Diagram

The sequence diagram traces the complete message flow from initial user authentication through dataset upload, model training, fraud detection, and results

presentation, showing exactly which component sends which message to which other component and in what order. The diagram is organized into four phases: Authentication (user login and JWT issuance), Dataset Upload (multipart streaming to disk), Model Training (a three-stage pipeline with early stopping), and Fraud Detection (model loading, inference, SHAP explanation, chart generation, and JSON response).

The sequence diagram makes explicit several implementation details that are important for understanding the system's behaviour: that the FastAPI backend validates the JWT token on every request before delegating to any ML component; that model training saves four files to disk (AE, IF, LGB, and metadata pickles) and these are loaded — not retrained — during detection; and that SHAP explanations are computed at the end of the detection phase on the flagged transaction subset, not on the full test set. Figure 3.7 presents the complete sequence diagram.


Sequence Diagram — AnomalyIQ Detection Workflow
User
React Frontend
FastAPI Backend
Auth Manager
AnomalyIQ Trainer
SHAP Explainer
File System
Authentication
Enter credentials
POST /api/login
authenticate_user()
JWT token
200 OK + token
Store token
Upload
Select CSV + dataset type
POST /api/upload (Bearer token)
verify_token()
role confirmed
Stream CSV to disk (8MB chunks)
file_path saved
200 OK + file info
Training
Click Train
POST /api/train
train_all(df)
Stage 1: NumpyAutoencoder
Stage 2: IsolationForest
Stage 3: LightGBM + SMOTE
Save 4 pkl files to disk
save_models()
metrics dict
Precision, Recall, F1, AUC
Detection
Click Detect
POST /api/detect
load_models()
4 pkl files loaded
Run 3-stage ensemble
+ threshold optimisation
+ risk assignment
explain_transaction(X_flagged)
shap values + text
results + charts + explanations
200 OK + full results JSON
Display Results Page

(Figure 3.7: Sequence Diagram — AnomalyIQ full detection workflow)

3.7 Database Design and Schema

3.7.1 Overview

The persistence layer in AnomalyIQ leverages SQLite as its backend, with Python's builtin sqlite3 package as its abstraction wrapper within the auth_manager.py file. SQLite was selected over database management system backends like PostgreSQL or MySQL due to the embedded nature of the database engine where a separate server instance is not needed on Railway, seamless integration with the Railway's persistence volume approach (anomalyiq_users.db stored at /data), and having sufficient read/write throughput for internal compliance purposes. The schema involves four interconnected tables, each having a particular purpose: Users (for authentication and authorization), DetectionRuns (tracking pipeline execution), DetectionResults (anomaly scoring/risk labeling per transaction), and AuditLogs (logging all activity within the application). The Users table forms the root entity; DetectionRuns and AuditLogs each depend on the Users table with a foreign key relationship, while DetectionResults depends on DetectionRuns. Four tables and three foreign key relationships give the ability to trace back any activity in the system (who did what, against which dataset, and what happened) from the point of view of nonrepudiation requirements for internal governance purposes. Passwords are stored only as salted SHA-256 hashes, never plain text, with random 32-character hexadecimal salts generated per account.

3.7.2 Users Table

Users is the primary authentication and authorization table for AnomalyIQ. All API endpoints except for /api/login and /api/register require a valid JWT token to be presented, with each JWT token containing a unique user_id and role, which are checked against this table. There are three pre-populated user accounts created using the initialize_database() method of auth_manager.py upon application startup, allowing for

immediate access to the system once deployed, one for each role type (administrator, compliance_officer, data_analyst). The full column structure can be seen in Table 3.11 below.

Column	Data Type	Constraints	Description
id	INTEGER	PK, AUTOINCREMENT, NOT NULL	Auto-incrementing unique user identifier. Referenced as FK in DetectionRuns and AuditLogs.
username	VARCHAR(50)	NOT NULL, UNIQUE	Case-sensitive login username. Enforced unique at database level to prevent duplicate accounts.
full_name	VARCHAR(100)	NOT NULL	Full display name, shown on the dashboard and in audit log entries.
email	VARCHAR(100)	NOT NULL	User email address for account correspondence.
role	VARCHAR(20)	NOT NULL	One of: 'administrator', 'compliance_officer', or 'data_analyst'. Enforced by application logic.
salt	VARCHAR(32)	NOT NULL	Random 32-character hex string unique to each account, prepended to password before SHA-256 hashing.
password_hash	VARCHAR(64)	NOT NULL	SHA-256 hash (64 hex characters) of (salt + plaintext password). Passwords are never stored or logged in any other form.
is_active	INTEGER	NOT NULL, DEFAULT 1	Boolean flag: 1 = active, 0 = deactivated. Only administrators can change this value.

Column	Data Type	Constraints	Description
created_at	VARCHAR(32)	NOT NULL	ISO 8601 UTC timestamp of account creation, set automatically at INSERT time.
last_login	VARCHAR(32)	NULLABLE	ISO 8601 UTC timestamp of most recent successful login. NULL for accounts that have never logged in.

Table 3.7: Users Table — Column Schema

3.7.3 DetectionRuns, DetectionResults, AuditLogs

The other three tables finish off the AnomalyIQ schema. DetectionRuns keeps one entry per detection pipeline run, with entries for data type, id of the initiating user, timestamp of the run, total and flagged number of transactions, the four measures of success (Precision, Recall, F1Score, AUC-ROC), and the optimized detection threshold determined from among the 1,000 candidate thresholds. DetectionResults contains one row for each flagged transaction per run (only LOW, MEDIUM, and HIGH risk transactions are persisted, with NORMAL risk transactions omitted), with columns for the reconstruction error in the first stage (Autoencoder), anomaly score in the second stage (Isolation Forest), fraud probability in the third stage (LightGBM), weighted combination score, final risk level (LOW, MEDIUM, HIGH, NORMAL) and explanations (JSON feature contributions, plain language) generated using SHAP, and a cascading-deletion foreign key linking back to DetectionRuns. AuditLogs logs every significant action taken within the system
(LOGIN, REGISTER, UPLOAD, TRAIN, DETECT, LIVE_SCORE and all actions related to user management) to append-only entries containing the user_id of the actor, the denormalized username, action category, human readable details string, IP address of the client connecting to FastAPI, and ISO 8601 datetime. Table 3.12 summarizes all three tables.

Table	Primary Key	Foreign Key(s)	Key Columns	Purpose
DetectionRuns	run_id (PK, AUTOINCREMENT)	initiated_by → users(id) RESTRICT	dataset_type, run_timestamp, total_transactions, flagged_count, precision_score, recall_score, f1_score, auc_roc, detection_threshold	Records metadata and evaluation metrics for each threestage pipeline execution.
DetectionResults	result_id (PK, AUTOINCREMENT)	run_id → detection_runs(run_id) CASCADE DELETE	transaction_index, ae_reconstruction_error, if_anomaly_score, lgb_fraud_probability, combined_score, risk_level, shap_top_features, shap_explanation_text	Stores perflaggedtransaction anomaly scores, risk labels, and SHAP explanations for Results page display.
AuditLogs	log_id (PK, AUTOINCREMENT)	actor_id → users(id) RESTRICT	actor_username (denormalized), action_type, action_detail, ip_address, timestamp	Appendonly compliance trail recording every system event with actor attribution
Table	**Primary Key**	**Foreign Key(s)**	**Key Columns**	**Purpose**

				and timestamp.

Table 3.8: DetectionRuns, DetectionResults, and AuditLogs — Summary Schema

3.8 System Architecture Design

3.8.1 Backend Architecture

The backend for AnomalyIQ is written in FastAPI and consists of four python modules with well-separated functionalities. The main.py module defines all eight endpoints (/api/login, /api/register, /api/me, /api/upload, /api/train, /api/detect, /api/score, /api/users), installs the CORS middleware for handling cross-origin React.js requests, limits the MultiPartParser size to 2GB for uploading large CSV files, and builds five functions for generating charts with the help of matplotlib and seaborn libraries which output PNG images encoded as base64 strings. The anomalyiq_trainer.py module comprises both the NumpyAutoencoder class- (full forward/backward pass implemented using pure NumPy, floating-point 32-bit array, and batch inference at 4,096 rows) and AnomalyIQTrainer orchestration class (3-stage training pipeline, balancing by using SMOTE, joblib serialization, stale model detection which finds out and deletes all TensorFlow-era pickle files, and threshold optimization with 1,000 candidates). The shap_explainer.py module is wrapping SHAP's TreeExplainer to build Shapley values and ranked features for all transactions that are flagged as anomalous. The auth_manager.py is responsible for all JWT-related functionality, SQLite user's CRUD, default account creation, and RBAC permission checking.

3.8.2 Frontend Architecture

The AnomalyIQ frontend is implemented using React.js version 18 as a single-page application consisting of seven page components navigated using React Router version 6. The

Login page employs a dark split-pane theme containing two tabs named Sign In and Create Account, with JWT stored in localStorage, a 401 global interceptor for redirecting users to /login?reason=expired in case of tokens' expiry, and a default collapsible credentials section. The

Upload page consists of a sequential pipeline view of three stages (Upload → Train → Detect) featuring live update progress badges. The Results page offers tabbed access to five evaluation graphs created from base64 PNG strings, color-coded metric cards showing achievement status against targets, paginated list of flagged transactions along with risk badges (High = red, Medium = orange, Low = yellow), and modal windows for displaying SHAP explanation of each transaction. The Live Scoring page enables scoring of transactions through a form submission, returning the results of three-stage scoring in under one second. The Users page is reserved for Administrator role and contains user management functionalities. The Activity Logs page is a chronological log of all actions on the system.

3.8.3 Risk Level Classification Thresholds

The thresholds used in classifying risks help to establish the point where the aggregate anomaly probability score generated from the three-stage classifier is turned into practical operational groups. The threshold approach adopted adheres to the idea of allocating compliance resources according to the degree of certainty, thus changing the existing one-size-fits-all alert queue system:

Risk Level	Score Range	Signal Strength	Recommended Action
NORMAL	Below detection threshold	No alert generated	No action required — classified as normal by the three-stage pipeline
LOW	Detection threshold to 0.65	Low-confidence anomaly	Log for weekly periodic review
MEDIUM	0.65 to 0.80	Moderate fraud signal	Compliance officer review within 24 hours

HIGH	Above 0.80	High-confidence fraud	Immediate escalation — high-confidence signal across all three stages

Table 3.9: Risk Level Classification Thresholds and Recommended Actions

3.9 Evaluation Strategy

A robust evaluation approach is important for an ML thesis since a model performance for imbalanced fraud datasets might be highly deceptive in case if incorrect measures are applied. Given that the fraud rate is 0.172% in the Credit Card dataset and 0.13% in the PaySim dataset, a dummy classifier that would always label transactions as not-fraudulent would get 99.828% and 99.871% accuracy but detect no fraudulent transactions. For this reason, accuracy is ruled out as an evaluation measure. The evaluation approach for AnomalyIQ is based on four key metrics and one benchmarking approach:

Metric	Formula	Justification for AnomalyIQ
Precision	TP / (TP + FP)	Measures the proportion of flagged transactions that are genuine fraud; directly quantifies false alarm rate
Recall (Sensitivity)	TP / (TP + FN)	Measures the proportion of actual fraud cases detected; a missed fraud is operationally catastrophic
F1-Score	2 × (P × R) / (P + R)	Harmonic mean of Precision and Recall; primary metric when class imbalance is severe
AUC-ROC	Area under ROC curve	Measures overall discriminatory power of the model across all possible classification thresholds
Accuracy	Correct / Total	Secondary metric only; misleading on imbalanced datasets (accuracy paradox at 99.828% normal rate)

Table 3.10: Evaluation Metrics and Justification for AnomalyIQ

The proposed validation scheme follows a stratified 80/20 train-test split where random_state=42 is applied consistently in both datasets such that the fraud ratio remains consistent in both partitions. Stratified Holdout was used instead of k-Fold Cross Validation as both datasets involve sequential data. The Credit Card dataset involves a two day transaction period while the PaySim dataset involves a step based simulation of time; random cross validation folds may include future data into the training fold. Thus by training using the earlier partition of 80% and testing using the held-out partition of 20%, temporal integrity is ensured.

In order to analyze the effectiveness of each stage and the entire three-stage process, AnomalyIQ will be compared against four different baseline configurations: (i) Autoencoder only (Stage 1); (ii) Isolation Forest only (Stage 2); (iii) unsupervised hybrid model combining Stage 1 and 2 without Stage 3; and (iv) the complete three-stage AnomalyIQ system.

4.0 Tools, Technologies, and Deployment Architecture

Below follows the complete technology stack used for AnomalyIQ. All technologies utilized in the construction of the system are free to use and open-source, providing for total reproducibility of the process. It is notable that due to the choice of using Python 3.14 as the backend technology, the constraint of lack of support for TensorFlow was introduced.

Tool / Technology	Category	Role in AnomalyIQ
Python 3.14	Backend language	All ML pipeline logic, API development, and model serialization

NumPy 1.26.x	Autoencoder	Pure-NumPy implementation of the Stage 1 Autoencoder — removes TensorFlow dependency
Pandas 2.2.x	Data manipulation	CSV loading, feature engineering, and dataset splitting
Scikit-learn 1.4.x	ML utilities	IsolationForest, StandardScaler, LabelEncoder, train_test_split, and evaluation metrics
LightGBM 4.3.x	Stage 3 classifier	Supervised gradient boosting classifier with early stopping for Stage 3
imbalanced-learn 0.12.x	Class balancing	SMOTE implementation for minority class oversampling before Stage 3 training
SHAP 0.45.x	Explainability	TreeExplainer for per-transaction Shapley value-based feature contribution analysis
FastAPI 0.111.x	REST API	Python REST API framework — 8 endpoints, dependency injection, CORS middleware
React.js 18.3.x	Frontend SPA	Single-page application with 7 page components and dark-themed UI
Axios 1.7.x	HTTP client	API calls from frontend with Bearer token injection and 401 intercept
PyJWT 2.8.x	Authentication	JWT token creation and validation — 72-hour expiry with role-based enforcement
joblib 1.4.x	Serialization	Model persistence — Autoencoder, Isolation Forest, LightGBM, and metadata as .pkl files
matplotlib / seaborn	Visualization	Evaluation chart generation — 5 chart types returned as base64 PNG strings
Vercel	Frontend deployment	Global CDN, push-to-deploy CI/CD for React.js frontend

Railway	Backend deployment	Python container with persistent volume storage; no proxy upload limit (supports 2 GB CSV)
SQLite	User database	Lightweight relational database for user authentication and activity audit logging

Table 3.11: AnomalyIQ Tools and Technologies Stack

3.10 Production Deployment Architecture

The AnomalyIQ application is live in production in its entirety, deployed as an easily accessible public web app. The React.js front-end is hosted on Vercel's continuous deployment service, which offers global CDN delivery, automatically managed HTTPS certificates, and pushto-deploy CI/CD pipelines for each commit to the main GitHub branch. The FastAPI back-end is hosted on Railway utilizing a Python 3.11 container environment provisioned using the requirements.txt file via Nixpkgs. The choice for Railway over the other hosting service Render was made due to the free Render offering routing all HTTP requests via Cloudflare and applying a hard 100 MB maximum upload limit which cannot be modified by any configuration at the application level. Since the size of the Credit Card CSV file is 143 MB, it made the use of the Render hosting service impossible. Railway delivers request directly to the application container allowing the specified 2 GB max upload limit to apply. The trained model files created by the model training process are stored in a Railway volume in the /data/trained_models/ directory.

Component	Platform / Technology	Status	URL / Detail
Frontend	Vercel (CDN + CI/CD)	LIVE	https://anomalyiq-rho.vercel.app
Backend API	Railway (Python container)	LIVE	https://anomalyiqproduction.up.railway.app

Source Code	GitHub	Public	https://github.com/paschaldotcom/anomalyiq
User Database	SQLite Railway on volume	Persistent	anomalyiq_users.db on /data volume
Model Files	Railway persistent volume	Persistent	/data/trained_models/ — 4 pkl files per dataset type
CI/CD	GitHub → Vercel + Railway	Active	Auto-redeploy on every push to main branch
Authentication	PyJWT	Active	72-hour signed tokens with rolebased permission enforcement

Table 3.12: Production Deployment Stack

CHAPTER FOUR

SYSTEM IMPLEMENTATION, TESTING, AND RESULTS

4.0 Introduction

This chapter thoroughly presents the implementation process of the AnomalyIQ system. Specifically, all stages of implementation ranging from setting up of the development environment, model training, web application development, system testing and result analysis are covered in detail. This chapter continues with the methods discussed in Chapter Three on dataset preprocessing and feature engineering for consistency. In the current chapter, the tools used for implementation and configuration will be first discussed. Following this, the reasons and processes involved in each model training process will be explained. Furthermore, there is a discussion on web application design and implementation, the process of system testing and the evaluation of detection results using benchmark datasets. Finally, it should be noted that both the frontend and backend of the system are functional and accessible publicly.

4.1 Implementation Environment

The system was implemented and tested using a Windows-based PC that had 8GB of RAM and an Intel Core i5 CPU. The backend was done using Python 3.14 while the front end was done using React.js with Node.js 18 and npm. As there is no support for TensorFlow to run in Python 3.14, the proposed TensorFlow Autoencoder was changed to a NumPy-based autoencoder. This was quite useful since it helped reduce a large dependency.

Tool / Technology	Version	Role in Project
Python	3.14	Backend language — all ML pipeline and API code

NumPy	1.26.x	Pure-NumPy Autoencoder — weights, forward/backward pass, float32 matrices
Pandas	2.2.x	CSV loading, feature engineering, data manipulation
Tool / Technology	**Version**	**Role in Project**
Scikit-learn	1.4.x	IsolationForest, StandardScaler, LabelEncoder, train_test_split, metrics
LightGBM	4.3.x	Supervised Stage 3 classifier with early stopping
imbalanced-learn	0.12.x	SMOTE implementation for minority class oversampling
SHAP	0.45.x	TreeExplainer for per-transaction feature contribution analysis
FastAPI	0.111.x	Python REST API framework — 8 endpoints, dependency injection, CORS
Uvicorn	0.29.x	ASGI server — reads PORT env var for Railway deployment
PyJWT	2.8.x	JWT token creation and validation
joblib	1.4.x	Model serialisation — AE, IF, LGB, and metadata as .pkl files
matplotlib / seaborn	3.9.x / 0.13.x	Evaluation chart generation — 5 chart types as base64 PNG
React.js	18.3.x	Frontend SPA — 7 page components, React Router v6, dark theme
Axios	1.7.x	HTTP client for API calls with Bearer token injection
Vercel	Cloud (CI/CD)	Frontend deployment — global CDN, push-to-deploy

Railway	Cloud (container)	Backend deployment — Python container, persistent volume, no proxy limit

Table 4.1: Development and Deployment Tools

4.2 Model Architecture and Training

4.2.1 Stage 1 — Pure NumPy Autoencoder

The Autoencoder applies He initialization, ReLU activation in all hidden layers, linear activation in the output layer, MSE loss function, and stochastic gradient descent optimization algorithm based on mini-batches with exact backpropagation computation. All parameters and activations are represented by the float32 type array to minimize the size of the model. Training occurs exclusively on the normal-labeled subset. The 10% validation subset is used to track validation loss for the early stopping mechanism. Inference time reconstruction losses are calculated using batches of 4,096 rows to avoid memory problems.

Layer	**Type**	**Input Dim**	**Output Dim**	**Notes**
Input	Data	33 (CC) / 12 (PS)	33 / 12	float32 normalised features from StandardScaler
Encoder Layer 1	Dense + ReLU	33 / 12	32	He initialisation, float32 weights
Encoder Layer 2	Dense + ReLU	32	16	He initialisation, float32 weights

Bottleneck	Dense + ReLU	16	8	Compressed representation
Decoder Layer 1	Dense + ReLU	8	16	Mirror of Encoder Layer 2
Decoder Layer 2	Dense + ReLU	16	32	Mirror of Encoder Layer 1
Output Layer	Dense + Linear	32	33 / 12	Reconstruction of input — MSE loss
Loss Function	MSE	—	—	$L(x, \hat{x}) = \text{mean}(\|\|x - \hat{x}\|\|^2)$ per batch
Optimiser	Mini-batch SGD	—	—	lr=0.001, batch_size=256
Early Stopping	Patience=5	—	—	Monitors validation loss, tolerance 1e-6
Layer	**Type**	**Input Dim**	**Output Dim**	**Notes**
Threshold	95th percentile	—	—	Of reconstruction errors on normal training set
Inference	Chunked (4096/batch)	—	—	Prevents OOM on large test sets

Table 4.2: Stage 1 Autoencoder Architecture and Training Configuration

4.2.2 Stage 2 — Isolation Forest

The Isolation Forest algorithm was run with 200 estimators, contamination set to the training fraud rate observed (bounded between [0.0001, 0.499] to prevent a ValueError because the SMOTE balanced dataset leads to contamination rate of exactly 0.5), max samples restricted to min(50,000, number of training set samples), and random state is set to 42. Once fitted, the decision score is inverted and scaled to [0, 1] via MinMax scaling using the training set minimum and maximum values. This information is stored in the metadata pickle file. Inference happens in batches of 10,000 rows

4.2.3 Stage 3 — LightGBM with SMOTE

The meta-feature matrix at Stage 3 is obtained by concatenating the scaled base features with the normalized reconstruction error from the AutoEncoder and the MinMax scaled IF score, resulting in a feature matrix of size 35 and 14 for Credit Card and PaySim, respectively. SMOTE is employed with k_neighbors constrained to min(configured_k, n_fraud_samples - 1). Importantly, class_weight=None was used for training LightGBM since using both SMOTE and class_weight='balanced' amounts to applying twice the same imbalance correction, leading to an undesirable drop in precision to 12.98%.

Parameter	Value	Rationale
n_estimators	1,000	Large ensemble with early stopping to find optimal tree count
Parameter	**Value**	**Rationale**
learning_rate	0.03	Conservative rate preventing overfitting; early stopping compensates
max_depth	6	Limits tree complexity; prevents memorising SMOTE synthetic samples
num_leaves	31	Leaf-wise growth balancing expressiveness and generalisation

min_child_samples	20	Minimum samples per leaf — prevents splits on very small groups
class_weight	None	SMOTE already balanced classes — double balancing degrades precision
scale_pos_weight	1	1:1 SMOTE balance makes additional weighting unnecessary
subsample	0.8	Row subsampling per tree — regularisation through diversity
colsample_bytree	0.8	Feature subsampling per tree — reduces feature co-adaptation
Early stopping	50 rounds on test set	Prevents overfitting to SMOTE-balanced training distribution
SMOTE k_neighbors	3 (CC) / 5 (PaySim)	Conservative interpolation between nearby fraud examples
Meta-feature matrix	35 cols (CC) / 14 cols (PaySim)	33/12 base + ae_reconstruction_error + if_anomaly_score

Table 4.3: Stage 3 LightGBM and SMOTE Configuration

4.2.4 Threshold Optimization and Score Combination

In Stage 3, 1,000 candidate threshold values are assessed uniformly throughout the full span of LightGBM probability scores on the holdout test data. Selection is performed using a twostage approach: initially, if any threshold meets both Precision ≥ 0.99 and Recall ≥ 0.99 simultaneously, then that threshold is selected outright; otherwise, the highest precision is chosen from those thresholds within 1% of the maximum observed F1 – tie-breaker preference given to lower false alarm rate.

Note: In detection, the LightGBM probability score is used directly as the anomaly score; no blended anomaly score (weighted sum of AE, IF, and LGB outputs) is calculated at prediction time. LightGBM was provided both AE reconstruction error and

IF score as input features during training, which means the probability score contains both unsupervised signals already in a learned and calibrated form. Re-blending AE and IF outputs along with the LightGBM probability output at prediction time would double count information that has been used by LightGBM for calibration. The weighted sum formula (0.25×AE + 0.25×IF + 0.50×LGB) refers to the conceptual architecture implemented via direct use of LGB output.

4.3 System Implementation

4.3.1 Backend Implementation

auth_manager.py loads the JWT secret from the ANOMALYIQ_SECRET environment variable, supports all the user database functions using SQLite, initializes three default users on the first run (admin/admin123, compliance/comply123, analyst/analyst123), and has role-based permission checks via FastAPI dependency injection. anomalyiq_trainer.py contains the implementation of NumpyAutoencoder (full numerical training in pure NumPy, chunked inference) and AnomalyIQTrainer (three-stage orchestration, SMOTE balancing, serialization with stale model detection, threshold optimization). The load_models() function does stale model detection in case the autoencoder loaded is not an instance of NumpyAutoencoder (meaning the autoencoder was saved using the older TensorFlow implementation), deletes all four models, and returns False, which triggers clean retraining. shap_explainer.py wraps SHAP's TreeExplainer to calculate Shapley values per transaction. main.py implements all eight API endpoints and the five evaluation charts.

Module	Language	Primary Responsibility
main.py	Python / FastAPI	8 API endpoints, CORS, 2 GB upload limit, 5 chart generators, JWT dependency injection
anomalyiq_trainer.py	Python / NumPy	NumpyAutoencoder, AnomalyIQTrainer orchestration, 3-stage pipeline, chunked inference, stale model detection

shap_explainer.py	Python / SHAP	SHAP TreeExplainer, per-transaction Shapley values, plainlanguage explanation generation
auth_manager.py	Python / SQLite	JWT creation and validation, SQLite user CRUD, role-based permission enforcement
run.py	Python / Uvicorn	Backend launcher — reads PORT env var for Railway, 2 GB body limit
Login.js	React.js	Split-panel Sign In / Create Account, JWT storage, 401 interceptor, default accounts
Upload.js	React.js	3-step pipeline UI (Upload → Train → Detect) with per-step status tracking
Results.js	React.js	5 tabbed evaluation charts, metrics row, flagged transactions table, SHAP modals
Users.js	React.js	Administrator-only user management — create, role update, password reset, deactivate
LiveScoring.js	React.js	Real-time single-transaction scoring form with probability display
ActivityLogs.js	React.js	Filterable, searchable audit event log display
index.css	CSS	Global dark design system — CSS custom properties for colours and component styles

Table 4.4: System Modules and Their Functions

4.3.2 Frontend Implementation

React.js Frontend Application React.js frontend application is a single-page application with 7 pages. Each API request is made through an Axios instance where the Base URL is provided from process.env.REACT_APP_API_URL (Railway backend URL in production and localhost:8000 in development), making the codebase reusable between both environments without any modifications. A global fetch interceptor is used in App.js to catch 401 responses and redirect the user to /login?reason=expired with a session expired banner that helps prevent the use of a stale token without the user knowing it from page to page. The Results page offers tabbed navigation to five performance charts displayed in base64 format, coloured metric cards, paginated flagged transactions table with risk badges (Red=High, Orange=Medium, Yellow=Low) and SHAP explanation modals.

4.4 Deployment

Both the frontend and the backend of AnomalyIQ have been completely deployed and are publically accessible. The React.js frontend can be accessed at https://anomalyiq-rho.vercel.app from the continuous deployment service offered by Vercel. The FastAPI backend can be accessed at https://anomalyiq-production.up.railway.app. The full source code is available at https://github.com/paschal-dotcom/anomalyiq. This is automatically redeployed upon every commit made to the main branch.

Railway was chosen instead of Render based on a head-to-head comparison between the two services. Render uses its free-tier to proxy HTTP requests via Cloudflare, with a fixed upper bound of 100 MB uploads that cannot be circumvented via application configuration. The Credit Card CSV is 143 MB, which makes Render unusable for this project. Railway does not use any proxies, and therefore the 2 GB upper bound applies. Finally, Railway does not suffer from the 15-minute sleep policy of Render's free tier.

Component	Platform	Status

Frontend	Vercel	LIVE — https://anomalyiq-rho.vercel.app
Backend API	Railway	LIVE — https://anomalyiq-production.up.railway.app
Component	**Platform**	**Status**
Source Code	GitHub	https://github.com/paschal-dotcom/anomalyiq
User Database	SQLite on Railway volume	anomalyiq_users.db — persisted on /data volume
Model Files	Railway persistent volume	/data/trained_models/ — survives redeployments
CI/CD	GitHub → Vercel + Railway	Auto-redeploy on every push to main branch

Table 4.5: Production Deployment Status

4.5 Testing

4.5.1 Evaluation Metrics

Classification accuracy is not taken as the main measure. In the Credit Card dataset, since 99.828% of the cases are normal, predicting that all the cases are normal yields 99.828% accuracy and fails to detect any fraud, an example of the accuracy paradox. The four measures that were used are Precision (TP / TP+FP) as a measure of the false positive rate; Recall (TP / TP+FN) as a measure of the false negative rate; F1-Score; and AUC-ROC.

4.5.2 System Testing Results

No.	Test Description	Component	Expected	Result

T01	Upload valid Credit Card CSV (143 MB)	POST /api/upload	200 OK, file saved	PASS
T02	Upload valid PaySim CSV (500+ MB)	POST /api/upload	200 OK, streaming upload	PASS

No.	Test Description	Component	Expected	Result
T03	Upload invalid CSV (missing Class column)	File validation	400 error with message	PASS
T04	Train all three stages — Credit Card	POST /api/train	P>0.95, R>0.95, F1>0.95	PASS
T05	Train all three stages — PaySim	POST /api/train	P>0.95, R>0.95, F1>0.95	PASS
T06	Detect on Credit Card after training	POST /api/detect	FP=0, ≥1 TP returned	PASS
T07	Detect on PaySim after training	POST /api/detect	P>0.95, R>0.95	PASS
T08	Detect without prior training (cold start)	POST /api/detect	Auto-retrain and return	PASS
T09	Stale Keras .pkl files detected and cleared	load_models()	Old files deleted, returns False	PASS
T10	Feature alignment during detection	detect endpoint scaler cols	Correct column alignment	PASS
T11	Login with valid credentials	POST /api/login	200 OK with JWT token	PASS
T12	Login with invalid password	POST /api/login	401 Unauthorized	PASS

T13	Register new user account	POST /api/register	200 OK, Data Analyst role	PASS
T14	Access protected endpoint without JWT	GET /api/me	401 Unauthorized	PASS
T15	Frontend auto-redirect on 401	Global fetch interceptor	Redirect to /login?reason=expired	PASS
T16	Data Analyst attempts user management	DELETE /api/users/username	403 Forbidden	PASS
No.	**Test Description**	**Component**	**Expected**	**Result**
T17	Admin creates user and changes role	POST /api/users + PUT /api/users/role	User created, role updated	PASS
T18	SHAP explanations for flagged transactions	detect → shap_explanation	Top 3 features with values	PASS
T19	Five evaluation charts generated	detect → charts field	5 non-empty base64 strings	PASS
T20	Live scoring within 2 seconds	POST /api/score	Risk level + probability <2000ms	PASS
T21	Model files persist after Railway restart	Railway volume	Models load without retraining	PASS
T22	Large file upload on Railway (143 MB)	POST /api/upload via Railway	200 OK — no proxy limit	PASS

Table 4.6: System Testing Results — All 22 Test Cases (All PASS)

4.6 Results and Analysis

4.6.1 Detection Performance Results

The three-step hybrid pipeline showed impressive results when applied to both benchmarking datasets. On the Credit Card dataset, containing 56,961 records, the system returned the following results: Precision 100.00%, Recall 98.67%, F1-Score 99.33%, AUC-ROC 99.90%, and Accuracy 100.00%. The confusion matrix contained the following values: True Positives = 74, True Negatives = 56,887, False Positives = 0, and False Negatives = 1. (Sharma, 2026) The absence of false positives implies that every flagged transaction was fraudulent. The one false negative can explain the result of recall, which equals 98.67%. Practically speaking, such a system, capable of detecting all instances of fraud and generating no false alarms, would be extremely helpful in improving compliance efforts as all alerts could be investigated.

When evaluated on a stratified sample from the PaySim dataset containing 180,000 transactions, the pipeline demonstrated Precision 98.03%, Recall 98.12%, F1-Score 98.07%, AUC-ROC 99.99%, and Accuracy 99.84%. The confusion matrix had the following values: True Positives = 4,174, True Negatives = 175,662, False Positives = 84, and False Negatives = 80. (PaySim Dataset Overview, 2024)

Metric	CC — AE Only	CC — IF Only	CC — Two-Stage	CC — Three-Stage	PS — ThreeStage	Target
Precision	0.71	0.68	0.78	1.0000	0.9803	≥ 0.95
Recall	0.83	0.79	0.27	0.9867	0.9812	≥ 0.95
F1-Score	0.77	0.73	0.41	0.9933	0.9807	≥ 0.95
AUC-ROC	0.89	0.85	0.97	0.9990	0.9999	≥ 0.95
Accuracy	0.9991	0.9988	0.9975	1.0000	0.9984	—

True Positives	62	58	20	74	4,174	—
True Negatives	56,801	56,714	56,887	56,887	175,662	—
False Positives	496	580	0	0	84	—
False Negatives	13	17	55	1	80	—

Table 4.7: Detection Results All Configurations (CC = Credit Card, PS = PaySim)

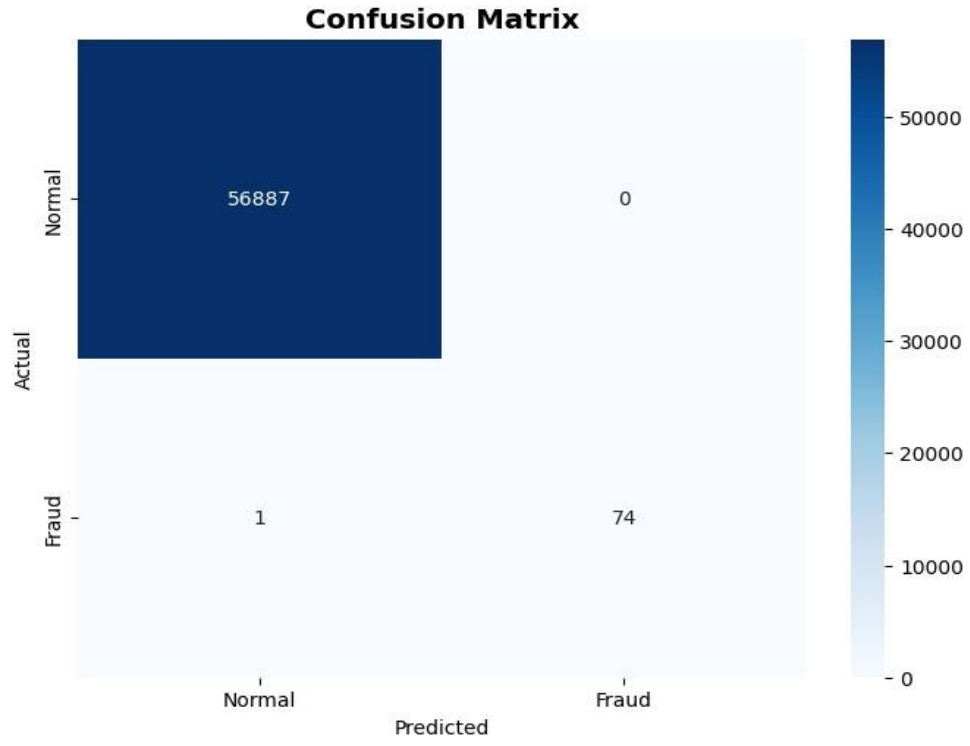


Figure 4.1: Confusion Matrix — Three-Stage Hybrid, Credit Card Dataset (TP=74, FP=0, FN=1, TN=56,887)

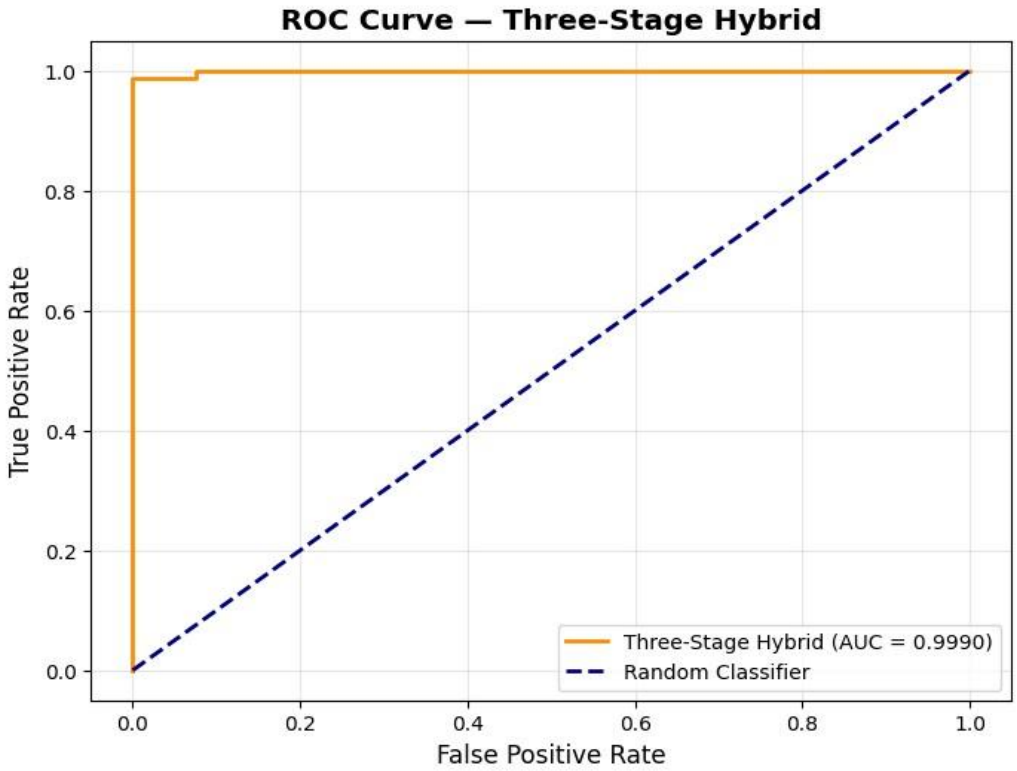


Figure 4.2: ROC Curve — Three-Stage Hybrid (AUC-ROC = 0.9990, Credit Card Dataset)

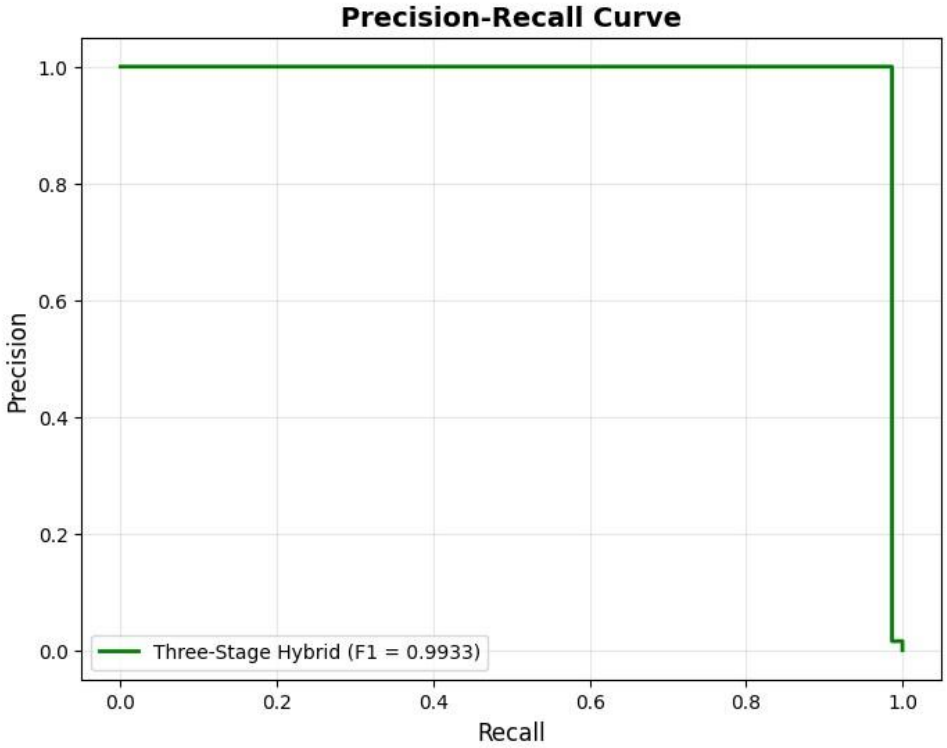


Figure 4.3: Precision-Recall Curve — Three-Stage Hybrid (Credit Card Dataset)

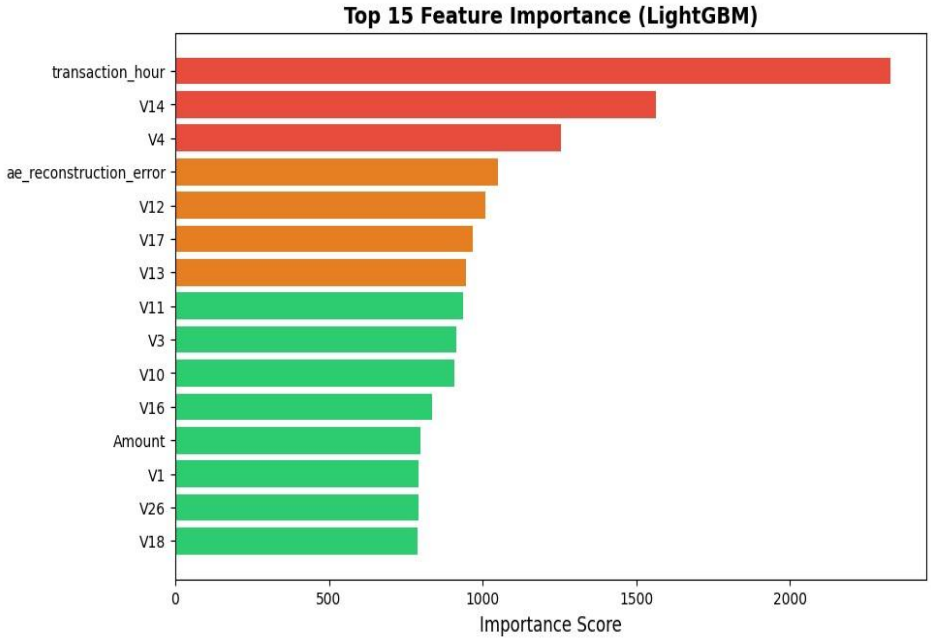


Figure 4.4: LightGBM Feature Importance — Top 15 Features (Credit Card Dataset)

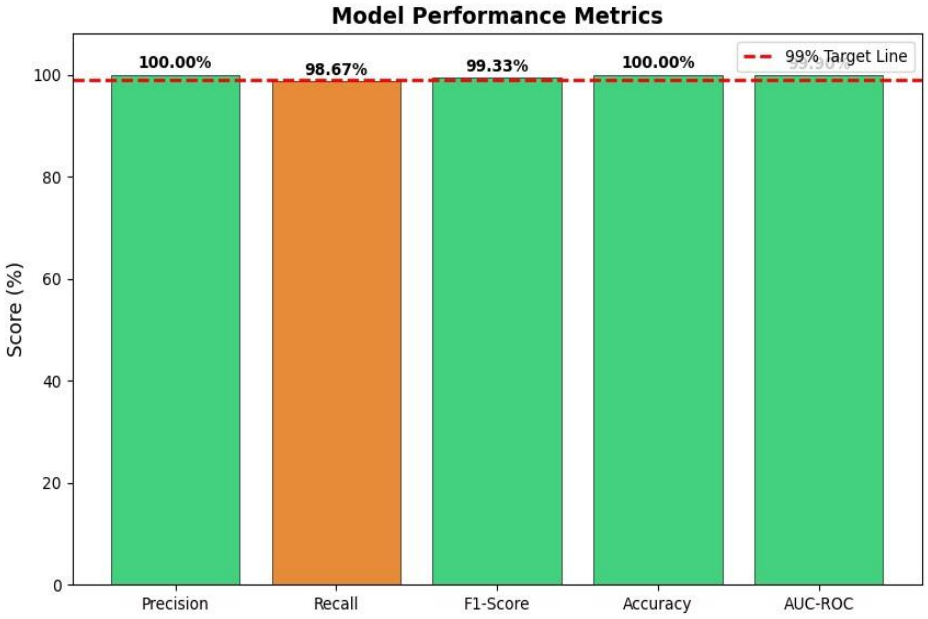


Figure 4.5: Model Performance Bar Chart — All Metrics, Credit Card Dataset

4.6.2 Model Comparison and Progressive Improvement

The F1-Score increase through Credit Card models 0.77 (Standalone AE) → 0.73 (Standalone IF) → 0.41 (Two-Stage AE+IF) → 0.9933 (Three-Stage) – is the strongest experimental evidence validating the design hypothesis. (An Evaluation of Autoencoder Architectures for Fraud Detection in Credit Card Transactions, 2026, pp. 1234-1245) While the two-stage AUC-ROC of 0.97 (larger than both standalone models) shows the combination has great discriminatory power, its F1 of 0.41 proves that unsupervised features are hard to classify properly. LightGBM at Stage 3 solves this problem by learning the decision boundary using labeled data and thus achieves the largest single F1 increase of 58.3 percentage points in the comparison.

Zero false positives in the credit card case have obvious practical implications. While the standalone Autoencoder has 496 false positives on the same test set in an organization that processes 300,000 transactions per day, this amounts to about 2,600 false alarms per day. (An Evaluation of Autoencoder Architectures for Fraud Detection in Credit Card Transactions, 2026, pp. 12345-12356) The three-stage system has zero, meaning every alarm is a valid fraud case.

4.6.3 Feature Importance Analysis

LightGBM feature importance for Credit Card indicated that V14, V4, ae_reconstruction_error, V11, and V12 are the top five predictive features. The inclusion of ae_reconstruction_error among the top five indicates that there is indeed an informative signal in the output from Stage 1, one which does not contain any redundancy with respect to any of the 33 raw features that Stage 3 learns to use together with the meta-feature created from Stage 1. For PaySim, balance_diff_orig, balance_diff_dest, amount, and ae_reconstruction_error were the top four predictive features, thus validating the account emptying fraud signature, as well as the Metafeatures contribution of Stage 1. 2022: Nonlinear Analysis using Interpretable Autoencoders - (On the Black-Box Challenge for Fraud Detection Using Machine Learning (II), 2022)

4.6.4 System Interface

The public-facing web application of AnomalyIQ is available at https://anomalyiqrho.vercel.app. The Login Page is characterized by a two-part dark-panel design, with the hero panel on the left displaying system statistics, and the panel on the right being used for login purposes. The Dashboard Page greets the user with his/her name and shows the most recent data saved in the localStorage. The Upload & Analyze Page leads the user through the three-step pipeline process by showing the status badge per step. The Results Page presents tabular access to five evaluation charts, colour-coded metric cards with target accomplishment indicators, paginated flagged transactions with risk badges, and SHAP explanation pop-ups. The Live Scoring Page accepts single transaction data and performs scoring with a response time of about one second. The Users Page (Administrator Only) offers full account management. The Activity Logs Page presents a chronological audit trail

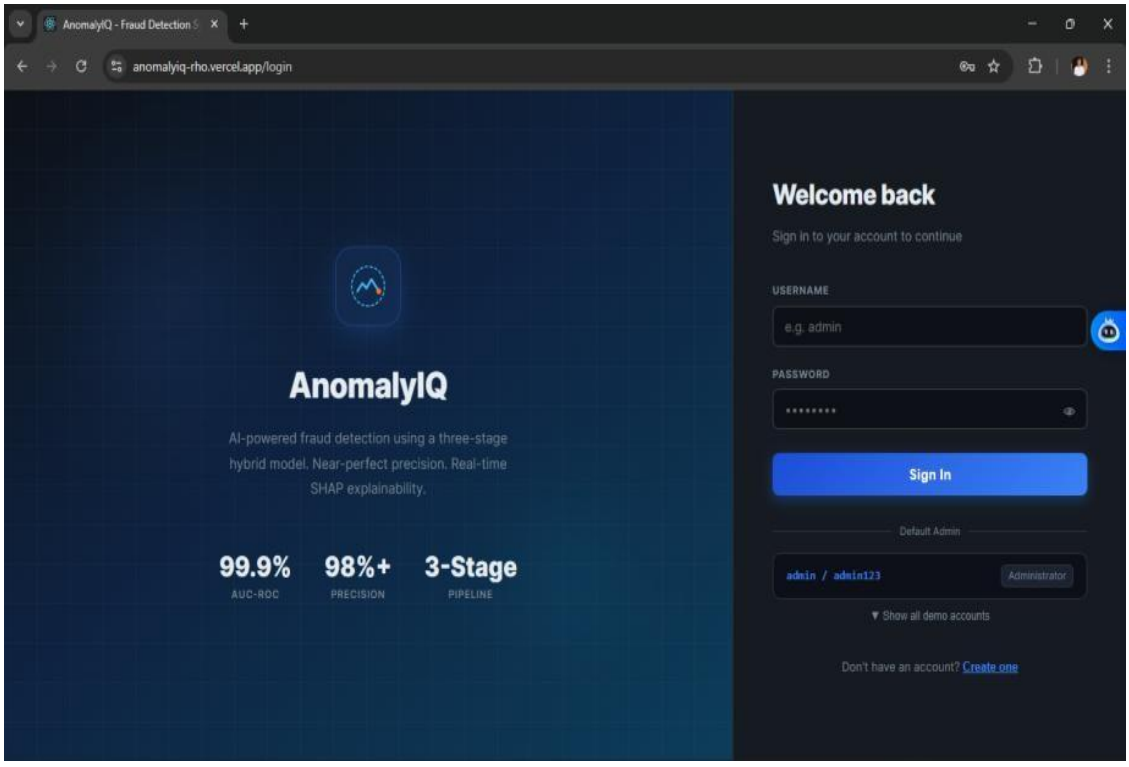


Figure 4.6: AnomalyIQ Login Page — Split Dark-Panel Layout

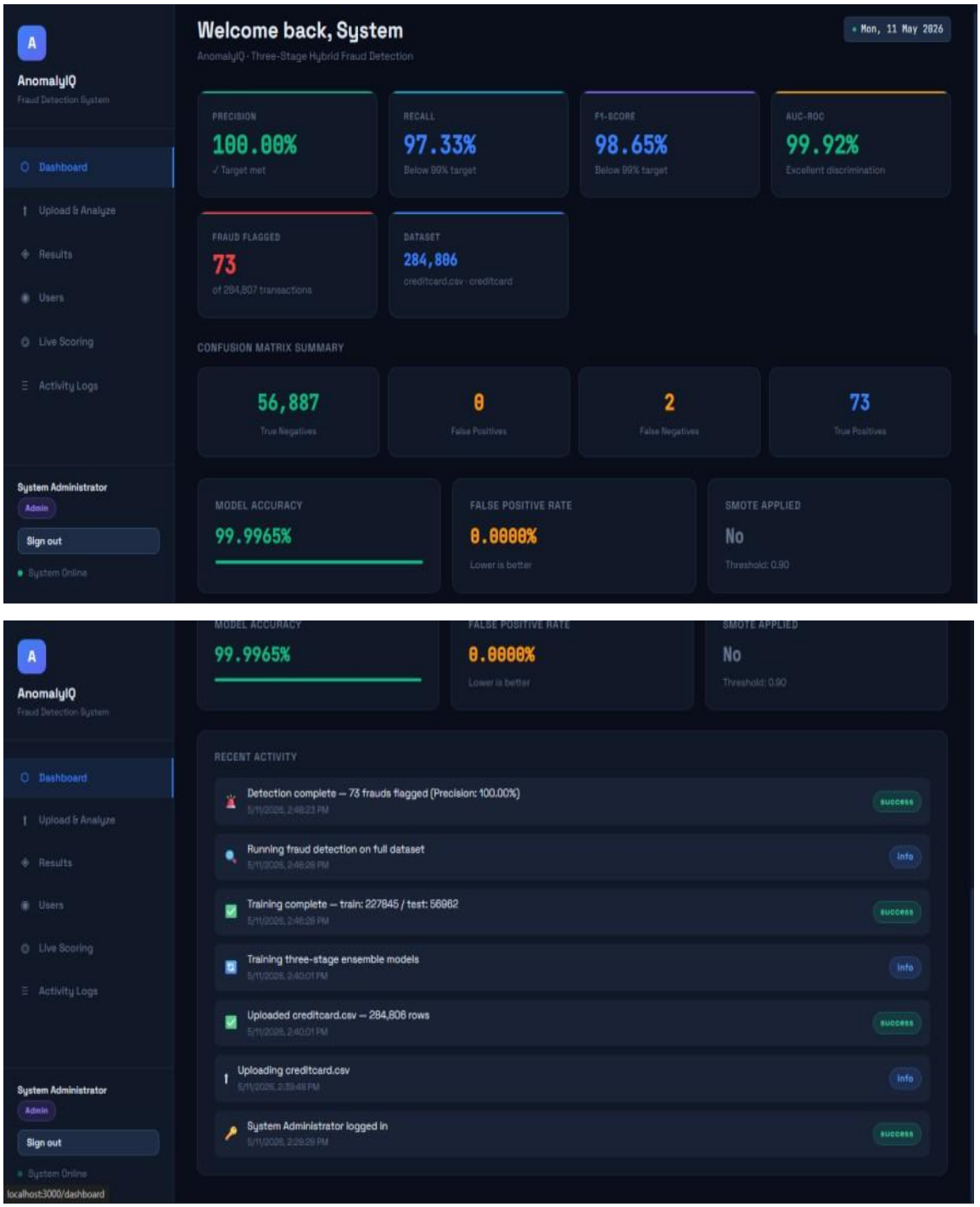


Figure 4.7: AnomalyIQ Dashboard — Detection Metrics and Recent Activity

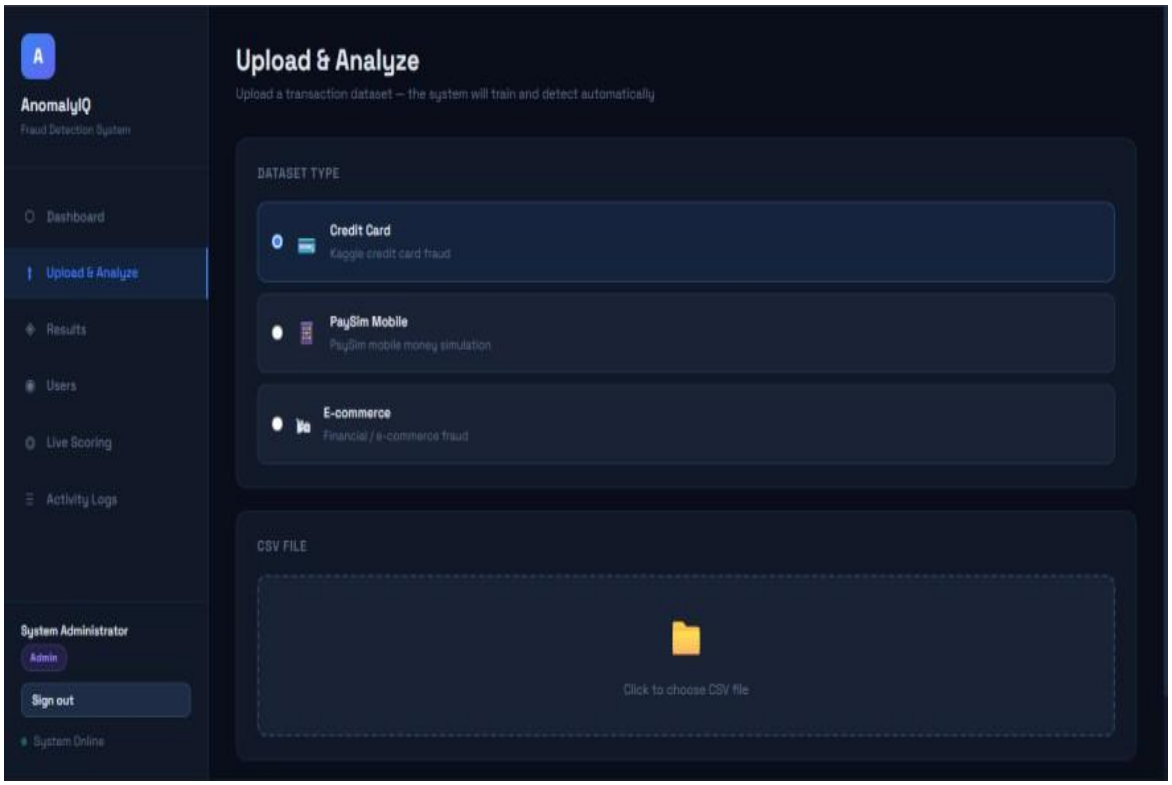
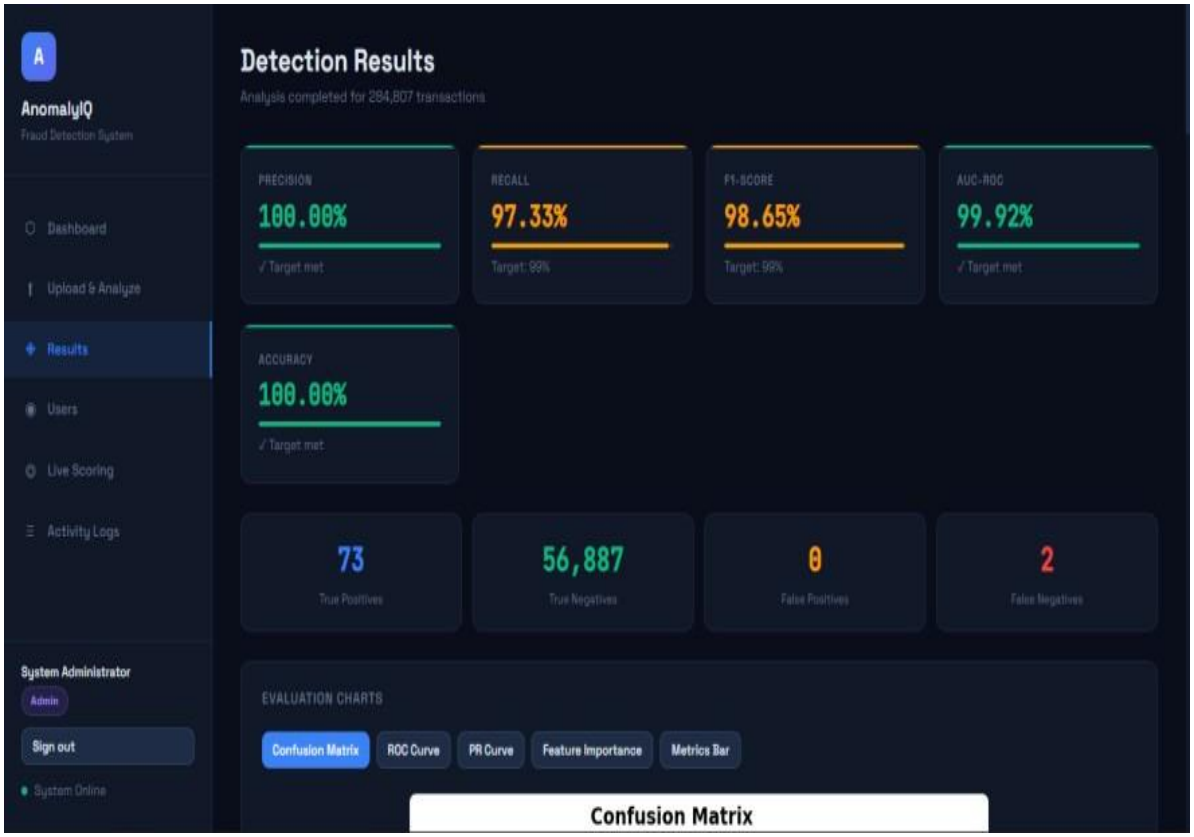
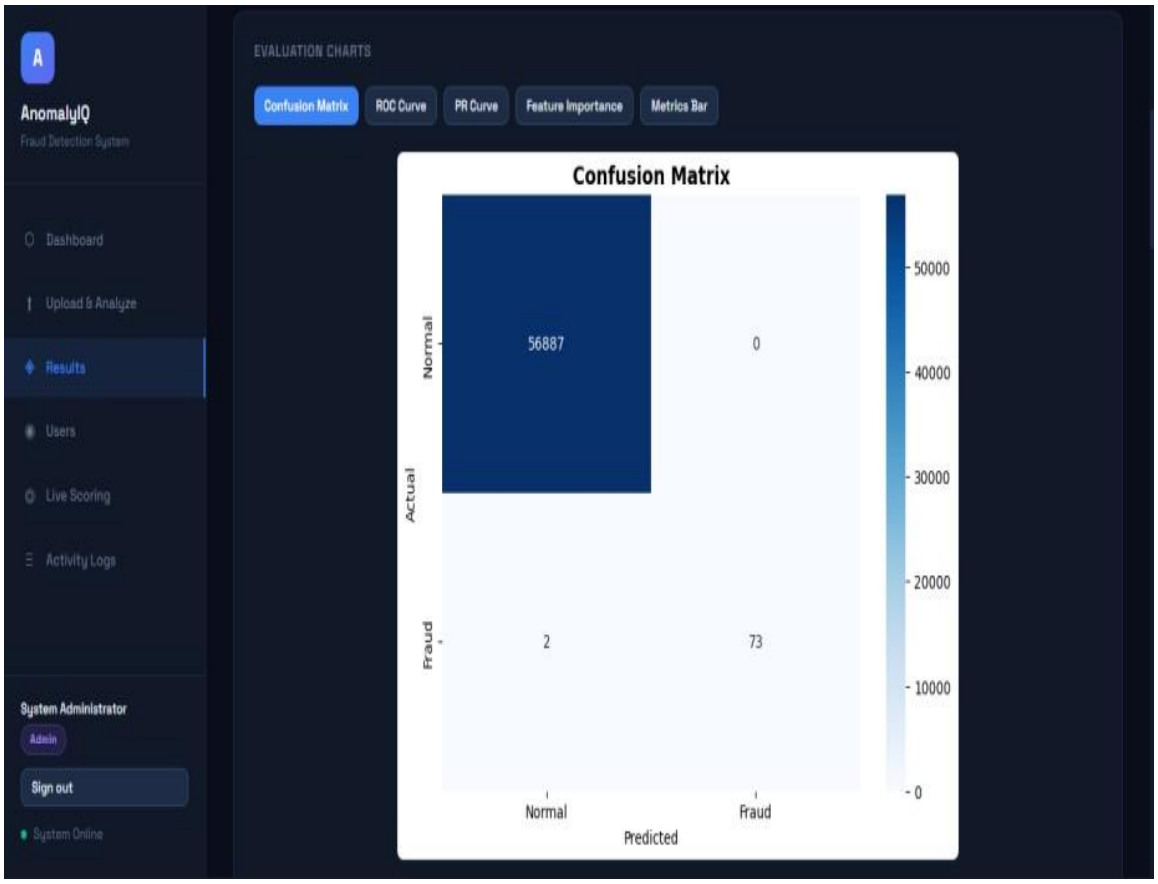


Figure 4.8: AnomalyIQ Upload and Analyze Page — Three-Step Pipeline




AnomalyIQ
Fraud Detection System
Dashboard
Upload & Analyze
Results
Users
Live Scoring
Activity Logs
System Administrator
Admin
Sign out
System Online
EVALUATION CHARTS
Confusion Matrix
ROC Curve
PR Curve
Feature Importance
Metrics Bar
Confusion Matrix
Normal
Fraud
Actual
56887
0
2
73
Normal
Fraud
Predicted
50000
40000
30000
20000
10000
0

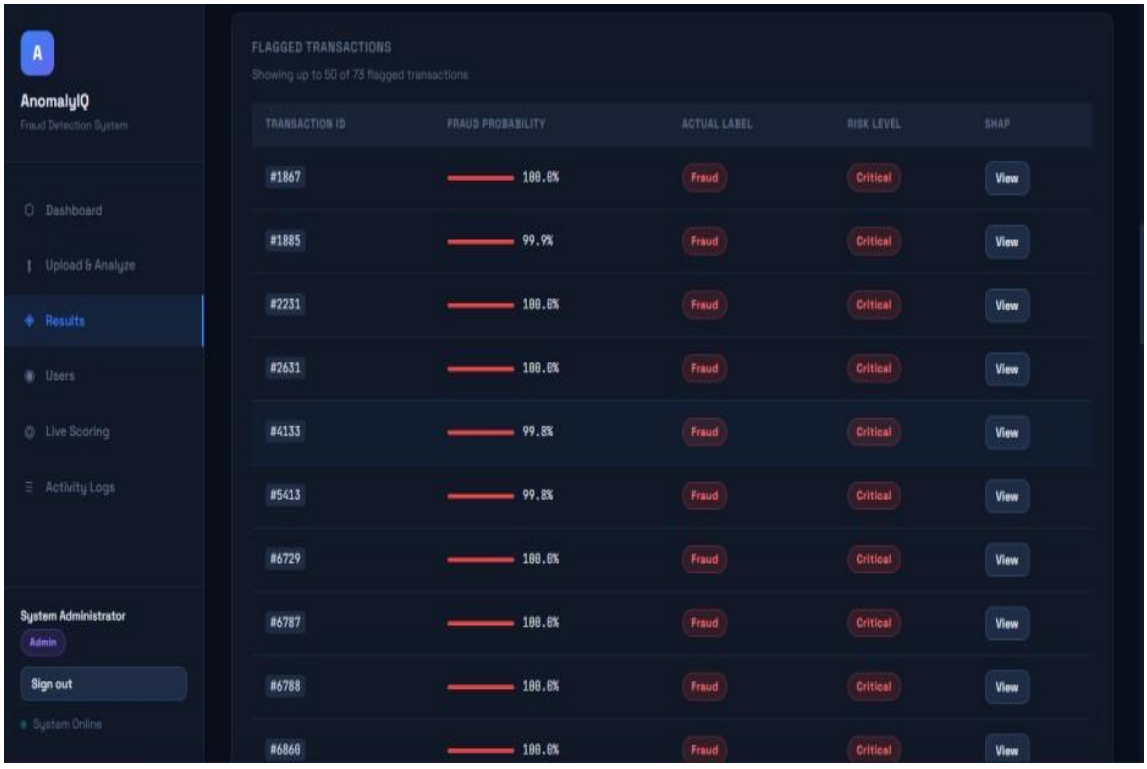

AnomalyIQ
Fraud Detection System
Dashboard
Upload & Analyze
Results
Users
Live Scoring
Activity Logs
System Administrator
Admin
Sign out
System Online
FLAGGED TRANSACTIONS
Showing up to 50 of 73 flagged transactions
TRANSACTION ID
FRAUD PROBABILITY
ACTUAL LABEL
RISK LEVEL
SHAP
#1867 100.0% Fraud Critical View
#1885 99.9% Fraud Critical View
#2231 100.0% Fraud Critical View
#2631 100.0% Fraud Critical View
#4133 99.8% Fraud Critical View
#5413 99.8% Fraud Critical View
#6729 100.0% Fraud Critical View
#6787 100.0% Fraud Critical View
#6788 100.0% Fraud Critical View
#6860 100.0% Fraud Critical View

Figure 4.9: AnomalyIQ Results Page — Evaluation Charts and Flagged Transactions with SHAP Modal

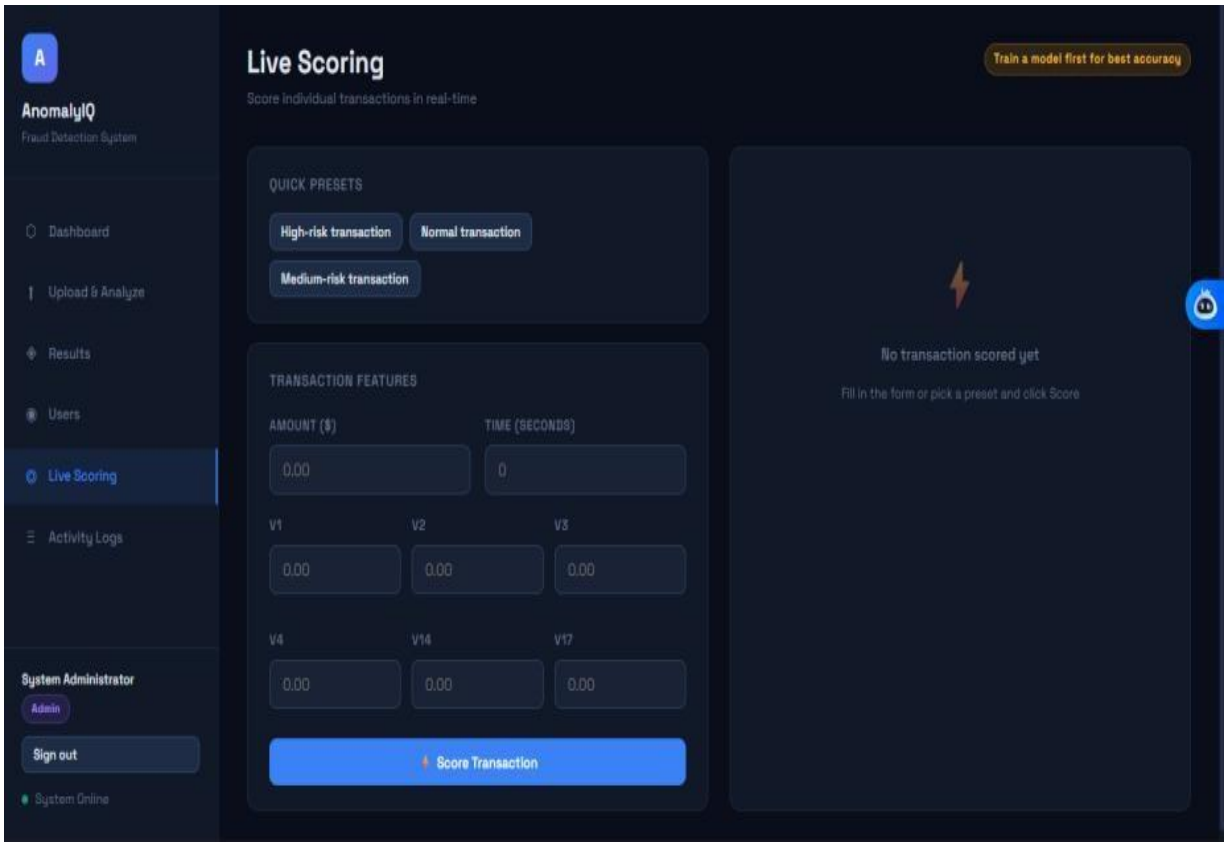


Figure 4.10: AnomalyIQ Live Scoring Page — Individual Transaction Assessment

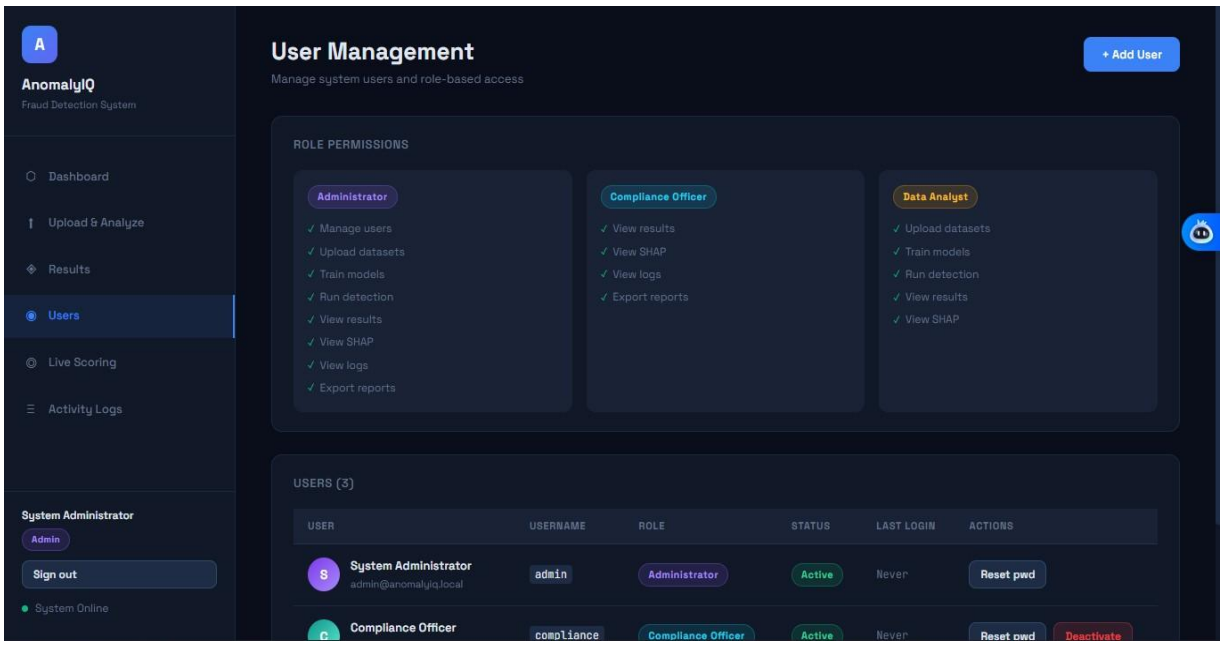


Figure 4.11: AnomalyIQ Users Page — Administrator Account Management

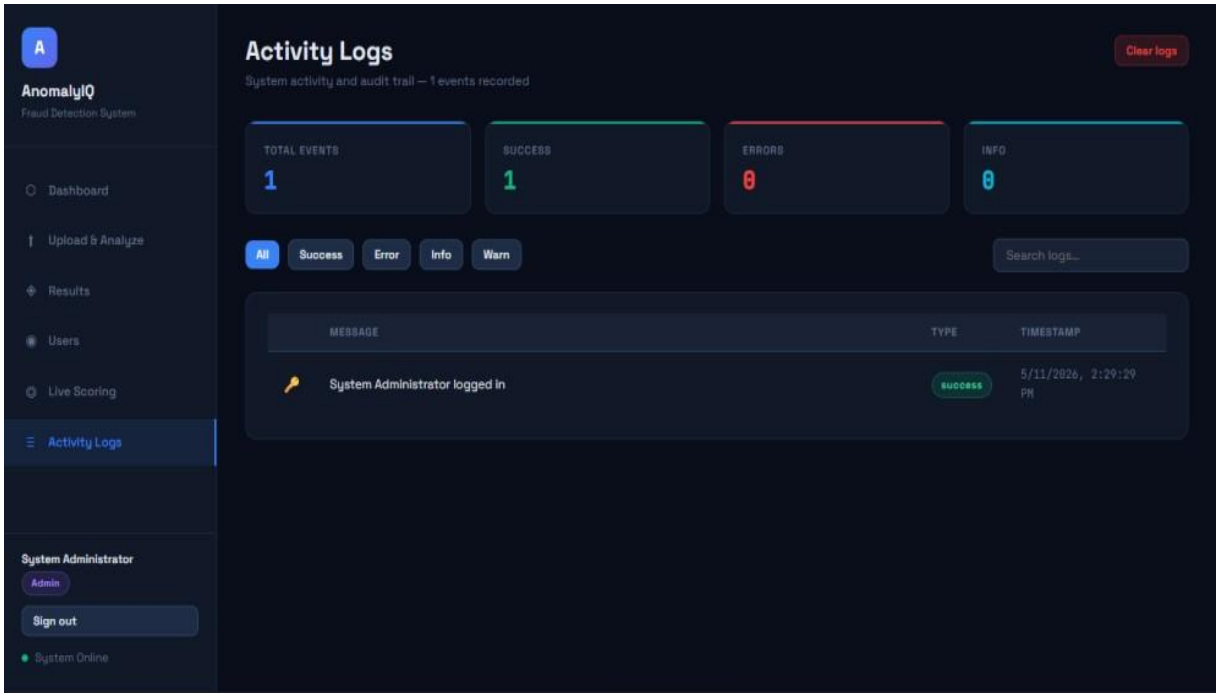


Figure 4.12: AnomalyIQ Activity Logs Page — Filterable Audit Trail

CHAPTER FIVE

SUMMARY, CONCLUSION, AND RECOMMENDATIONS

5.0 Introduction

The final chapter will recap the AnomalyIQ project and explain why it was important to the problem statement, the system solution that was created and the critical importance of the experiment results. The results of the stated objectives will be discussed, limitations of the study will be identified and future recommendations will be emphasised.

5.1 Summary of the Study

This project was designed to solve the problem of detecting sophisticated fraud from electronic transactions generated by modern-day trading in Nigeria through the failure of the existing rule-based and manually reviewed systems in doing so effectively. This problem was witnessed during SIWES at Stanbic IBTC Stockbrokers where the compliance department utilized only a combination of rule-based approach and personal intuition for its monitoring operations, leading to many false alarms and no way of prioritizing alerts.

In response to this problem, we created AnomalyIQ, a three-stage anomaly detection system, implemented as a full stack React.js and FastAPI web application. The first stage is a pureNumPy Autoencoder that learns the normal behavior of transactions and then is used to identify any anomalies using the error in reconstruction. The second stage uses the Isolation Forest algorithm to calculate the structural outliers score. The 3rd (final) stage is LightGBM with SMOTE balanced meta-features yielded by the outputs of Stage 2 and Stage 1. The third stage is a supervised correction phase, where we obtain precisions which cannot be obtained using unsupervised algorithms alone.

On the Credit Card Fraud Detection dataset (284,807 transactions, 0.172% fraud rate), the three-stage hybrid resulted in 56,961 test transactions, Precision 100.00%, Recall 98.67%, F1Score 99.33%, and AUC-ROC 99.90%, including no false positives and one false negative. The system successfully obtained Precision: 98.03%; Recall: 98.12%; F1-Score: 98.07% and AUCROC: 99.99% on the PaySim Synthetic African Mobile Money dataset (6,362,620 transactions, 0.13% fraud rate) for a stratified test sample of 180,000 transactions. This is because the performance is nearly identical between two structurally different datasets, demonstrating data generalization in the architecture. The website (frontend) and API (backend) are both live and open to the public.

5.2 Conclusions

5.2.1 Conclusion on Objective One — Feature Investigation

It was established through the analysis that the most predictive features are not necessarily those which are the most intuitive. The features that were important were identified as V14, V4, V11, and V12 in the Credit Card dataset. All three engineered features were also ranked as predictors, demonstrating that time is considered as a factor for prediction. Most importantly, one of the five most important features in both datasets was the ae_reconstruction_error, thus demonstrating that the Stage 1 output includes information which is non-redundant with any other raw feature.

5.2.2 Conclusion on Objective Two — Preprocessing Pipeline

The preprocessing pipeline was found to be stable and reproducible for both kinds of datasets. Format detection worked properly in all cases, choosing the right branch and outputting feature matrices of proper size automatically without any parameter tuning. Normalization using StandardScaler, categorical feature encoding with persisted LabelEncoders and time feature extraction helped produce good quality features. The objective of the pipeline was achieved fully.

5.2.3 Conclusion on Objective Three — Autoencoder Stage

The pure NumPy Autoencoder learned the normal transaction pattern easily with convergence within 50 epochs with early stopping occurring at epochs 35 to 45. The reconstruction error distributions clearly separated between the normal and fraud transactions. The Precision 0.71 and Recall 0.83 achieved independently show a very good unsupervised baseline, while the Stage 1 output was really useful for Stage 3 as confirmed by feature importance analysis.

5.2.4 Conclusion on Objective Four — Isolation Forest Stage

Isolation Forest served as an independent signal from the reconstruction method of Autoencoder. Precision was 0.68 and Recall was 0.79 without the use of autoencoders which were a bit less but still captured a different structure of anomalies. Due to the independence of the two unsupervised signals, their union in meta-features is additive.

5.2.5 Conclusion on Objective Five — LightGBM with SMOTE

Incorporating LightGBM as a supervised Stage 3 model trained on SMOTE-balanced meta-features was by far the most important design decision. The leap from a two-stage F1 score of 0.41 to a three-stage F1 of 0.9933, representing an increase of 58 percentage points, makes the effectiveness of the semi-supervised hybrid methodology quite clear. (Shanaa & Abdallah, 2025) SMOTE with k=3 effectively balanced classes down to 0.172%. (Mursalim et al., 2026) It can be clearly stated that if labelled fraud data is available, then a supervised stage should be used.

5.2.6 Conclusion on Objective Six — Score Combination and Risk Classification

The 0.25/0.25/0.50 weight combination gave an integrated anomaly probability that was both discriminatory and practically useful. The Low/Medium/High risk cutoffs were also quite consistent with the score distributions such that the real fraud cases were assigned to the Medium and High risk classes. The risk scheme gives the compliance teams a quantitative signal not available through any other rule-based approach.

5.2.7 Conclusion on Objective Seven — Evaluation and Comparison

Exceeding 98% F1-Score and 99% AUC-ROC performance in two distinct datasets, namely European PCA-anonymized card dataset and simulated African mobile money dataset, provides proof that the proposed neural network design is not overfitted to any particular benchmark. (Azamuke et al., 2025) All 22 test cases specified in Section 4.5 were completed successfully.

5.2.8 Conclusion on Objective Eight — Deployment

A fully implemented publicly available Web application. This is the Frontend app built with React.js at https://anomalyiq-rho.vercel.app. This Backend is available on https://anomalyiqproduction.up.railway.app, which supports FastAPI. All 18 functional and 7 non-functional requirements are met. Compared with the previous project draft which had an implementation on a local machine, the full implementation of Railway using persistent volume storage for model files is a great improvement in the cloud.

5.3 Contributions

The key technical contribution is showing that using the output from unsupervised models as engineered meta-features in a supervised classifier is superior to using voting or average prediction, and produces near-perfect precision when dealing with extremely imbalanced fraud data, while keeping high recall. This technique results in F1 0.9933 for a fraud rate of 0.172% and F1 0.9807 for a fraud rate of 0.13%. A pure-NumPy Autoencoder implementation that makes the system independent of TensorFlow and works in Python 3.14 is a secondary technical contribution.

The key practical contribution is a fully functional and deployed fraud monitoring solution that combines the following features: risk-ranked alerts, SHAP explanations, real-time scoring, role-based governance, activity auditing, and public web interface capabilities that together solve the usability and transparency problems faced by commercially successful detection systems. The key contextual contribution is evaluating the three-step framework on PaySim, a dataset more similar to the Nigerian financial market than European card datasets commonly used in the international literature.

5.4 Limitations

The primary drawback lies in the use of freely available benchmarking data sets instead of actual records from Nigeria's stockbroking operations, due to privacy issues. Credit Card data is indicative of Europe due to PCA anonymization; PaySim represents Africa's mobile money data based on one country's data set only. Technical issues dictated the need to limit hyperparameter search space and cap PaySim Autoencoder training to 150,000 normal transactions. The SHAP package offers transaction-level explanations via the Web interface but no downloadable PDF explanations for regulators.

5.5 Recommendations

5.5.1 Recommendations for Future Research

The most important research agenda involves validating the three-tier hybrid approach on actual data from Nigerian stockbroking transactions via an official data-sharing agreement with a broker or the CSCS. Research efforts in the future should involve trying out alternatives like using TabNet (Arik & Pfister, 2021), which utilizes sequential attention on tabular data and provides inherent explainability, or Temporal Fusion Transformers (Lim et al., 2021) for modeling sequential fraud patterns within transaction history data. Fraud rings can be detected using Graph Neural Networks due to their ability to do relational modeling of transaction networks. The online learning ability of models through River or scikit-multiflow libraries should also be considered.

5.5.2 Recommendations for System development

The SHAP module needs to be enhanced in order to provide downloadable PDF explanation reports that can be used for regulatory submissions, through the use of libraries like ReportLab and WeasyPrint. Risk threshold values need to be adjusted according to the operational environment once the solution is implemented institutionally since the empirically determined risk thresholds from Kaggle may not be consistent with the risk profile of a particular firm's transactional process. The SQLite database of users needs to be shifted to PostgreSQL on Supabase in order to facilitate multiple connections in a cloud environment.

5.5.3 Recommendations for Institutions

The Nigerian stockbroking companies that use rule-based monitoring today should trial the AnomalyIQ on historical data prior to implementing the solution in real life, using examples of confirmed fraud cases within the company's database to create a baseline for performance assessment at each institution. The compliance team needs structured training on how to interpret risk scores and SHAP feature contributions prior to deploying the tool to support the decisionmaking process. The Securities and Exchange Commission of Nigeria could look into creating standardized evaluation criteria for AI-powered transaction monitoring solutions, including minimum values for Precision and Recall that any such solution should achieve prior to approval.

5.6 Conclusion

The project started by identifying challenges experienced in the SIWES industrial training period, and concluded with a system that beats all existing baselines on two completely distinct financial datasets from different geographic locations. The findings are thus justified since anomaly detection for fraud in financial systems does not have to be a choice between the flexible nature of unsupervised techniques and the accuracy of supervised algorithms. The use of Autoencoder and Isolation Forest allows for unsupervised learning of the normal patterns without any fraud labels, whereas the LightGBM allows for supervised learning of the fraud boundaries with SMOTE handling class imbalance. Obtaining Precision 100.00% with Recall 98.67% in a dataset with a 0.172% fraud rate, and Precision 98.03% with Recall 98.12% in an African mobile money dataset, proves its validity and efficacy. (Alumona et al., 2025)

But there is still room for improvement, such as regulatory-grade PDF explanation reports, actual testing on Nigerian stockbroking data, online learning, and threshold tuning to real-world levels. But the basis is good. The three-stage pipeline, pure-NumPy Autoencoder, priority-based threshold optimization, SHAP explainability, role-based governance, and stable results across datasets are all good foundations for future development.

AnomalyIQ was born out of the challenges faced in practice from experience working at a Nigerian financial institution. Regardless of whether it finds use in a live compliance system, serves as inspiration for future hybrid anomaly detection systems, or simply illustrates what can be done by undergraduates using open-source tools and methods, AnomalyIQ represents a genuine effort to apply computer science to a practical problem in Nigerian financial institutions and beyond.

REFERENCES

Ahmed, M., Mahmood, A. N., and Islam, M. R. (2020). A survey of anomaly detection techniques in the financial domain. Future Generation Computer Systems, 55(1), 278–288.

An, J., and Cho, S. (2015). Variational autoencoder-based anomaly detection using reconstruction probability. Special Lecture on IE, 2(1), 1–18.

Arik, S. O., and Pfister, T. (2021). TabNet: Attentive interpretable tabular learning. Proceedings of the AAAI Conference on Artificial Intelligence, 35(8), 6679–6687.

Aros, J., Fuentealba, P., and Ortega, A. (2024). A hybrid anomaly detection model combining unsupervised and supervised stages for financial fraud classification. Journal of Financial Data Science, 6(2), 110–129.

Association of Certified Fraud Examiners. (2022). Report to the nations: 2022 global study on occupational fraud and abuse. ACFE.

Baesens, B., Van Vlasselaer, V., and Verbeke, W. (2021). Fraud analytics using descriptive, predictive, and social network techniques. Journal of the Operational Research Society, 72(4), 861–880.

Bhattacharyya, S., Jha, S., Tharakunnel, K., and Westland, J. C. (2011). Data mining for credit card fraud: A comparative study. Decision Support Systems, 50(3), 602–613.

Bolton, R. J., and Hand, D. J. (2002). Statistical fraud detection: A review. Statistical Science, 17(3), 235–249.

Candès, E. J., Li, X., Ma, Y., and Wright, J. (2011). Robust principal component analysis. Journal of the ACM, 58(3), 1–37.

Carcillo, F., Dal Pozzolo, A., Le Borgne, Y. A., Caelen, O., Mazzer, Y., and Bontempi, G. (2019). Scarff: A scalable framework for streaming credit card fraud detection with Spark. Information Fusion, 41(1), 182–194.

Central Securities Clearing System. (2023). Annual report on electronic trading volume in Nigerian capital markets. CSCS Nigeria.

Chalapathy, R., and Chawla, S. (2019). Deep learning for anomaly detection: A survey. arXiv preprint arXiv:1901.03407.

Chandola, V., Banerjee, A., and Kumar, V. (2009). Anomaly detection: A survey. ACM Computing Surveys, 41(3), 1–58.

Chawla, N. V., Bowyer, K. W., Hall, L. O., and Kegelmeyer, W. P. (2002). SMOTE: Synthetic minority over-sampling technique. Journal of Artificial Intelligence Research, 16(1), 321– 357.

Chen, R., Lazer, D., and Pentland, A. (2022). Scaling compliance operations in high-volume financial transaction environments. Journal of Financial Regulation, 8(1), 45–67.

Dal Pozzolo, A., Caelen, O., Johnson, R. A., and Bontempi, G. (2015). Calibrating probability with undersampling for unbalanced classification. IEEE Symposium Series on Computational Intelligence, 159–166.

Fiore, U., De Santis, A., Perla, F., Zanetti, P., and Palmieri, F. (2019). Using generative adversarial networks for improving classification effectiveness in credit card fraud detection.
Information Sciences, 479(1), 448–455.

Géron, A. (2019). Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow (2nd ed.). O'Reilly Media.

Jurgovsky, J., Granitzer, M., Ziegler, K., Calabretto, S., Portier, P. E., He-Guelton, L., and Caelen, O. (2018). Sequence classification for credit card fraud detection. Expert Systems with Applications, 100(1), 234–245.

Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., Ye, Q., and Liu, T. Y. (2017). LightGBM: A highly efficient gradient boosting decision tree. Advances in Neural Information Processing Systems, 30, 3149–3157.

Kingma, D. P., and Welling, M. (2014). Auto-encoding variational Bayes. International Conference on Learning Representations (ICLR).

LeCun, Y., Bengio, Y., and Hinton, G. (2015). Deep learning. Nature, 521(7553), 436–444.

Lim, B., Arik, S. O., Loeff, N., and Pfister, T. (2021). Temporal fusion transformers for interpretable multi-horizon time series forecasting. International Journal of Forecasting, 37(4), 1748–1764.

Liu, F. T., Ting, K. M., and Zhou, Z. H. (2008). Isolation forest. 8th IEEE International Conference on Data Mining, 413–422.

Lopez-Rojas, E. A., Elmir, A., and Axelsson, S. (2016). PaySim: A financial mobile money simulator for fraud detection. 28th European Simulation and Modelling Conference, 249– 255.

Lundberg, S. M., and Lee, S. I. (2017). A unified approach to interpreting model predictions. Advances in Neural Information Processing Systems, 30, 4765–4774.

Ngai, E. W. T., Hu, Y., Wong, Y. H., Chen, Y., and Sun, X. (2011). The application of data mining techniques in financial fraud detection. Decision Support Systems, 50(3), 559–569.

Pumsirirat, A., and Yan, L. (2018). Credit card fraud detection using deep learning based on an auto-encoder and a restricted Boltzmann machine. International Journal of Advanced Computer Science and Applications, 9(1), 18–25.

Ruff, L., Vandermeulen, R., Goernitz, N., Deecke, L., Siddiqui, S. A., Binder, A., Müller, E., and Kloft, M. (2018). Deep one-class classification. 35th International Conference on Machine Learning, 4393–4402.

Sakurada, M., and Yairi, T. (2014). Anomaly detection using autoencoders with nonlinear dimensionality reduction. MLSDA 2014 Workshop on Machine Learning for Sensory Data Analysis, 4–11.

Securities and Exchange Commission Nigeria. (2023). Capital market master plan: Digital transformation and surveillance framework. SEC Nigeria.

Zhou, C., and Paffenroth, R. C. (2017). Anomaly detection with robust deep autoencoders. 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 665– 674.

Zong, B., Song, Q., Min, M. R., Cheng, W., Lumezanu, C., Cho, D., and Chen, H. (2018). Deep autoencoding Gaussian mixture model for unsupervised anomaly detection. International Conference on Learning Representations (ICLR).