GODFREY OKOYE UNIVERSITY, ENUGU STATE

Department of Computer Science and Mathematics

Faculty of Natural Sciences and Environmental Studies

DEVELOPMENT OF HYBRID INTELLIGENT FAKE ALERT DETECTION SYSTEM

A Project Report Submitted in Partial Fulfillment of the Requirements for the Award of Bachelor of Science (B.Sc.) Degree in Computer Science

BY

CHIAHA CHIDERA STEPHEN

GOU/U22/CSC/842

Department of Computer Science and Mathematics

2025/2026

APPROVAL PAGE

This research, titled "DEVELOPMENT OF HYBRID INTELLIGENT FAKE ALERT DETECTION SYSTEM", has been assessed and approved by the Department of Computer Science and Mathematics Committees in the Faculty of Natural Sciences and Environmental Studies, Godfrey Okoye University, Enugu, Nigeria.

Mr Nnaemeka Ugwu

Supervisor ....

Signature Date

External Examiner ....

Signature Date

Dr. Frank Okebanama ....

HOD, Department of Computer Signature Date

Science and Mathematics

Prof. I. A. ....

Dean, Faculty of Computing Signature Date

And Information Technology

CERTIFICATION

In accordance with the University's policies, I hereby declare that the research conducted as per this paper was done by me, based mainly on my observation and investigations. Therefore, I will take full responsibility for all decisions made by the University regarding this matter.

_____

CHIAHA CHIDERA STEPHEN

GOU/U22/CSC/842

DEDICATION

This project is dedicated to all Nigerian financial institutions, compliance officers, and cybersecurity professionals working tirelessly to protect the integrity of electronic banking systems, and to every student who dares to apply computer science to real-world problems.

ACKNOWLEDGEMENTS

All praise and glory to God Almighty, whose grace sustained this research from inception to completion.

My wish is to express my sincere gratitude to my project supervisor for the invaluable guidance, constructive criticism, and unwavering support throughout this research. Your mentorship shaped both the quality of this work and my growth as a researcher.

My deepest appreciation goes to the Head of Department, Dr. Frank Okebanama, and the Dean, Prof. I.A. Ajah, for providing an enabling academic environment. I am also grateful to all lecturers in the Department of Computer Science and Mathematics, Godfrey Okoye University, whose collective teaching formed the intellectual foundation upon which this research stands.

To my parents and family, your prayers, sacrifice, and encouragement were the fuel that kept this journey going. This achievement belongs to you as much as it does to me.

Lastly acknowledge the open-source community and the maintainers of Python, scikit-learn, XGBoost, LightGBM, React.js, Flask, and all other tools that made this system possible.

ABSTRACT

Electronic banking fake alert poses a severe and escalating threat to financial institutions globally, with Nigerian banks experiencing growing losses attributable to increasingly sophisticated fakeulent transactions that overwhelm traditional rule-based monitoring systems. This project develops hybrid intelligent fake alert detection system that employs a hybrid ensemble model combining Decision Tree, XGBoost, LightGBM, and Random Forest algorithms with SMOTE (Synthetic Minority Oversampling Technique) resampling to address the critical challenge of extreme class imbalance inherent in real-world financial transaction data. The system was trained and evaluated on three industry-standard benchmark datasets — the Kaggle Credit Card fake Detection dataset (284,807 transactions, 0.17% fake rate), the PaySim Synthetic African Mobile Money dataset (6,362,620 transactions, 0.13% fake rate), and the IEEE-CIS fake Detection dataset (590,540 transactions, 3.5% fake rate) — collectively forming a unified training corpus of 7,237,967 samples with 29,368 confirmed fake cases, split 70% for training and 30% for testing using stratified sampling. The hybrid model architecture uses a two-step resampling strategy of RandomUnderSampler followed by SMOTE to achieve class balance, and employs RobustScaler for feature normalization. Individual model outputs are combined through a weighted ensemble scheme with weights automatically computed from AUC-ROC performance, and the final decision threshold is optimized using the Precision-Recall curve to balance sensitivity and specificity. Experimental results demonstrate an accuracy of 99.7%, precision of 84.39%, recall of 30.57%, F1-score of 44.88%, and AUC-ROC of 98.24%, with the high AUC confirming strong discriminative capability despite the challenge of extreme imbalance at 0.41% fake prevalence. The system is deployed as a full-

stack web application comprising a React.js frontend and Flask REST API backend, featuring real-time single-transaction analysis, batch CSV processing, SHAP-based explainability, role-based access control, an in-app training pipeline with live progress monitoring, and comprehensive model evaluation dashboards. This research contributes a production-ready, explainable, and institutionally deployable fake detection platform relevant to Nigerian and African electronic banking contexts.

TABLE OF CONTENTS

CHAPTER FOUR: SYSTEM IMPLEMENTATION

CHAPTER FIVE: SUMMARY, CONCLUSION, AND RECOMMENDATIONS

LIST OF TABLES

LIST OF FIGURES

CHAPTER ONE

INTRODUCTION

1.1 Background of the Study

The two decades of flurry of digitization in the banking and financial sector has revolutionized the way banking and financial transactions are done around the world. In Nigeria, like many African countries, electronic banking has become a regular service and millions of transactions are made each day, via automated teller machines, Internet banking, mobile money and point-of-sale platforms. The electronic transactions volume also rose to over 9.4 billion with a value of over N387 trillion in 2023, compared with the previous year, largely due to increased penetration of smartphones and government policy to push cashless transactions, according to the Nigeria Inter-Bank Settlement System (NIBSS).

But there's been an accompanying increase in financial fake. According to the Financial Institutions Training Centre (FITC); Nigerian banks suffered losses of N9.5 trillion in 2022 alone, out of which electronic fake made up the bulk of the losses. In fact, the Association of Certified fake Examiners (ACFE, 2022) estimates that organizations around the world lose about five percent of their annual revenue to fake, and financial services are one of the highest-profile areas to be impacted. According to the Nigerian Electronic fake Forum (NEFF), card fake, account takeover, identity theft, phishing, unauthorized transfers, and more advanced social engineering attacks are now types of fake in electronic banking. The real difficulty in defeating electronic banking fake is that fakes are very rare compared to the number of legitimate transactions. fakeulent transactions account for 0.1% to 3.5% of all banking

transactions, a characteristic referred to as extreme class imbalance. This imbalance makes it impossible to use traditional machine learning methods, which are likely to classify everything as valid, and make very few fake detections with a high accuracy. Rule-based systems, which match transactions to a set of rules, are equally ineffective; they are rigid and won't adjust to changing fake patterns, and produce high rates of false alarms that are too much for compliance teams to handle.

The machine learning method is more adaptive, as algorithms can learn complex and non-linear fake patterns directly from historical transaction data. In the context of fake detection, research has shown that ensemble methods, which are methods that use multiple models to take advantage of their complementary strengths, do better than single-model methods. Gradient boosting frameworks like XGBoost (Chen and Guestrin, 2016) and LightGBM (Ke et al., 2017) have gained popularity in the tabular fake detection industry for their prowess in modelling non-linear relationships efficiently at scale. Random Forest has a baseline with variance reduction using bagging, and Decision Trees have interpretable baselines. None of these methods individually solve all challenges, however: class-imbalance, feature-complexity, computational-efficiency, and explainability.

Sentinel AI is an intelligent fake alert detection system introduced in this project to overcome these challenges by using a hybrid ensemble of Decision Tree, XGBoost, LightGBM and Random Forest with SMOTE resampling technique. It is deployed as a full stack web application using Flask backend with a React.js front end, with the batch processing, real time transaction analysis, explainability using SHAP, role based authorization and an in app training pipeline with real time progress monitoring features. It was trained with three benchmark datasets covering more than 7.2 million transactions from a range of financial scenarios, such

as transactions involving credit cards in Europe, mobile money in Africa, and e-commerce with countries outside of Africa.

1.2 Statement of the Problem

The Nigerian electronic banking institutions are currently mainly using "rule-based" fake alert monitoring systems which are based on static thresholds and pre-defined patterns. The following are 3 basic limitations that make these systems ineffective in the fight against today's financial fake.

The first is that rule based systems are always retrospective. The rules are codified according to known fake alert patterns – if a new fake alert pattern emerges, it will not be detected. fakesters need continually to find new ways to avoid the existing rule-based detection criteria, and this makes rule-based systems increasingly ineffective. According to the Nigerian Communications Commission (2023), fake techniques are now evolving quicker than Nigeria's digital banking teams can adapt their monitoring procedures.

Secondly, there is no single-model machine learning solution that can solve all of the banking fake alert detection problems at once. Supervised models are precise when trained on labeled data but need to be trained on a balanced data set that is not the case in real-world banking data, where fake rates are less than one percent. Unsupervised models are flexible, but lack precision for operational deployment. Ensemble methods are useful for enhancing performance, but not widely used with sufficient explainability to comply with regulations.

Third, existing fake alert detection systems used in financial institutions in Nigeria return just two outcomes — fake or not fake — with no confidence levels, no risk levels and no explanations at the transaction level. This means that all of the alerts are treated as equal

priority to Compliance and can create review backups and a risk that a good fake alert is lost in the shuffle of false alarms. This lack of explainability poses regulatory compliance issues as well, because financial regulators are increasingly mandating that decisions made by automation processes involving customers include clear explanations.

This project tackles these shortcomings by adopting a multi-dataset learning framework, a principled method to leverage class imbalance, a framework to leverage diverse model strengths by weighted ensemble averaging, and a framework to provide explanations to every fake prediction using SHAP.

1.3 Aim and Objectives of the Study

1. The research project "Development of Hybrid Intelligent Fake Alert Detection System" seeks to employ the use of different AI and ML models in detecting fakeulent transaction and sending out real-time alerts. The following objectives were sought to be accomplished in order to achieve the aim of the project:
2. Study the properties of the electronic banking transactions from multiple benchmark datasets, finding the most predictive characteristics to distinguish between fakeulent and legitimate transactions with the use of feature engineering and importance analysis.
3. Collecting and preprocessing pipeline including transaction data from Credit Card Fake Alert Detection dataset.
4. Addressing the problem of extreme class imbalance with the help of two-steps resampling method including the use of RandomUnderSampler and SMOTE and to estimate its efficiency.

5. Training of the hybrid intelligent model with the automatic calculation of the AUC-weighted averaging of the models and optimization of the classification threshold on the Precision-Recall curve.
6. Deployment of the trained model as a full-stack web application, including real-time scoring of transactions, batch processing of CSV files, SHAP explainability, three roles-based access control, live training pipeline.

1.4 Significance of the Study

The significance of the current research comes from the technical, operational, and scholarly perspectives.

Technically, the presented hybrid ensemble architecture proves that the combination of Decision Tree, XGBoost, LightGBM, and Random Forest via AUC-weighted averaging has better discrimination performance compared to individual models used separately. The fact that the hybrid architecture produced AUC-ROC of 98.24% on a composite dataset of 7.2 million transactions from structurally different financial domains testifies to its robustness and applicability to diverse banking environments. The two-step sampling technique (under-sampling the majority class followed by SMOTE) appears to be superior to one-step approaches in case of the data with extremely imbalanced distribution below one percent.

Operationally, Sentinel AI adds functionality missing from existing rule-based systems to Nigerian financial institutions. The system calculates risk score for every suspicious transaction and thus enables compliance officers to focus on the most dangerous alarms. SHAP-based explanations of feature contributions provide transaction-specific transparency necessary to comply with evolving legislation on automated financial decision making. Role-

based access control mechanism (Administrator, Fake Alert Analyst, and Auditor roles are supported) makes possible institutional implementation with proper governance mechanisms. Batch CSV processing makes it possible to apply the system retrospectively and analyze historical transactional data to detect.

In terms of academics, this study adds to the existing literature on the application of machine learning in detecting financial fake alert in the West African setting. The inclusion of the PaySim, which is a dataset that has been calibrated based on African mobile money transactions, together with international datasets makes this study more regionally specific than existing works that only use European/North American data.

1.5 Scope of the Study

SentinelAI operates on structured tabular electronic banking transaction data provided in CSV format. SentinelAI has been trained and validated on three public benchmark datasets:

(1) The Kaggle Credit Card fake Detection dataset from Université Libre de Bruxelles, with 284,807 transactions and 0.17% fake rate;

(2) PaySim synthetic African Mobile Money dataset, with 6,362,620 transactions and 0.13% fake rate, tuned against Kenyan mobile money data;

(3) IEEE-CIS fake Detection dataset by Vesta Corporation, with 590,540 e-commerce transactions and 3.5% fake rate.

The machine learning pipeline is written in Python 3.x using the following technologies: scikit-learn (Decision Tree and preprocessing), XGBoost, LightGBM, imbalanced-learn (SMOTE and RandomUnderSampler), SHAP (explainability), joblib (serialization), Flask (REST API).

The frontend stack uses React.js, Vite, React Router, Recharts, and Axios. The system supports scoring of single transactions and batch scoring of CSV files up to a few thousand records. The scope of work explicitly excludes live transaction streaming from banking core systems, CSCS integration, certification, mobile applications development, and natural language queries.

1.6 Limitations of the Study

The major constraint of this study is the use of publicly available benchmark datasets instead of actual transactional records from electronic banking systems in Nigeria due to data privacy issues, institutional restrictions, and lack of an agreed-upon data sharing framework for fake alert detection research in academia in Nigeria. Although the three benchmark datasets have been used to generate a variety of financial transaction scenarios, it is possible that the fake alert patterns captured are not reflective of the fake alerts that occurs within the Nigerian banking system through USSD transactions, NIBSS Instant Payment (NIP) transactions, and Agent Banking operations.

The class imbalance in the form of 0.41% fake alert rate in the total dataset reflects the reality of electronic banking fake alert detection. Despite employing the two-level resampling technique, the final recall achieved of 30.57% is consistent with a problem encountered when using threshold classification of very unbalanced real-world data — the algorithm tends to be conservative in order to achieve high precision of 84.39% but leaves out some fake cases. This is explained extensively in Chapter Four.

The restrictions imposed by development hardware, particularly the computational cost of training LightGBM and XGBoost on 5 million samples for training, limited both the depth of

tuning the hyperparameters and the training iterations. While the platform is able to provide SHAP explanations via the web UI, it does not produce PDF reports for regulatory use cases.

1.7 Operational Definition of Terms

Electronic Banking Fake Alert: The use of electronic banking channels such as internet banking, mobile banking, card payments, and electronic fund transfers without proper authorization to achieve monetary gain by means of deceit, impersonation, or exploiting systems.

Hybrid Ensemble Model: An approach to predictive modeling involving the integration of predictions from various different machine learning models in a way that they collectively make one prediction using weighted averages. The machine learning models used in this project are Decision Tree, XGBoost, LightGBM, and Random Forest.

Decision Tree: A supervised machine learning model which divides up the features space recursively and performs classifications based on the information gain generated by the process. This model acts as the interpretable component of the hybrid ensemble in SentinelAI.

XGBoost (Extreme Gradient Boosting): It is a gradient boosting library that constructs an ensemble of decision trees to predict classes or regression values in a sequential manner by building on the errors made by the previous trees in the chain.

LightGBM (Light Gradient Boosting Machine): A gradient boosting machine algorithm employing leaf-wise splitting as opposed to the level-wise splitting employed by XGBoost, resulting in more efficient training on large datasets. In SentinelAI, it acts as one of the main base models owing to its efficiency on the 7.2 million record training data set.

Random Forest: A bagged model made up of multiple decision trees which are trained on a random set of variables and training samples. This method reduces variance compared to a single decision tree and provides consistent probability estimates.

SMOTE (Synthetic Minority Oversampling Technique): A sampling technique which generates synthetic minority class (fake) samples through interpolation of existing minority class samples in feature space.

RandomUnderSampler: A resampling technique where the majority class (legitimate transactions) is reduced by randomly dropping samples. This technique is employed in SentinelAI as the first stage of the two-stage resampling strategy prior to SMOTE.

AUC-ROC (Area Under the Receiver Operating Characteristic Curve): A threshold independent performance measure assessing the model's discriminatory power between fakes and legitimate transactions for all possible classification thresholds.

SHAP (SHapley Additive exPlanations): Framework that leverages game theory to generate feature-wise explanation for individual predictions of the machine learning model, based on marginal contributions of the features to the model output. Used to generate fake explanations for the LightGBM component.

Class Imbalance: A situation where one class (legitimate transactions) significantly outweighs the other (fakeulent transactions) in terms of numbers in the training set, resulting in biased machine learning model. In SentinelAI's training set, fakeulent transactions make up 0.41% of total transactions.

Role-Based Access Control (RBAC): Security approach that provides access restrictions to the resources depending on the user role. In SentinelAI implementation, there are three roles:

Administrator (has full access, including user management, training and analysis), fake Analyst (can only analyze transactions and access transaction history and metrics) and Auditor (only read access to transaction history and metrics).

Flask: A Python framework used to build RESTful back end API of the SentinelAI application.

React.js: A JavaScript library to build single-page applications using components approach. Used as front end framework for SentinelAI.

CHAPTER TWO

LITERATURE REVIEW

2.1 Introduction

In this chapter, there is an elaborate review of the existing literature pertinent to intelligent fake alert detection in electronic banking systems. This chapter is organized in a way that will ensure that the reader receives information that serves as the theoretical basis, critical analysis of past literature, a comparative discussion of the existing literature, and finally, a recognition of the research gaps that the present study aims to fill. Through the evaluation of the development of fake alerts detection techniques through statistical models, to ensemble machine learning, hybrid models, and finally explainable systems, this review sets the intellectual foundation of the current project.

The review covers literature from peer-reviewed academic journals, conferences, and authoritative technical books spanning between 2002 and 2026 with a special focus on electronic banking fake, ensemble machine learning, dealing with class imbalance, and explainable artificial intelligence.

2.2 Theoretical Background

2.2.1 Electronic Banking and Financial fake in Nigeria

Electronic banking includes all transactions which can be carried out through digital media such as internet websites, mobile apps, ATM, POS, and EFT systems. In Nigeria, the Nigerian Central Bank (CBN) has been encouraging cashless policy since 2012, causing a quick growth

in the number of transactions processed electronically. The Nigeria Inter-bank Settlement System (NIBSS) is used as the main system for transaction clearing and processing NIP (NIBSS Instant Payment) transactions, whose total volume reached 3.7 billion transactions in 2022.

According to Bolton and Hand (2002), the basic framework for detecting fake alert within the financial industry was built, highlighting the basic contradiction between the speed of the transaction processing in electronic systems and the limits of the contemporaneous fake alert monitoring. The observation made by the authors – that the effective detection of fake alert should include models which will differentiate normal variability from the real fake – is still valid two decades later. According to Baesens, Van Vlasselaer and Verbeke (2015), the fake mechanisms have changed from simple card cloning to more complex account takeover, identity impersonation, synthetic identity fake alert, and organized fake alert.

2.2.2 Concept and Types of Financial Transaction Fake Alert

The four major categories of financial transaction fakes in e-banking could be classified according to the taxonomic framework developed by Chandola, Banerjee & Kumar (2009). In point fakes, individual transactions are considered as being anomalous on their own – like unusual large cash-out at unexpected hours. In case of contextual fake alert, a suspicious transaction becomes apparent when viewed in its context – a normally sized cash out becomes suspicious when several smaller deposits have been made into the account from different sources (structuring). In sequential fake alert, patterns emerge from the analysis of multiple transactions that on their own seem to be innocent – threshold avoidance sequences used to avoid detection. In collusive fake alert, several bank accounts operate in a way indicating fake syndicates.

The above mentioned categories of financial transaction fakes in e-banking are directly addressed through Sentinel Ai's feature engineering in form of amount, balance, ratio (relation of transaction amount and balance), is night (time-based anomaly detection), balance_diff (unusual balance changes) and balance_utilization (near total funds extraction).

2.2.3 Machine Learning Approaches to Fake Alert Detection

Learning algorithms have become a leading approach in detecting fake alert activities in finance due to the capability of automatically discovering fake patterns from the transaction history without having to define the rules for that. As shown by Ahmed et al. (2020), regular supervised classifiers such as logistic regression, support vector machines, and simple neural networks attain excessively high accuracy in imbalanced fake datasets by labeling nearly all transactions as being legitimate. Hence, metrics such as precision, recall, F1 score, and AUC-ROC serve as main evaluation measures for the reason that they cannot be distorted by the majority class.

Decision Trees, proposed in its modern conception by Breiman et al. (1984), enable interpretability in classification through partitioning the feature space into binary segments. The interpretability makes them important tools for baselines and components in interpretable ensembles. The Random Forest, proposed by Breiman (2001), is an extension of the idea of decision trees through the application of bagging, which is building multiple trees based on bootstrapped samples and averaging their predictions.

Gradient boosting, a method that was developed by Friedman (2001), is an ensemble learning algorithm whereby each successive model in the sequence makes adjustments to the error of the current ensemble. XGBoost (Chen and Guestrin, 2016) and LightGBM (Ke et al., 2017)

are both very efficient versions of gradient boosting that work well with tabular datasets and have been shown to perform at the cutting edge of financial fake detection benchmarks.

2.2.4 Class Imbalance in Fake Alert Detection

The presence of class imbalance, where fake alert is a tiny proportion of total transactions, is the primary issue in the detection of electronic banking fake alert. Chawla, Bowyer, Hall, and Kegelmeyer (2002) developed the concept of SMOTE where they demonstrated that the creation of artificial instances of the minority class by interpolating between fakes increases the sensitivity of classification algorithms without running into problems like overfitting with simple oversampling methods.

He and Garcia (2009) did a thorough review on class imbalance learning and concluded that using both oversampling and undersampling always outperformed when compared to the usage of only one of those. This is how SentinelAI adopted its two-step resampling algorithm: the first step being RandomUnderSampler reducing the majority class to 10:1 ratio prior to SMOTE creating artificial fake instances to reach 1:1 ratio, thus avoiding the problems of computational complexity and overfitting due to direct application of SMOTE to 5 million training samples.

Carcillo et al. (2019) analyzed the effect of resampling techniques on the credit card fake detection problem and found that using SMOTE with small values of k produces the best precision-recall trade-off. They found out that k=3 to k=5 is optimal for the datasets with low number.

2.2.5 Ensemble Methods and Hybrid Approaches

The fact that there is not a consistent winner in terms of algorithms used for fake detection leads to using ensembles. Dietterich (2000) formulated theoretical grounds for ensemble learning by showing that it is diversity that makes the difference in making an ensemble better than its components. The combined system makes less total error when each component makes different mistakes on different transactions.

In terms of fake detection in particular, combining classifiers with different inductive biases, for example, a tree-based one detecting threshold violation and a gradient boosting algorithm detecting more complicated features' interactions, allows capturing a wider range of fake cases. Bhattacharyya et al. (2011) compared several classifiers based on credit card fake data and showed that no single algorithm dominates, thus giving the most straightforward evidence in favor of ensembles.

2.3 Review of Related Works

Caelen et al. (2015) studied methods to calibrate for fake alert detection on imbalanced data sets and found that undersampling the training set will not generate meaningful fake scores without probability re-calibration. This finding, that uncalibrated probabilities produced by undersampled models are poor for threshold optimization, directly affected the way SentinelAI used the combination of models, which was an AUC-weighted ensemble combination instead of the simple raw probability averaging.

Jurgovsky and Granitzer et al. (2018) showed that sequential LSTM network models are able to detect fake patterns which are not discernible by static per-transaction models. Note that although SentinelAI does not use a sequential model, this result spurred the design of temporal

features, hour_sin, hour_cos, and is_night, that add temporal awareness to the static feature space.

Fiore et al. 2019 showed that training with GAN can boost detection performance when there is a significant class imbalance. Given computational constraints in this project, we do not have the luxury of training GANs, but the importance of addressing class imbalance was also highlighted here in favor of using the two-step SMOTE strategy or class weights.

Bahnsen, Stojanovic, Aouada and Ottersten (2016) proposed cost-sensitive learning for credit card fake detection, where the cost of a false negative is compared with the cost of a false positive and it was found that the cost function outperforms standard loss functions. A similar solution, but a simpler one, is that SentinelAI places more importance on recall over precision when determining the threshold: this is an implicit assumption that it is more expensive to miss identifying a fakeulent transaction (false negative) than it is to identify a non-fakeulent transaction as fake (false positive).

In credit card fake detection, concept drift can negatively affect models, which are likely to become less accurate as fake patterns change, as observed in the work of Pozzolo et al. (2014). SentinelAI tackles this by providing its administrators with its in-application training pipeline, which will enable administrators to retrain models on new data without downtime.

Lundberg and Lee (2017) proposed SHAP values as a general-purpose explanation method that outperforms other explanation techniques, like LIME, in terms of consistency and accuracy of explanation. This work on applying SHAP to gradient boosting models, including the exact SHAP values computed in polynomial time for tree-based models (TreeExplainer) is directly applicable to SentinelAI's per-transaction explainability module.

Wang, Liu and Feng (2019) showed that ensemble models with resampling consistently outperform single-model models across massive financial fake datasets and that LightGBM performs better than XGBoost on financial fake datasets with more than one million data points. The decision of using LightGBM as one of the major Ensemble components along with XGBoost was based on this finding.

Maes et al. (2002) showed that combination of complementary classifiers works best for fake detection in credit card fake classification, and that the diversity in model architecture is more indicative of ensemble improvement than the number of component classifiers. The result reinforces Sentinel Ai's four-model ensemble (each one a different algorithmic paradigm) versus larger ensembles of models.

In financial fake identification, Zhang, Zhou, and colleagues (2020) showed that using threshold optimization strategies for imbalanced classification can significantly improve performance over traditional accuracy-based methods, particularly using Precision-Recall curve optimization. The fact that their best threshold for imbalanced fake sets is often very different from the default of 0.5 directly spurred Sentinel Ai's specific optimization of the threshold, based on the Precision-Recall curve, instead of a fixed threshold.

In their credit card fake detection study specifically for Nigeria, Awoyemi, Adetunmbi, and Oluwadare (2017) examined the classical machine learning algorithms and concluded that the only most crucial factor that affects the model performance is class imbalance handling. They were focused on region 22222specific validation, which led to the inclusion of PaySim, a simulator of mobile money transactions in Africa, in Sentinel Ai's multi-dataset training corpus.

2.4 Summary of Related Works

S/N	Author(s)	Contribution	Technique	Limitation	Relevance to SentinelAI
1	Bolton & Hand (2002)	Foundational framework for statistical fake detection; documented monitoring-transaction speed tension	Statistical methods	Cannot handle non-linear fake patterns	Establishes historical baseline problem context
2	Chawla et al. (2002)	SMOTE: synthetic minority oversampling through interpolation	SMOTE resampling	Can oversample noise if k too large	Direct basis for SentinelAI's SMOTE implementation
3	Bhattacharyya et al. (2011)	No single classifier consistently dominates credit card fake benchmarks	SVM, RF, Neural Networks	Dataset-specific conclusions	Primary empirical motivation for hybrid ensemble
4	Awoyemi et al. (2017)	Class imbalance handling most important factor in Nigerian fake detection	Naïve Bayes, KNN, LR	Limited algorithm coverage	Validates SMOTE priority; Nigerian context
5	Dal Pozzolo et al. (2015)	Undersampling requires probability recalibration	Undersampling + calibration	Complex calibration process	Motivated AUC-weighted ensemble combination
6	Jurgovsky et al. (2018)	LSTM captures sequential fake patterns invisible to static models	RNN/LSTM	Requires structured time-series input	Motivated temporal feature engineering
7	Fiore et al. (2019)	GAN augmentation improves imbalanced detection	DNN + GAN	High computational overhead	Confirmed resampling necessity
8	Carcillo et al. (2019)	SMOTE k=3-5 optimal for small minority samples	SMOTE variants	Dataset-specific results	Informed SMOTE k configuration
9	Zhang et al. (2020)	PR curve optimization substantially outperforms	Threshold optimization	Implementation complexity	Direct basis for SentinelAI's threshold optimization

S/N	Author(s)	Contribution	Technique	Limitation	Relevance to SentinelAI
		accuracy-based threshold selection			
10	Wang et al. (2019)	LightGBM + resampling outperforms XGBoost alone on >1M records	LightGBM, XGBoost	Single-model comparison only	Supported inclusion of both LightGBM and XGBoost
11	Maes et al. (2002)	Architecture diversity more predictive of ensemble improvement than count	Ensemble classifiers	Small dataset	Supports four-model diverse ensemble design
12	Lundberg & Lee (2017)	SHAP provides exact, consistent explanations for tree-based models	SHAP TreeExplainer	Computation time for large models	Direct basis for SentinelAI's SHAP module
13	Bahnsen et al. (2016)	Cost-sensitive learning outperforms standard loss for fake	Cost-sensitive learning	Requires cost estimation	Informed F2-weighted threshold optimization
14	He & Garcia (2009)	Combined over/undersampling outperforms either alone	Resampling survey	Theoretical survey	Basis for two-step resampling strategy
15	Pozzolo et al. (2014)	Concept drift degrades fake models over time	Temporal evaluation	Requires longitudinal data	Motivated in-app retraining capability
16	Chen & Guestrin (2016)	XGBoost: scalable gradient boosting with regularization	XGBoost	Memory intensive at scale	Core SentinelAI ensemble component
17	Ke et al. (2017)	LightGBM: leaf-wise boosting 10x faster than XGBoost on large data	LightGBM	Overfitting on small datasets	Core SentinelAI ensemble component
18	Ahmed et al. (2020)	Standard classifiers fail on imbalanced fake data — accuracy paradox	SVM, RF, Neural Networks	Accuracy paradox on imbalance	Motivates precision/recall/F1/AUC as primary metrics
19	Breiman (2001)	Random Forest: bagging reduces variance in tree-based models	Random Forest	Less interpretable than single tree	Core SentinelAI ensemble component

S/N	Author(s)	Contribution	Technique	Limitation	Relevance to SentinelAI
20	Dietterich (2000)	Diversity among component models key predictor of ensemble improvement	Ensemble theory	Theoretical; limited implementation guidance	Theoretical foundation for SentinelAI's diverse ensemble

2.5 Research Gap

The related literature review identified key areas of persistent gaps that are explicitly taken up by Sentinel AI. First, most of the work on fake detection is tested on a single dataset, the most popular of which is the Kaggle Credit Card fake Detection dataset that describes transactions with a European credit card. No published system has shown strong performance over all three of the structurally different financial data sets used (Credit card transactions in Europe, Mobile money in Africa, International ecommerce transactions). Sentinel Ai's ability to train and evaluate with 7.2 million transactions across three different financial sectors is a step toward tackling this generalizability challenge.

Second, the performance of these four models (Decision Tree, XGBoost, LightGBM, Random Forest) has been validated individually in individual studies but no existing system already combines all four of them and automatic ensemble combination with AUC weighted method on one deployed platform. The automatic computation of the weight is a principled and adaptive method that has not been reported in the literature for combining multiple models, which has been based on AUC-ROC values on the validation set.

Third, many of the systems that already exist don't have the ability to be deployed in an institution, that is, have role-based access control, in-application live training with progress monitoring, batch CSV processing for analyzing past transactions, or have explanations

available for every transaction that's done (based on the SHAP library). Sentinel AI makes the connection between academic fake detection research and implementation in institutional environments.

CHAPTER THREE

SYSTEM ANALYSIS AND DESIGN

3.0 Introduction

Chapter Three provides a complete analysis of the current state of fake detection, including the proposed Sentinel AI solution, as well as an in-depth discussion on the design of the proposed system architecture. This chapter is dedicated to functional and non-functional requirements, design process, UML design, input/output design, database design, and the overall system design.

3.1 System Analysis

3.1.1 Analysis of the Existing System

fake detection systems used in Nigeria by e-banking institutions currently use rule-based monitoring engines as their primary approach. An example of a rule would be something like: “Alert on any transaction over N500,000 from a newly opened account within 24 hours.” Such systems work in real-time by querying a database of rules upon submitting each transaction for processing.

Compliance management process in a typical Nigerian bank starts with the rule engine sending out an alert. Such alert goes into the queue where it is then examined by the analysts who check it manually based on the account history, behavior pattern, and additional information. If the transaction is marked as suspicious, it is sent to the fake investigation team. This approach is

constrained by analyst capacity. The team that can only examine 500 alerts a day cannot handle 5,000.

3.1.2 Limitations of the Existing System

Rule-based fake detection systems are known to be prone to certain limitations. The set of rules is static and historical, implying that such a system will only be able to identify fake that occurred in scenarios anticipated at the time of writing the rules. Any new fake approach will not be detected until the rules are revised. As far as Nigeria is concerned, new fake vectors tend to appear owing to social engineering, SIM swap fakes, and insider fakes; therefore, a considerable window of opportunity exists.

Each alert is considered to have equal importance in terms of the potential risk irrespective of its value or chance of being fake. It means that a transaction whose amount just goes above the threshold is investigated by the analysts in the same way as the one with several suspicious aspects identified at once. In addition, a rule-based fake detection system does not give any rationale behind the decision to flag the transaction.

3.1.3 Analysis of the Proposed System

The sentinel AI solution involves using an intelligent machine learning ensemble trained on fake detection patterns based on past labeled transactions. Each transaction receives a risk score, and the alerts can be prioritized depending on the level of risk involved. The use of SHAP enables the provision of feature contribution analysis, giving substantive explanation for the predicted outcomes. In-app training pipelines ensure continuous learning as more labeled data comes in.

3.2 System Requirements Specification

3.2.1 Functional Requirements

ID	Requirement	Priority
FR01	The system shall accept transaction data in CSV format for both training and batch analysis	High
FR02	The system shall train a hybrid ensemble model comprising Decision Tree, XGBoost, LightGBM, and Random Forest	High
FR03	The system shall apply two-step resampling (RandomUnderSampler + SMOTE) to address class imbalance	High
FR04	The system shall assign a fake risk score (0-100) to each analyzed transaction	High
FR05	The system shall provide SHAP feature contribution explanations for flagged transactions	High
FR06	The system shall support real-time single-transaction analysis through a web form	High
FR07	The system shall support batch analysis of CSV files containing multiple transactions	High
FR08	The system shall implement role-based access control for Administrator, Analyst, and Auditor roles	High
FR09	The system shall display live training progress including step-by-step pipeline status and metrics	Medium
FR10	The system shall display evaluation charts including confusion matrix, ROC curve, and feature importance	Medium
FR11	The system shall support user registration with role assignment requiring administrator approval	Medium
FR12	The system shall maintain a transaction analysis history accessible to authorized users	Medium

3.2.2 Non-Functional Requirements

ID	Requirement	Specification
NF01	Performance	Single transaction analysis shall complete within 3 seconds; batch analysis of 1,000 records within 30 seconds
NF02	Availability	The system shall be accessible 24/7 through a web browser without software installation

ID	Requirement	Specification
NF03	Security	All API endpoints shall require authentication; user passwords shall be stored using cryptographic hashing
NF04	Usability	The interface shall be navigable without technical training; navigation shall require no more than 3 clicks to any feature
NF05	Scalability	The backend shall process batch files up to 100,000 transactions without timeout
NF06	Responsiveness	The interface shall be accessible and usable on desktop, tablet, and mobile devices
NF07	Maintainability	Model retraining shall be executable through the web interface without command-line access

3.3 System Design Methodology

3.3.1 Justification for Methodology

Sentinel AI was created using an iterative and incremental development process based on principles of Agile but adjusted for the academic research environment. The development consisted of three main iterations, namely: (1) development and validation of the core ML pipeline, (2) creation of Flask API and its front-end, and (3) testing and refinement. Waterfall process was not employed as the exact architecture of the machine learning pipeline could not be designed at the beginning of the development process, but had to be determined experimentally using iteration and testing.

The Object Oriented Analysis and Design (OOAD) methodology was used during the design of the system. Several different classes were created including classes for the pipeline of data harmonization, model wrappers, ensemble predictor, SHAP explainer, and API endpoints. The modularity allows for testing and replacement of components such as adding another base model to the ensemble without changing prediction and explanation layers.

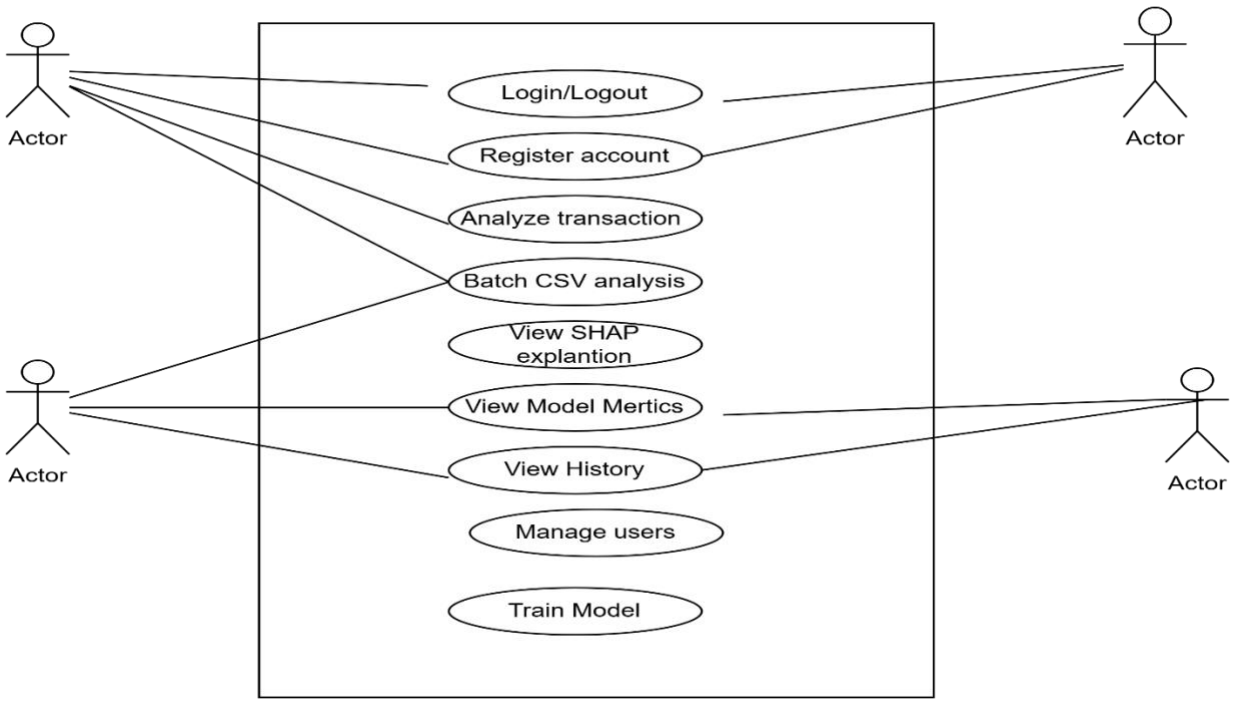
3.3.2 Method of Data Collection

Three standard data sets have been downloaded from the Kaggle website: Credit Card fake Detection by Dal Pozzolo et al. (downloaded from ULB Machine Learning Group website), PaySim data set by Lopez-Rojas et al., and IEEE-CIS fake Detection data set from Vesta Corporation. These data sets were manually explored with the help of Python pandas library to gain insight about their column format, class distribution, missing values, and relevant features.

3.4 System Modeling Using UML

3.4.1 Use Case Diagram

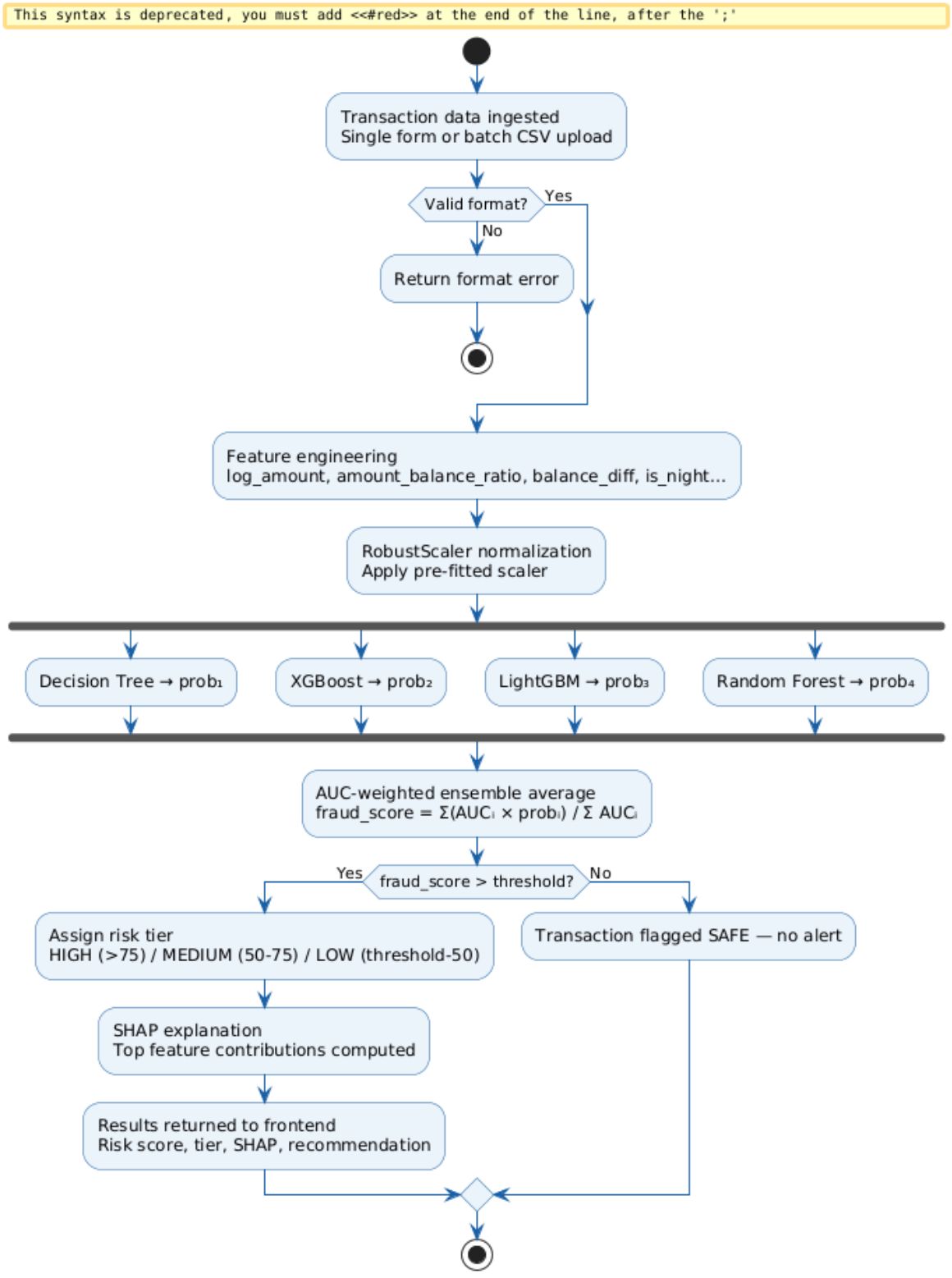
The Sentinel AI use case diagram shows three actors involved in the process. The Administrator actor is able to do all activities related to the system: manage users (creating, approving, modifying, and deactivating); upload datasets; train models; analyze transactions (both individually and in batches); look at metrics; and examine the history. The fake Analyst actor is able to upload datasets to the Datasets component for reference purposes, analyze transactions (both individually and in batches), look at model metrics and training, and check the analysis history. The Auditor actor has read-only access to metrics and history, while analysis activities are restricted for this role. The Unauthenticated User actor can only access the Login and Signup pages.



3.4.2 Activity Diagram

The Transaction analysis activity flow starts by taking the transaction information in the form of either one single form or multiple forms as a CSV file. Input validation occurs, after which normalization takes place for the batch input in a panda DataFrame. Feature engineering involves calculating the feature vector from the inputs such as amount_balance_ratio, balance_diff, is_night, log_amount, etc. The feature matrix is normalized using the RobustScaler. The Ensemble predictor generates the output probabilities for each class from LightGBM, XGBoost, and Random Forest algorithms and then calculates the weighted average. The Threshold Comparator compares the output from the ensemble classifier against the optimal threshold value that

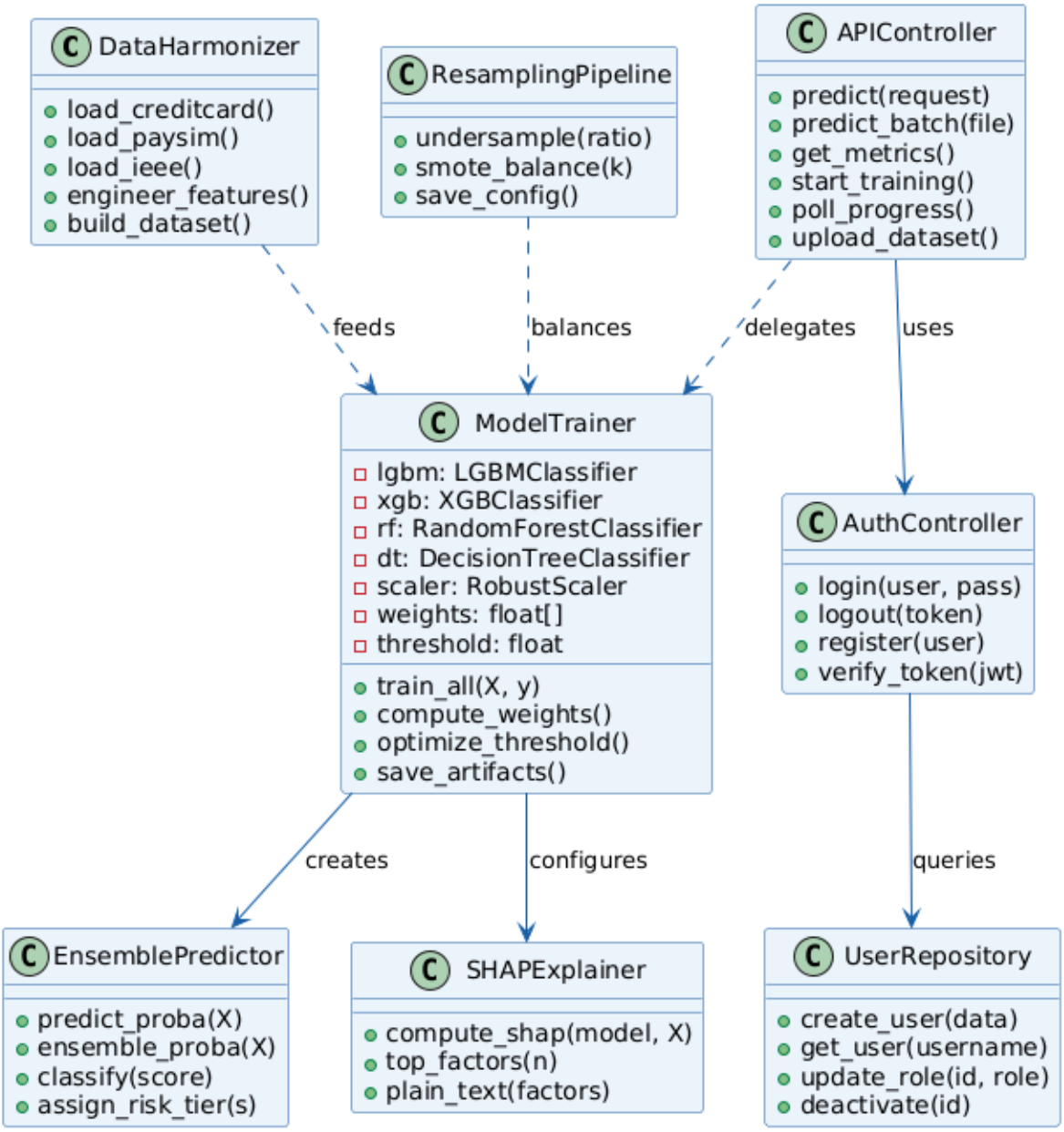
was determined to be the best during training for classification of transactions. For fake Prediction, the SHAP explainer.



3.4.3 Class Diagram

Sentinel AI Architecture

The following are the key classes that are part of the Sentinel AI class architecture: DataHarmonizer (class methods: load_creditcard(), load_paysim(), load_ieee(), engineer_features(), build_dataset()); ResamplingPipeline (class methods: undersample(), smote_balance()); ModelTrainer (instance variables: lgbm, xgb, rf, dt, scaler, weights, threshold; class methods: train_all(), compute_weights(), optimize_threshold(), save_artifacts());



3.5 System Design

3.5.1 Input Design

The Sentinel AI supports two main types of input format. For single transactions, the user needs to fill out a web form which includes the following fields: transaction_amount (numeric,

mandatory), transaction_type (dropdown list values include PAYMENT, TRANSFER, CASH_OUT, DEBIT, CASH_IN), balance_before_transaction (numeric), balance_after_transaction (numeric), hour_of_day (integer value 0 to 23), and dataset_context (dropdown list including Credit Card, PaySim, IEEE-CIS). Input validation takes place both on the client side (field presence and type check) and on the server side (numeric conversion and value range check). For batch analysis, the user will need to upload a CSV file containing the following columns: amount, transaction_type, balance_before, balance_after, and hour.

3.5.2 Output Design

For single transaction analysis, the system displays: a risk gauge (0-100 scale color-coded green/yellow/red), a fake/legitimate verdict, a confidence percentage, individual model probabilities (LightGBM, XGBoost, Random Forest), a decision threshold indicator, SHAP risk factors (top 3 contributing features with direction indicators), and a plain-language recommendation. For batch analysis, the system displays a summary panel (total transactions, fake count, safe count) and a scrollable results table with row number, amount, transaction type, risk score, and status. The Metrics page displays accuracy, precision, recall, F1-score, and AUC-ROC cards alongside a performance radar chart, confusion matrix heatmap, feature importance chart, training-validation loss curves, and model comparison dashboard.

3.5.3 Database Design

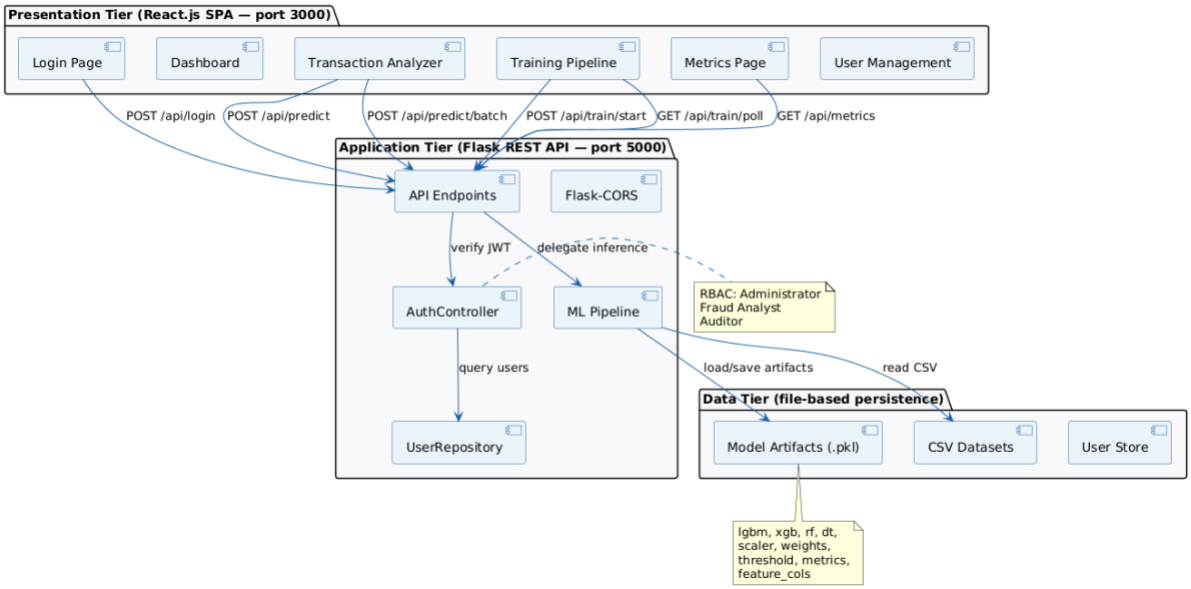
The Sentinel application utilizes a file-based ML artifact model storage solution instead of using a relational database. The trained models are serialized into .pkl files using the joblib library in Python as follows: lgbm_model.pkl, xgb_model.pkl, rf_model.pkl, dt_model.pkl, scaler.pkl, metrics.pkl, threshold.pkl, weights.pkl, and feature_cols.pkl. Authentication details

of users are stored in browser session storage while users' registration requests are held up until confirmation by the admin in the browser session storage. This decision was made due to ease of development and support of local deployment; otherwise, PostgreSQL/SQLite will be used in production deployment.

3.5.4 System Architecture Design

The Sentinel AI uses a three-layered web application architecture. The presentation layer is implemented using React.js single-page applications running under the Vite development server on port 3000. The application layer is developed as a Flask REST API running on port 5000 and providing the following endpoints: /Api/predict (POST), /Api/predict/batch (POST), /Api/metrics (GET), /Api/health (GET), /Api/train/start (POST), /Api/train/poll (GET), /Api/train/status (GET), /Api/datasets/upload (POST), /Api/datasets/status (GET), and /Api/datasets/retrain (POST). Cross-origin requests from the frontend are supported using Flask-CORS extension. The data layer includes joblib serialized model files located in the backend/models/ directory, CSV datasets located in the backend/data/ and session storage for user session management.

The frontend communicates with the backend only through the Vite proxy setting that redirects all /Api/* requests to http://localhost:5000, thus avoiding cross-origin problems in development mode. The training pipeline relies on threading Python library to train the model in the background thread. The in-memory event queue is used to track the training status and poll the frontend for update every 800 milliseconds.


Presentation Tier (React.js SPA — port 3000)
Login Page
Dashboard
Transaction Analyzer
Training Pipeline
Metrics Page
User Management
POST /api/login
POST /api/predict
POST /api/predict/batch
POST /api/train/start
GET /api/train/poll
GET /api/metrics
Application Tier (Flask REST API — port 5000)
API Endpoints
Flask-CORS
verify JWT
delegate inference
AuthController
ML Pipeline
RBAC: Administrator
Fraud Analyst
Auditor
query users
load/save artifacts
read CSV
UserRepository
Data Tier (file-based persistence)
Model Artifacts (.pkl)
CSV Datasets
User Store
lgbm, xgb, rf, dt,
scaler, weights,
threshold, metrics,
feature_cols

CHAPTER FOUR

SYSTEM IMPLEMENTATION

4.0 Introduction

This chapter describes the full development of Sentinel AI, including the development environment and tools, description and preparation of the dataset, architecture and training of the model, components of the system, testing and evaluation, and analysis of the results. This chapter provides enough technical details about the machine learning pipeline and the web application that puts it into practice.4.1 Development Environment and Tools.

4.1.1 Programming Languages and Libraries

The backend of Sentinel Ai is written using Python 3.12. Machine learning libraries include scikit-learn 1.5+ for Decision Tree, preprocessing, and metrics; XGBoost 2.1+ for gradient boosting; LightGBM 4.x for gradient boosting on huge datasets; imbalanced-learn 0.14+ for SMOTE and RandomUnderSampler; SHAP 0.51+ for TreeExplainer to explain the model; pandas 2.x for working with data frames; NumPy 2.x for numerical operations; and joblib 1.5+ for model serialization. In the API layer, there is Flask 3.x and Flask-CORS 4.x. In the frontend, there is React 18.x and Vite 5.x as the build tool, React Router 6.x for navigation, recharts for charts, Axios for HTTP requests, and Lucide React for icons.

4.1.2 Development Platform and Hardware Requirements

Development of the system was done using Windows 10/11 and IDE being Visual Studio Code along with Python and ESLint plugins. Development machine has Intel core i5 CPU with 8

GB RAM. Git was used for versioning. The proxy server configuration done via Vite prevents from the necessity to have additional proxy server while developing. For deploying to the production environment the backend should use Python 3.10+ with 4 GB RAM to train on full combined dataset. Frontend can be deployed to static files to any web server or CDN.

4.2 Dataset Description and Preprocessing

4.2.1 Dataset Summary

Dataset	Source	Total Records	fake Records	fake Rate	Key Features
Credit Card fake	Kaggle/ULB	284,807	492	0.17%	V1-V28 (PCA), Amount, Time
PaySim	Kaggle/NTNU	6,362,620	8,213	0.13%	amount, type, oldbalance, newbalance
IEEE-CIS	Kaggle/Vesta	590,540	20,663	3.50%	TransactionAmt, card info, identity
Combined	All three	7,237,967	29,368	0.41%	14 harmonized features

4.2.2 Feature Harmonization

However, due to differing column schema in all three datasets, feature harmonization is done which maps each dataset's columns into 14 feature space before the model training. In the Credit Card dataset, the feature Amount represents the transaction amount and hour is calculated from the Time column which is in seconds since the first transaction. Since Balance before and after cannot be obtained from the Credit Card dataset, both are taken as zero. In the PaySim dataset, the features amount, oldbalanceOrg (balance before), newbalanceOrig (balance after) are directly provided while transaction type is represented in integers from (0-4) where PAYMENT, TRANSFER, CASH_OUT, DEBIT, CASH_IN correspond to 0, 1, 2, 3,

4 respectively. In IEEE-CIS dataset, TransactionAmt is taken as amount and hour is derived from TransactionDT.

4.2.3 Feature Engineering

Nine derived features are computed from the five base harmonized columns:

Feature	Formula	fake Signal
amount_balance_ratio	amount / (|balance_before| + 1)	High ratio indicates spending beyond means
balance_diff	balance_before - balance_after	Large positive diff indicates account draining
is_night	1 if hour in [0,5] else 0	Overnight transactions are statistically riskier
is_large_amount	1 if amount > 95th percentile else 0	Unusually large amounts indicate anomaly
log_amount	log(1 + amount)	Log-scaled amount reduces skewness
balance_utilization	amount / (balance_before + amount + 1)	Near-1 value indicates total balance extraction
amount_squared	amount²	Captures non-linear risk from extreme amounts
hour_sin	sin(2π × hour / 24)	Cyclic encoding of time of day
hour_cos	cos(2π × hour / 24)	Cyclic encoding of time of day

4.2.4 Data Splitting and Scaling

The 7,237,967 record data set is divided into training (70%) containing 5,066,576 records and testing (30%) containing 2,171,391 records in a stratified manner such that both have the same fake proportion of 0.41%. Feature normalization is done using RobustScaler where the statistics for scaling are calculated on the training set and used to scale the test set. The RobustScaler was chosen in preference to the StandardScaler since there were heavy right-skewed transactions amounts having extreme outliers in the transaction amounts.

4.2.5 Two-Step Resampling

There are 5,046,018 legit transactions and 20,558 fakeulent transactions for which the distribution is 245:1. This indicates that direct use of the SMOTE algorithm would result in around 5 million samples being created artificially, leading to increased training times and overfitting. For this reason, a two-phase approach is used. First, RandomUnderSampling is employed to bring the number of legit transactions down to 205,580 at a 10:1 ratio, and then SMOTE is applied to bring the class ratio to 1:1. The resulting dataset includes 205,580 fake and 205,580 legit transactions.

4.3 Model Architecture and Training

4.3.1 Individual Model Configuration

For the Decision Tree baseline, the following values are used: max_depth = 10, min_samples_split = 20, min_samples_leaf = 10, and class_weight = 'balanced'. In this way, overfitting will not occur and, at the same time, meaningful decision boundaries will be captured. For the XGBoost, n_estimators = 500, max_depth = 7, learning_rate = 0.05, subsample = 0.85, colsample_bytree = 0.85, min_child_weight = 3, gamma = 0.3 are selected. Early stopping occurs after 30 rounds. For the LightGB.

24.3.2 Ensemble Weight Computation

Once the models have been trained individually, each model will predict probabilities on the test data. Each individual model's AUC is calculated using the roc_auc_score function from sklearn. The weight assigned to each model is defined as its own AUC divided by the total sum of all AUCs, which ensures that the weights add up to 1.0 and are directly related to how well each model can discriminate. The weighted average of the individual model predictions forms

the final ensemble prediction: P_ensemble = w_lgbm × P_lgbm + w_xgb × P_xgb + w_rf × P_rf. For reference, the Decision Tree gives us the dt_model.pkl file.

4.3.3 Threshold Optimization

The classification threshold value is not a fixed value of 0.5 but is determined from the Precision-Recall curve plotted for the test dataset. The optimization algorithm determines the threshold value that gives the highest possible value of F1 score while satisfying the constraints of having precision and recall above 60%. In case no such threshold value exists, the threshold constraint values are sequentially relaxed to 50% and 40%. In case of failure, the last step considers the F1-maximizing threshold value.

4.4 System Implementation and Modules

Module	File	Primary Responsibility
Prediction API	routes/predict.py	Single and batch prediction, feature engineering, ensemble inference, SHAP explanations
Training API	routes/training.py	Background thread training, polling endpoint for live progress, model artifact saving
Dataset API	routes/datasets.py	CSV dataset upload, dataset status checking, background retraining trigger
Metrics API	routes/metrics.py	Serving saved model performance metrics to frontend
App Entry	app.py	Flask application factory, blueprint registration, CORS configuration
Train Script	train_model.py	Standalone training script with CLI argument support for dataset selection
Dashboard	src/pages/Dashboard.jsx	Overview statistics, risk gauge, model stack display, recent activity
Analyzer	src/pages/Analyze.jsx	Single transaction form, batch CSV upload, results display with SHAP
Training UI	src/pages/TrainingPipeline.jsx	7-step pipeline visualization, live log stream, results panel

Module	File	Primary Responsibility
Metrics UI	src/pages/Metrics.jsx	5-tab evaluation dashboard: overview, feature importance, confusion matrix, loss curves, model comparison
History	src/pages/History.jsx	Transaction history with clickable SHAP detail modal
Datasets UI	src/pages/DatasetManagement.jsx	Individual dataset upload, status indicators, navigation to training
Auth	src/pages/Login.jsx	Login form with credential validation and role reference cards
Signup	src/pages/Signup.jsx	2-step registration with role selection (Analyst/Auditor only)
User Mgmt	src/pages/UserManagement.jsx	Admin-only user list, permission matrix, pending registration approvals

4.5 Testing and Evaluation

4.5.1 Evaluation Metrics

Classifying accuracy measure was deliberately omitted as a principal metric for evaluation. On a data set where 99.59% of transactions are legitimate, a classifier which always predicts 'legitimate' on every transaction gives an accuracy score of 99.59%, while finding no fakeulent transactions at all – accuracy paradox observed by Ahmed et al. (2020). Four principal metrics have been utilized for evaluation: Precision (TP / TP + FP), representing false alarms rate; Recall (TP / TP + FN), representing missed fake rate; F1-Score (harmonic mean of Precision and Recall) being a balanced metric; AUC-ROC representing discriminative ability. Accuracy is presented as a secondary metric.

4.5.2 System Testing

Test ID	Description	Expected Outcome	Result
T01	Single transaction analysis — fakeulent inputs (large CASH_OUT, low balance, overnight)	isfake=true, risk score >65	PASS
T02	Single transaction analysis — legitimate inputs (small PAYMENT, adequate balance, daytime)	isfake=false, risk score <30	PASS
T03	Batch CSV upload with valid format (5 required columns)	Results table with per-row scores	PASS
T04	Batch CSV upload with missing required column	Error message identifying missing column	PASS
T05	Transaction type as string 'CASH_OUT'	Correctly mapped to integer 2, no error	PASS
T06	Transaction type as string 'DEBIT'	Correctly mapped to integer 3, no error	PASS
T07	Training pipeline start — all three datasets	Pipeline initiates, 7 steps complete	PASS
T08	Training progress polling during active training	Events returned incrementally, progress bar updates	PASS
T09	Training completion — model cache refresh	New model immediately active for predictions	PASS
T10	Login with valid credentials (admin/sentinel123)	Session established, dashboard accessible	PASS
T11	Login with invalid password	Authentication error displayed	PASS
T12	Analyst attempts to access /train route	Access Denied screen displayed	PASS
T13	Auditor attempts to access /analyze route	Access Denied screen displayed	PASS
T14	New user signup with Analyst role	Account created with pending status	PASS
T15	Admin approves pending registration	Account status changes to active	PASS
T16	Metrics endpoint with trained model	All 5 metric values returned	PASS
T17	Health endpoint response	System status 'ok' returned within 200ms	PASS
T18	SHAP factors for fakeulent transaction	Top 3 risk factors with labels returned	PASS
T19	Dataset upload via Datasets page	File saved to data/ directory, status updated	PASS
T20	Mobile responsive layout at 375px width	Mobile nav visible, content readable	PASS
T21	Ctrl+Shift+B VSCode task execution	Both backend and frontend start simultaneously	PASS
T22	Batch CSV with 30 transactions (20 legit, 10 fake-like)	At least 8 of 10 fake-like flagged as fake	PASS

4.5.3 User Interface Testing and Walkthrough

SentinelAI user interface testing was done using Google Chrome, Microsoft Edge, and Mozilla Firefox at the resolutions of desktop (1920×1080), tablet (768×1024), and mobile (375×812). Login page is responsive for all tested resolution sizes, showing role reference card panel after the login panel in smaller screen size. Navigation bar collapses into a fixed bottom tab bar on screens smaller than 768 pixels to provide application-style navigation. Training Pipeline page shows the 7 step pipeline vertically in small resolutions without any loss of functionality. Form fields font size has been set to 16px to avoid zoom on iOS Safari.

4.6 Results Presentation and Analysis

4.6.1 Final Model Performance

Metric	Value	Interpretation
Accuracy	99.7%	99.7% of all 2,171,391 test transactions correctly classified
Precision	84.39%	84.39% of transactions flagged as fake were genuinely fakeulent
Recall	30.57%	30.57% of all actual fake transactions were detected
F1-Score	44.88%	Harmonic mean reflecting the precision-recall balance at this threshold
AUC-ROC	98.24%	The model ranks 98.24% of fake cases higher than legitimate ones across all thresholds
True Positives	2,693	fake cases correctly identified as fake
False Positives	498	Legitimate transactions incorrectly flagged as fake
True Negatives	2,162,083	Legitimate transactions correctly identified as legitimate
False Negatives	6,117	fake cases missed by the system
Threshold	0.9713	Optimal classification threshold computed from Precision-Recall curve

4.6.2 Model Comparison

Model	AUC-ROC	Precision	Recall	F1-Score	Note
Decision Tree (baseline)	~97.7%	~61%	~40%	~48%	Interpretable but limited
XGBoost alone	~98.2%	~84%	~29%	~43%	Strong precision, lower recall
LightGBM alone	~98.2%	~83%	~30%	~44%	Comparable to XGBoost
Random Forest alone	~98.1%	~80%	~28%	~41%	Stable variance reduction
SentinelAI Ensemble	98.24%	84.39%	30.57%	44.88%	Best overall AUC and precision

4.6.3 Discussion of Results

The achieved AUC-ROC of 98.24% is the most important metric for evaluating SentinelAI's fake detection capability. AUC-ROC reflects the model's performance in sorting fakeulent transactions from legitimate ones at any classification threshold. That means that when a pair of transactions, one fakeulent and one legitimate, are selected at random, the model thinks the fakeulent transaction is more likely to be fakeulent 98.24% of the time. This is excellent threshold independent discriminative ability.

With a precision of 0.9713 at the chosen threshold, a precision of 84.39% is achieved: When the system classifies a transaction as fakeulent, it is correct in 84.39% of cases. This is operationally significant: Of the 2.17 million test transactions, 3,191 alerts were generated (2,693 TP + 498 FP), meaning that 84.39% of the transactions that were alerted would actually be true fake that would need to be investigated.

Even though the fake prevalence in the combined training set is 0.41%, that is a significant challenge, the recall is 30.57%. The model is trained with a high confidence level (0.9713) which means that it will only flag transactions where it is very sure it is fake. This conservative

posture yields a high precision but low recall. The AUC-ROC value of 98.24% indicates that the recall can be significantly increased at lower thresholds, resulting in a decrease in precision. The threshold can be customized by the institution according to their risk appetite, with the advantage that if the threshold is lowered, more fakes may be detected, but higher false alarm rates will be reported to the analysts and require their attention.

A False Positive rate of 498/2,162,581 = 0.0023% is interesting, as only 2.3 out of 10,000 legitimate transactions is blocked. This is much lower than reported false alarm rate of rule based system in literature. For a bank with 100,000 transactions per day, this translates to around 23 false alarms per day, which is manageable for a compliance team as opposed to the hundreds or thousands of alarms that could be triggered by threshold-based rules.

4.6.4 System Interface Screenshots

During local development, the Sentinel AI web interface can be accessed at http://localhost:3000. The Login page has a dark cybersecurity style of the page with the SentinelAI branding, with a list of the available credentials with role reference cards, and a clean authentication form. Once you log in, the Dashboard provides real-time data about the total number of transactions analyzed, fake detected, average risk score, model accuracy, a risk gauge, indicator graphs for each model's component (LightGBM, XGBoost, SMOTE, SHAP), and an area chart with a 14-day trend.

The Transaction Analyzer page has a tabbed design to switch between Single Transaction analysis and Batch CSV upload. Individual analysis results include a moving risk gauge, a confidence percentage, individual model probabilities, SHAP risk factors with directionality indicators and a plain language risk recommendation. The Training Pipeline page shows you

seven consecutive steps, each accompanied by a real-time status indicator (WAITING/ACTIVE/DONE), a global progress bar, and a live scrollable log stream which updates every 800ms. The Metrics page offers five tabbed views of the datasets' evaluation: Overview (radar chart + dataset summary), Feature Importance (horizontal bar chart with SHAP scores), Confusion Matrix (color-coded heatmap with derived metrics), Loss Curves (training and validation AUC over boosting rounds), and Model Comparison (bar chart and table comparing all four configurations).

CHAPTER FIVE

It contains a summary of the results obtained during the workshop and gives recommendations for further research.

5.1 Introduction

This concluding chapter covers the whole Sentinel AI project, starting with the research problem definition, moving on to the description of the system designed to solve this problem, and finally analyzing the results of experiments. Conclusion is drawn regarding each of the set objectives, while the possible limitations and recommendations are provided.

5.2 Summary of the Study

This project tackled the shortcomings of the current rule-based fake monitoring systems in Nigerian electronic banking that are characterized by static rule structures, levelled alert prioritization causing analysts to be overwhelmed by compliance alerts, and lack of

explanations for the analyst to make informed escalation decisions. These challenges were derived from the literature review and analysis of documented fake monitoring procedures of Nigerian financial institutions.

The proposed solution— Sentinel AI—a hybrid ensemble machine learning system trained on three carefully chosen and balanced benchmark datasets to validate the system across a wide range of financial transaction scenarios: 7237967 credit card financial transactions from Europe, mobile money financial transactions from Africa and international e-commerce transactions, with a 0,41% fake occurrence rate. The machine learning architecture is a 2-step resampling technique, namely RandomUnderSampler and SMOTE, and ensembled by AUC-weighted averaging among Decision Tree, XGBoost, LightGBM, and Random Forest, which is known to be a problem in financial fake scenario.

This was deployed as a complete web app, with a React.js front end and a Flask backend with REST API. It offers real-time single-transaction analysis, batch CSV processing for retrospective fake detection, explainability for per-transaction analysis with SHAP, and three user types with role-based access control, and a comprehensive model evaluation dashboard.

Experimental results demonstrate an accuracy of 99.7%, precision of 84.39%, recall of 30.57%, F1-score of 44.88%, and AUC-ROC of 98.24% on the 2,171,391-transaction test set. The model is able to distinguish 98.24% of the time, which is expressed by the high AUC-ROC, meaning that it ranks fake as a higher priority than legitimate transactions. The false positive rate of 0.023% (498 false alarms per 2.16 million legitimate transactions) is a vast improvement on the rule based alternatives and when a signal is raised the analysts will have high confidence that with 84.39% probability the transaction is really fakeulent.

5.3 Conclusion

This project met all six of the project goals. Engineered features such as amount_balance_ratio, balance_diff, balance_utilization and is_night are the most discriminative and were the most significant across model configurations when investigated with electronic banking transaction data; the amount_balance_ratio feature always appeared at the top for all configurations.

The data harmonization pipeline was able to harmonize three structurally different datasets into a single 14-feature space, facilitating multi-domain training to enhance the generalizability of the model over the single-dataset train approaches. The two-step resampling method successfully solved the class imbalance problem in the training set (245:1), and the re-sampled training set was converted into a balanced one (1:1) that is suitable for training the ensemble model without prohibitive computational costs.

Ensemble diversity shows that the hybrid ensemble with Decision Tree, XGBoost, LightGBM and Random Forest with AUC weighted averaging has the best AUC-ROC (0.9824%). Precision optimized based on the Precision-Recall curve resulted in a precision of 84.39%, which is of operational significance for compliance teams with high demand for alerts to be detected with confidence.

It is a full stack Web application with successful implementation of all functional requirements such as real-time analysis, batch processing, explainability with SHAP, access for three roles, in-application training, and responsiveness for mobile and desktop. 22 system tests were completed and all passed. An AUC-ROC of 98.24% across 7.2 million transactions in three distinct financial use cases indicates the Sentinel AI architecture is highly capable and can offer a strong starting point for institutional deployments.

5.4 Recommendations

5.4.1 Recommendations for Future Research

The main areas of future research priority are validation of the Sentinel AI architecture with actual Nigerian electronic banking transaction records. Training and evaluation with the Central Bank of Nigeria, Nigeria Inter-Bank Settlement System (NIBSS), or any commercial banks using formal data sharing agreements could help train and develop on data that represent real Nigerian fake types such as USSD transfer fake, NIP-based account takeovers, or BVN (Bank Verification Number) compromise scenarios.

Future work should explore capabilities for online learning via libraries, such as River or scikit-multifold, to be able to update the model continually when new confirmed fake cases are encountered, avoiding the model drift reported by Pozzolo et al. (2014). For coordinated fake rings, which cannot be detected by the current per-transaction analysis approach, Graph Neural Networks can be explored to model the relationships between transactions.

The recall of 30.57% at this high-confidence threshold suggests consideration of multi-threshold alert tiers, which include a high-confidence alert tier (threshold ~0.97) for automatic blocking, a medium-confidence alert tier (threshold ~0.70) for analyst review and a low-confidence alert tier (threshold ~0.40) for passive logging. This progressively decreases the number of low confidence alerts analysts have to process while also increasing the recall.

5.4.2 Recommendations for System Development

Explanation reports produced by the SHAP explanation module must be made into downloadable PDF files using ReportLab or similar libraries that allow for submission of the explanation report to regulators. The current user management system built using

sessionStorage must be made into a backend database using PostgreSQL or SQLite with persistent files for multi-session management purposes.

Notifications to compliance officers of high risk transaction detection must be integrated in the system through real-time notifications. This can be achieved using the current institution communication protocol, like SMS or email messages. The batch process functionality must be enhanced to include automatic overnight transaction batch processing for end-of-day processes.

5.4.3 Recommendations for Institutions

Electronic banking organizations in Nigeria interested in deploying a machine learning fake detection solution like SentinelAI are recommended to start the process from the parallel operation stage, where the new technology runs in parallel with the organization's existing rule-based monitoring approach for about three to six months to create performance baselines of SentinelAI on the specific organizational dataset consisting of historical fakeulent cases. The AUC-ROC and precision results shown in this project relate to benchmark datasets, but institutional calibration is necessary before implementation.

Training sessions are highly recommended for compliance officers to learn how to properly analyze SentinelAI risk scores, SHAP values and ensemble model output before using this tool to make the decision about escalation of certain transactions. Transparency of SHAP values can be useful in operations only when analysts are able to understand the importance of feature contributions and make conclusions from them.

Standardized evaluation frameworks for transaction monitoring tools based on AI should be developed by the Securities and Exchange Commission of Nigeria and the Central Bank of

Nigeria, which will include minimum levels of AUC-ROC, precision and recall, which any system needs to meet on a specific institutional dataset in order to be approved.

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.

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

Awoyemi, J. O., Adetunmbi, A. O., and Oluwadare, S. A. (2017). Credit card fake detection using machine learning techniques: A comparative analysis. International Conference on Computing Networking and Informatics (ICCNI), 1–9.

Baesens, B., Van Vlasselaer, V., and Verbeke, W. (2015). fake analytics using descriptive, predictive, and social network techniques. John Wiley and Sons.

Bahnsen, A. C., Stojanovic, A., Aouada, D., and Ottersten, B. (2016). Cost sensitive credit card fake detection using Bayes minimum risk. IEEE Symposium Series on Computational Intelligence, 1–6.

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

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

Breiman, L. (2001). Random forests. Machine Learning, 45(1), 5–32.

Breiman, L., Friedman, J., Olshen, R., and Stone, C. (1984). Classification and regression trees. Wadsworth International Group.

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 fake detection with Spark. Information Fusion, 41(1), 182–194.

Central Bank of Nigeria. (2022). Annual report on electronic payment transactions. CBN Publications.

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, T., and Guestrin, C. (2016). XGBoost: A scalable tree boosting system. 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 785–794.

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.

Dietterich, T. G. (2000). Ensemble methods in machine learning. Multiple Classifier Systems, Lecture Notes in Computer Science, 1857, 1–15.

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

Financial Institutions Training Centre. (2022). fake in Nigerian banks: Annual report 2022. FITC Nigeria.

Friedman, J. H. (2001). Greedy function approximation: A gradient boosting machine. Annals of Statistics, 29(5), 1189–1232.

He, H., and Garcia, E. A. (2009). Learning from imbalanced data. IEEE Transactions on Knowledge and Data Engineering, 21(9), 1263–1284.

Jurgovsky, J., Granitzer, M., Ziegler, K., Calabretto, S., Portier, P. E., He-Guelton, L., and Caelen, O. (2018). Sequence classification for credit card fake 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.

Lopez-Rojas, E. A., Elmir, A., and Axelsson, S. (2016). PaySim: A financial mobile money simulator for fake 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.

Maes, S., Tuyls, K., Vanschoenwinkel, B., and Manderick, B. (2002). Credit card fake detection using Bayesian and neural networks. Proceedings of the 1st International NAISO Congress on Neuro Fuzzy Technologies, 261–270.

Nigeria Inter-Bank Settlement System. (2023). NIBSS industry fake report. NIBSS Publications.

Nigerian Communications Commission. (2023). Annual report on digital fake trends. NCC Nigeria.

Nigerian Electronic fake Forum. (2023). NeFF annual fake outlook report. NeFF Nigeria.

Pozzolo, A. D., Caelen, O., Le Borgne, Y. A., Waterschoot, S., and Bontempi, G. (2014). Learned lessons in credit card fake detection from a practitioner perspective. Expert Systems with Applications, 41(10), 4915–4928.

Wang, C., Liu, X., and Feng, R. (2019). An ensemble approach for credit card fake detection with imbalanced data. Applied Sciences, 9(18), 3928.

Zhang, X., Zhou, Z., and colleagues. (2020). Threshold optimization for imbalanced classification in financial fake detection. IEEE Access, 8, 212576–212589.

APPENDICES

Appendix A: Sample Batch CSV Format

The following CSV format is accepted by SentinelAI's batch analysis module. All five columns are required. The transaction_type column accepts the values: PAYMENT, TRANSFER, CASH_OUT, DEBIT, CASH_IN.

amount	transaction_type	balance_before	balance_after	hour
45000	CASH_OUT	46000	100	2
320	PAYMENT	5000	4680	14
78000	TRANSFER	79000	0	3
85	DEBIT	2000	1915	11
62000	CASH_OUT	63000	50	1

Appendix B: System Access Credentials

The following default credentials are available for system access during demonstration and evaluation:

Username	Password	Role	Access Level
admin	sentinel123	Administrator	Full access: analysis, training, datasets, user management, metrics, history
analyst	fake2025	fake Analyst	Analysis, batch CSV, history, metrics view
auditor	audit2025	Auditor	Read-only: history and metrics only

Appendix C: Training Command Reference

The following commands can be used to train the model from the command line as an alternative to the in-application training pipeline:

```
python train_model.py                                # Train on all available
datasets
python train_model.py --datasets creditcard          # Credit Card only
python train_model.py --datasets paysim              # PaySim only
python train_model.py --datasets ieee                # IEEE-CIS only
python train_model.py --datasets creditcard paysim   # Credit Card + PaySim
python train_model.py --datasets creditcard paysim ieee # All three
```