



**FEDERATED LEARNING FOR CREDIT CARD FRAUD DETECTION: A
PRIVACY-PRESERVING APPROACH WITH CONTROLLED NOISE
INTEGRATION**

RUSUL MAHDI ABDULHADI AL MAQADAS

JANUARY 2025

ÇANKAYA UNIVERSITY

GRADUATE SCHOOL

COMPUTER ENGINEERING DEPARTMENT

INFORMATION TECHNOLOGY MASTER'S THESIS

**FEDERATED LEARNING FOR CREDIT CARD FRAUD DETECTION: A
PRIVACY-PRESERVING APPROACH WITH CONTROLLED NOISE
INTEGRATION**

RUSUL MAHDI ABDULHADI AL MAQADAS

JANUARY 2025

ABSTRACT

FEDERATED LEARNING FOR CREDIT CARD FRAUD DETECTION: A PRIVACY-PRESERVING APPROACH WITH CONTROLLED NOISE INTEGRATION

AL MAQADAS, RUSUL MAHDI ABDULHADI
INFORMATION TECHNOLOGY MASTER'S THESIS

Supervisor: Assist. Prof. Dr. Murat SARAN

January 2025, 89 Pages

With the rapid increase in e-payment technology, cards are present as one of the essential tools. However, with this growth comes a risk of fraudulent attacks that may cause involved parties' losses and damage. Banks aim to establish strong fraud detection systems to protect assets and obey regulator rules. Therefore, developing a model that copes with security and integrity requirements is crucial. This research project introduced advanced Machine Learning (ML) models, such as Deep Neural Network (DNN) and ensemble learning with AdaBoost, to detect fraud while addressing the skewed nature of datasets using balancing methods. Additionally, it facilitates collaborative learning among banks using Federated Learning (FL) while preserving data privacy. In this study, the FL model was tested against various percentages of label-flip attacks to evaluate resilience against malicious acts by banks trying to sabotage the learning process. The models were tested on two datasets: a real dataset from an Iraqi bank and the known Kaggle *creditcard* dataset. Models were assessed on a set of performance metrics to cover all aspects of the methods. Results showed that ensemble learning with Random Forest (RF) and AdaBoost achieved remarkable performance across both datasets. Moreover, the FL cosine-based worked better than the existing Federated Average method. Lastly, the proposed approach combining RF+AdaBoost with FL cosine aggregation surpassed the existing method when validated on a private bank dataset achieving 96.47% accuracy and 94.58%

recall. The present study contributed to the academic literature by addressing the lack of real datasets in the fraud detection domain.

Keywords: Credit Card Fraud, Ensemble Learning, Deep Neural Network, Federated Learning, Malicious Attacks



ÖZET

KREDİ KARTI DOLANDIRICILIĞI TESPİTİ İÇİN FEDERATED LEARNING: KONTROLLÜ GÜRÜLTÜ ENTEGRASYONU İLE GİZLİLİK KORUYUCU YAKLAŞIM

**AL MAQADAS, RUSUL MAHDI ABDULHADI
BİLGİ TEKNOLOJİLERİ YÜKSEK LİSANS TEZİ**

Danışman: Dr. Öğr. Üyesi Murat SARAN

Ocak 2025, 89 Sayfa

E-ödeme teknolojisinin hızla artmasıyla birlikte, kartlar önemli araçlardan biri olarak öne çıkmaktadır. Ancak, bu büyüme dolandırıcılık saldırılarının riskini de beraberinde getirerek ilgili tarafların zarar görmesine ve kayıplar yaşamasına neden olabilir. Bankalar, varlıklarını korumak ve düzenleyici kurallara uymak için güçlü dolandırıcılık tespit sistemleri kurmayı hedeflemektedir. Bu nedenle, güvenlik ve bütünlük gereksinimlerini karşılayan bir model geliştirmek kritik öneme sahiptir. Bu araştırma projesinde, Derin Sinir Ağı (DNN) ve AdaBoost ile topluluk öğrenimi gibi gelişmiş Makine Öğrenimi (ML) modelleri tanıtılarak, veri setlerindeki dengesiz yapıyı dengeleme yöntemleriyle dolandırıcılık tespit edilmiştir. Ayrıca, veri gizliliğini koruyarak bankalar arasında Federated Learning (FL) aracılığıyla iş birliği içinde öğrenme kolaylaştırılmıştır. Bu çalışmada, FL modeli, öğrenme sürecini sabote etmeye çalışan bankaların kötü niyetli eylemlerine karşı dayanıklılığı değerlendirmek için farklı oranlarda etiket tersine çevirme saldırılarına karşı test edilmiştir. Modeller, bir Irak bankasından alınan gerçek bir veri seti ve tanınmış Kaggle kredi kartı veri seti üzerinde test edilmiştir. Yöntemlerin tüm yönlerini kapsayacak bir dizi performans metriği ile değerlendirilmiştir. Sonuçlar, Random Forest (RF) ve AdaBoost ile topluluk öğreniminin her iki veri setinde de dikkate değer bir performans sergilediğini göstermiştir. Ayrıca, FL cosine tabanlı yöntem, mevcut Federated Average

yönteminden daha iyi sonuçlar vermiştir. Son olarak, RF+AdaBoost ile FL cosine birleşimini içeren önerilen yaklaşım, özel bir banka veri setinde doğrulama yapıldığında %96.47 doğruluk ve %94.58 geri çağırma oranı ile mevcut yöntemi geride bırakmıştır. Bu çalışma, dolandırıcılık tespiti alanında gerçek veri setlerinin eksikliğini ele alarak akademik literatüre katkıda bulunmuştur.

Anahtar Kelimeler: Kredi Kartı Dolandırıcılığı, Topluluk Öğrenimi, Derin Sinir Ağı, Federated Learning, Kötü Niyetli Saldırıları



ACKNOWLEDGEMENT

I want to express my deep appreciation and deepest gratitude to my advisors, Dr. Murat Saran and Dr. Nurdan Saran, for their unwavering support and invaluable guidance throughout my thesis journey. Their expertise and insights helped me shape my idea and add depth and clarity in various aspects. I am profoundly grateful to my family, colleagues, and everyone who supported me throughout my journey as a foreign student in Ankara in Türkiye.

TABLE OF CONTENTS

STATEMENT OF NONPLAGIARISM	iii
ABSTRACT	iv
ÖZET	vi
ACKNOWLEDGEMENT	viii
TABLE OF CONTENTS	ix
LIST OF TABLES	xiii
LIST OF FIGURES	xiv
LIST OF SYMBOLS AND ABBREVIATIONS	xv
CHAPTER I	1
INTRODUCTION	1
CHAPTER II	3
BACKGROUND	3
CHAPTER III	13
RELATED WORK	13
3.1 OVERVIEW OF THE LITERATURE REVIEW	13
3.2 DEEP LEARNING APPROACH TO CREDIT CARD FRAUD DETECTION	16
3.3 ENSEMBLE LEARNING APPROACH TO CREDIT CARD FRAUD DETECTION.....	16
3.4 FEDERATED LEARNING WITH GRAPH NEURAL NETWORKS APPROACH TO CREDIT CARD FRAUD DETECTION	18
3.5 FEDERATED AVERAGING WITH DISTANCE-BASED WEIGHTED AGGREGATION APPROACH TO CREDIT CARD FRAUD DETECTION.....	18
CHAPTER IV	21
METHODOLOGY	21
4.1 DEALING WITH THE IMBALANCED NATURE OF FRAUD DATA.....	22
4.1.1 Synthetic Minority Over-sampling Technique (SMOTE)	23
4.1.2 Hybrid Balancing (SMOTE+RUS)	24

4.2 DATASETS USED IN THIS STUDY	25
4.2.1 Original Iraqi Bank Dataset	25
4.2.2 Kaggle Dataset.....	28
4.3 MODEL ARCHITECTURE.....	28
4.3.1 Artificial Neural Network (ANN) Model	28
4.3.1.1 Overview of the Model	28
4.3.1.2 Data Preprocessing and Balancing.....	30
4.3.1.3 Hyperparameter Tunning	30
4.3.1.4 Architecture Design	30
4.3.1.5 Model Training.....	31
4.3.2 Long Short-term Memory (LSTM)-based Deep Learning Model.....	32
4.3.2.1 Overview of the Model	32
4.3.2.2 Data Preprocessing and Balancing.....	32
4.3.2.3 Hyperparameter Tunning	33
4.3.2.4 Architecture Design	33
4.3.2.5 Model Training.....	34
4.3.3 RF+AdaBoost Model.....	34
4.3.3.1 Overview of the Model	34
4.3.3.2 Data Preprocessing and Balancing.....	35
4.3.3.3 Architecture Design	36
4.3.3.4 Model Training.....	36
4.4 APPLICATION OF FEDERATED LEARNING	37
4.4.1 An Integration of Federated Learning and Graph Neural Networks for Privacy-preserving Financial Security.....	37
4.4.1.1 Overview of the model.....	37
4.4.1.2 Data Preprocessing.....	38
4.4.1.3 Architecture Design	39
4.4.1.4 Model Training.....	40
4.4.1.4.1 Local Training	40
4.4.1.4.2 Global Training	40
4.4.2 Federated Averaging with Distance-based Weighted Aggregation Approach.....	41
4.4.2.1 Overview of the Model	41
4.4.2.2 Data Preprocessing.....	41

4.4.2.3 Architecture Design	42
4.4.2.4 Model Training.....	43
4.4.2.4.1 Local Training	43
4.4.2.4.2 Global Training	43
4.5 EVALUATION METRICS.....	43
4.5.1 Confusion Matrix.....	43
4.5.2 Classification Metrics	44
4.5.3 Visual Representations.....	45
4.5.4 Statistical Metrics	45
4.6 POST-PROCESSING	46
4.6.1 Adding and Detecting Noise in FL Systems.....	46
4.6.1.1 Overview of the Model	46
4.6.1.2 Architecture Design	46
CHAPTER V	48
EXPERIMENTS	48
5.1 HARDWARE	49
5.2 EXPERIMENTAL ANALYSIS OF FRAUD DETECTION IN IRAQI BANKING TRANSACTION	49
5.2.1 ANN Model	49
5.2.2 LSTM Model	50
5.2.3 RF+AdaBoost Model.....	50
5.2.4 Summary of Model Performance.....	51
5.2.5 Graph-based Neural Network with FL Cosine Model.....	52
5.2.6 SMOTE+RF+AdaBoost with FL Cosine Model	52
5.2.7 SMOTE+RF+AdaBoost with FL Model	53
5.2.8 Execution Time Analysis	53
5.2.9 Using ANN Model with FL Cosine Similarity	54
5.2.10 Adding and Detecting Noise in FL with Cosine System	55
5.2.11 Adding and Detecting Noise in FL with FedAVG-DWA System	55
5.3 EXPERIMENTAL ANALYSIS OF FRAUD DETECTION IN KAGGLE DATASET.....	56
5.3.1 Using ANN Model.....	56
5.3.2 Using the LSTM Model.....	56
5.3.3 Using SMOTE+ RF+AdaBoost Model	57

5.3.4 Using Graph-based Neural Network with FL Cosine Model	57
5.3.5 Using SMOTE+ RF+AdaBoost+FL Cosine Model	58
5.3.6 Using ANN with FL Cosine Model	58
5.3.7 Using FL Cosine and Noise Detection	59
CHAPTER VI.....	60
DISCUSSION AND CONCLUSION.....	60
CHAPTER VII	65
FUTURE WORK	65
REFERENCES.....	66



LIST OF TABLES

Table 1: Design of ANN Model.....	31
Table 2: Design of LSTM Model.....	34
Table 3: ANN Model Metrics	49
Table 4: LSTM Model Metrics	50
Table 5: Comparison Results between RF and RF+AdaBoost	50
Table 6: Results of Graph-based FL Cosine-based Model	52
Table 7: Comparison results between RF+AdaBoost and RF+AdaBoost+FL	53
Table 8: Confusion Matrix of RF+AdaBoost with FL Cosine-based	53
Table 9: Comparison Results between RF+AdaBoost and RF+AdaBoost+FL DWA	53
Table 10: Confusion Matrix of RF+AdaBoost with FL DWA	53
Table 11: Comparison Results between ANN and ANN+ FL Cosine-based	54
Table 12: Evaluation Metrics at Different Noise Levels.....	55
Table 13: Results of ANN Model	56
Table 14: Results of ANN Model using Exact Model Design.....	56
Table 15: Results of the LSTM Model	57
Table 16: Results of RF+AdaBoost Model	57
Table 17: Results of Graph-based FL Cosine-based Model	58
Table 18: Results of RF+AdaBoost+FL Cosine-based Model	58
Table 19: Results of ANN with FL Cosine-based Model	59
Table 20: Evaluation Metrics at Different Noise Levels.....	59
Table 21: Performance Comparison with Existing Literatures	64

LIST OF FIGURES

Figure 1: Basic ANN Design	7
Figure 2: Basic LSTM Design	9
Figure 3: Basic FL Design	11
Figure 4: DNN Design from Paper	16
Figure 5: Methodology Process Flowchart	22
Figure 6: Distribution of Normal and Fraudulent Transactions.....	26
Figure 7: Collect USD by doing Transactions with no Product	27
Figure 8: RF+AdaBoost Design.....	35
Figure 9: FL Design	37
Figure 10: Confusion Matrix of ANN Model	49
Figure 11: Confusion Matrix of LSTM Model	50
Figure 12: Confusion Matrix of SMOTE+RF+AdaBoost Model.....	51
Figure 13: Comparison of MCC between Various ML Models	51
Figure 14: Time Analysis of Both Aggregation Methods	54
Figure 15: Models Comparison between PB_dataset and Kaggle Against <i>Accuracy</i>	60
Figure 16: Models Comparison between PB_dataset and Kaggle Against <i>Recall</i> ...	61

LIST OF SYMBOLS AND ABBREVIATIONS

SYMBOLS

TF TensorFlow

ABBREVIATIONS

AdaBoost	: Advanced Boosting
ANN	: Artificial Neural Network
CCFDS	: Credit Card Fraud Detection Systems
CNN	: Convolutional Neural Network
DNN	: Deep Neural Network
DT	: Decision Tree
ECS	: Epsilon Cluster Selection
FDS	: Fraud Detection Systems
FedAvg	: Federated Average
GNN	: Graph Neural Network
GRU	: Gated Recurrent Unit
IID	: Independent and Identically Distributed
KNN	: K-Nearest Neighbors
LR	: Logistic Regression
LSTM	: Long Short-Term Memory
MDI	: Mean Decrease Impurity
ML	: Machine Learning
MLP	: Multilayer Perceptron
NN	: Neural Network
PB_dataset	: Private Bank dataset
RF	: Random Forest
RL	: Reinforcement Learning
RNN	: Recurrent Neural Network
RUS	: Random Under-Sampling

SGD	: Stochastic Gradient Descent
SMC	: Secure Multi-Party Computation
SMOTE	: Synthetic Minority Over-sampling Technique
SVM	: Support Vector Machine
UL	: Unsupervised Learning
XGBoost	: Extreme Gradient Boosting



CHAPTER I

INTRODUCTION

The increase in cashless transactions and the e-commerce era we live in today make the world a small city due to economic globalization, and countries lean more and more toward electrical payments. Banks and payment services companies issue many types of cards to fulfill the different needs of various customers, but with these advancements comes a dark side, which is the fraudulent attacks that have a catastrophic effect on the financial level by causing millions of dollars in losses and reputational damage of these businesses [1]. Card fraud is any illegal or deceivable act to gain profit using issued debit/credit/prepaid cards. Thus, banks try to build robust Fraud Detection Systems (FDS) to catch and detect fraud as soon as possible. Machine learning (ML) is one of the popular methods to detect fraud, especially after the enormous development of e-payment, which involves many transactions done daily [2],[3]. In recent years, many deep neural networks, ensemble learning, and optimization methods have been used to tackle this classification issue and handle the challenging nature of imbalanced datasets, as fraudulent activities are rare compared to legitimate ones. Moreover, different feature spaces and complex pattern behavior need to be adequately analyzed as it is not applicable to rely only on humans to investigate large bulks of data. Another challenge is the data sensitivity. Privacy and confidentiality concerns [4] restrict learning different patterns from different banks. According to this, we proposed advanced ML methods to detect fraud, focusing on balancing capturing fraudulent activities without affecting legitimate transactions done by actual cardholders while dealing with the challenges presented by this domain.

Our project uses Deep Neural Networks (DNN) and ensemble learning to detect fraud, followed by selecting the best model to further apply Federated Learning (FL) in a collaborative environment. We compare different aggregation methods and select the most effective one to achieve knowledge sharing between banks without breaking data sensitivity and learning various fraud patterns, which is crucial in the fraud detection domain. We further enhance the FL model by adding an FL Detector

to detect malicious bank or abuse participants in this collaborated field. The models tested on the Private Bank dataset (PB_dataset) of cards issued by an Iraqi bank, which is vital due to the scarcity of research using real datasets. In addition to that, various experiments have been carried out on suggested models to see the effect of FL on different ML methods once with our obtained *PB_dataset* as a primary dataset and second, with the public Kaggle *creditcard* dataset as a supplementary dataset to create a benchmark for comparison with multiple dataset sizes and characteristics. The models have been tested on the *PB_dataset* of transactions related to cards issued by an Iraqi bank for the second half of 2023, along with the Kaggle public dataset under the name *creditcard*.

This study aims to build an effective ML model to detect fraud along with the application of enhanced FL for a well-structured environment between learning banks, preserving data privacy. Also, handling issues related to this domain are the imbalanced nature of the dataset and the overfitting problem.

While the main scope of our project is credit card fraud detection using real datasets, we also mentioned in possible future work on the importance of preventing fraud, highlighting the role of banks in raising awareness about the correct use of cards and informing the cardholders about best practices to protect their credentials not to fall victim of fraudulent activities.

The rest of this thesis is divided into Chapter II background that walks the reader through different types of fraud and the classification of ML models. Also, the importance of FL and its various types are presented. Then, chapter III introduces a literature review in the domain of fraud detection in recent years. Moreover, multiple models were selected to inspire the methodology part of this thesis. After that, chapter IV explains the practical part of the thesis, followed by the experimental part that presents the results of each design under Chapter V. Following Chapter VI is a discussion and conclusion of the results. Lastly, further work suggestions are provided under Chapter VII.

CHAPTER II

BACKGROUND

This chapter provides an overview of the core concepts necessary to understand fraud detection and ML techniques and the methods used to improve their effectiveness. We begin by discussing types of fraud and the process of fraud detection. Also, explain the different ML types according to the dataset and the problem that needs to be solved using ML. Next, we introduce specific algorithms used in fraud detection, such as Random Forests, AdaBoost, Artificial Neural Network (ANN), and Long Short-term Memory Networks (LSTMs). We then explore the key challenges faced in fraud detection, such as class imbalance, adversarial attacks, and privacy concerns. Finally, we discuss federated learning and its classification as a privacy-preserving approach for addressing these challenges and provide context for its role in this thesis.

Fraud is the act of misleading, deceiving, or doing illegitimate events to gain, whether deceiving people, companies, or institutions [5]. There are different types of fraud, like online fraud, identity theft, and retail fraud, such as replicating cards, shoplifting, skimmed data, or Point of Sale (POS) deceptive and many other types [5][6]. Fraud detection is a set of actions to identify and prevent wrongful behavior. Banks or financial institutions try to protect their assets, reputation, and integrity by updating their ways to detect and safeguard their resources [5]. Credit card fraud detection is used to recognize and avoid fraudulent transactions to stop any damage that affects financial institutions. The process implies using different algorithms and ML methods to inspect data and infer patterns that help identify fraudulent behavior [7]. The procedure involves data collection, which is critical in fraud detection, especially with a large amount of data due to the revelational technology and e-payment that made it significant to collect data from different sources like user profiles and transaction history.

After that, data needs to be checked to ensure its quality. Preprocessing is the procedure of cleaning, standardizing, and encoding data to ensure data are transformed for better analysis. Each dataset has many features according to their nature; feature engineering is the process of identifying features, such as amount, time, and geographical place, to distinguish between legitimate and abnormal behavior. The data are prepared and ready to be trained using ML methods that rely on data history to derive patterns that help detect fraud. Different ML methods exist, like traditional ML, neural networks, and ensemble techniques. Depending on the result of model training, a decision is made, like flagging for further assessment or blocking. Since fraudsters are constantly updating their ways and methods, taking advantage of any weakness, fraud detection models need to be enhanced and checked for feedback to ensure that the systems are up to date [7].

To understand ML and its application, defining ML is valuable. ML combines the power of computer science with mathematical analysis to produce a predictive model. ML, a subset of artificial intelligence (AI), uses historical information or data to make decisions and predictions without writing a specific program to do it. ML performs complex computational tasks that humans cannot do, especially with the enormous amount of data; ML can discover patterns that help businesses make better decisions. The basic approaches of ML, depending on the data that want to be predicted, are supervised learning, unsupervised learning, semi-supervised learning, and reinforcement learning [5], Ensemble Learning, and Neural network [8].

With supervised learning, which is considered one of the commonly used ML, the data are labeled, which means that the supervised ML algorithm takes input data with known output for each related input [8]. To simplify the idea even more, supervised learning is like student learning under the adviser's supervision, as the supervised algorithms teach the machine to predict the output correctly. There exist many applications of supervised learning like fraud detection, risk assessment, image classification and many more applications. Supervised learning is classified into regression and classification [9]. In regression issues, there is a correlation between input and output variables, so it could be a linear or non-linear relationship between input and output, such as weather guessing. In contrast, the output is categorical with classification issues, either two or multi-classes like true/false or animal image classification [9].

The other type of ML is Unsupervised Learning (UL), which deals with unlabeled data that infers behaviors or patterns without depending on the associated output. The system learns from within, unleashing knowledge by analyzing data. UL approaches are essential to building AI systems as they enable independent decisions based on large amounts of unlabeled data, which is crucial to solving more complex tasks than supervised learning. It is worth mentioning that despite the ability to solve complicated issues, it is hard to measure its success compared to supervised learning [10].

The main types of UL are clustering, association, anomaly detection, and autoencoder. We briefly explain each type, starting with clustering, which divides data into several parts so that each part may belong to one cluster [10]. As for association, it focuses on the relationship between features in the dataset [8]. The other type of UL is anomaly detection, which locates outliers in the dataset. It is widely used to detect abnormal behavior that shifts from the norm. Many anomaly detection applications exist, such as fraud and intrusion detection [10]. The last type of UL is Autoencoder, which uses a neural network to transform the data (use encoder and decoder) to represent the data [10].

After explaining supervised and unsupervised ML, we are going to discuss semi-supervised ML, to fill up the gap between supervised and unsupervised learning, an emerging innovation of semi-supervised learning is presented to bridge between not having enough label datasets due to time-consuming and expensive procedures and enormous unlabeled dataset as it leverages the use of element from supervised learning to enhance overall performance of the whole dataset. The most well-known algorithm is self-training, where the model uses a labeled dataset to train and then uses the knowledge obtained to predict an unlabeled dataset. The other algorithm is Semi-Supervised Support Vector Machines (S3VM), which is derived from SVM and uses label and unlabeled datasets to generate decision boundaries [11].

Another approach of ML is Reinforcement Learning (RL), which is based on trial and error to give feedback in the form of rewards or punishments to adjust the behavior to obtain more rewards over time. This approach does not require prior knowledge of the domain issue, which makes it very appropriate for problems where the best solution is not identified or hard to obtain. Many fields explore RL, like gaming, robotics, and finance. As RL emerges, it needs to be improved, and challenges must be addressed [12].

As the field of ML grows and affects many fields, ensemble learning is presented as an advanced approach that combines many models to enhance the performance of the final model. Ensemble learning is constructing many classifiers and integrating their outcome to decrease the overall difference. This method achieved higher performance when compared to the standalone classifier. Ensemble learning can be used homogeneously by applying some type of learning or heterogeneously with different kinds of learning methods. ML models always try to get the best accuracy for prediction, so with the power of ensemble learning, the final model is utilized by combining many models and averaging them together, which reduces bias and overfitting. To imagine the ensemble learning method, consider only one person who decides what is best for a company compared to doing a meeting and asking other employees about their opinion, then doing a majority voting to select the best decision; that is precisely what ensemble learning is doing. Different algorithms are used in ensemble learning; the most widely used in research are Bagging, Boosting, and Stacking.

We will briefly explain each technique, starting with Bagging, also called bootstrap aggregation. Different models are trained in different sets of data, which means each model will train multiple sets of data, and the results will be combined by aggregating majority voting for better final prediction. With the bagging technique, the result is less variant and more robust. A famous example of ensemble learning using the bagging technique is RF, which combines many decision trees to enhance accuracy compared with one decision tree. The other technique used is Boosting, which merges the results of multiple models and alters the weak learner into a strong learner through many rounds. With boosting, weighted aggregation is applied to base learners. Famous boosting methods like AdaBoost adaptive boosting and Gradient Boosting are used in different ways to sum up base models [13].

Lastly, we discuss Stacking, the most used approach where ensemble learning uses different base models and their predictions as input to a new model called a blender or meta-model. It trains the blender using these predictions to generate enhanced final output [13].

Neural Network Learning is the final topic related to ML types, which has been notable and widely used in recent years. In a simple definition, a neural network is a set of algorithms that mimics human brain-connected neurons [14]. The human brain takes different inputs or features and analyses them using many neurons, and the output

is shared between neurons by doing this over and over until the best optimal output is reached [15]. Neural networks can perform complex computational tasks that take high dimensional inputs and produce the best output [14]. Different types of artificial neural networks are based on the problem parameters and mathematical activities. We will present the basic kinds of Neural Networks (NN), starting with the simplest form of ANN, which is feed-forward NN that takes the input, passes it through one or many layers, and exits output in one direction. Hence, it only has forward propagation, which passes information in one way. ANN consists of the input layer, one or many hidden layers, and an output layer, as shown in **Figure 1**.

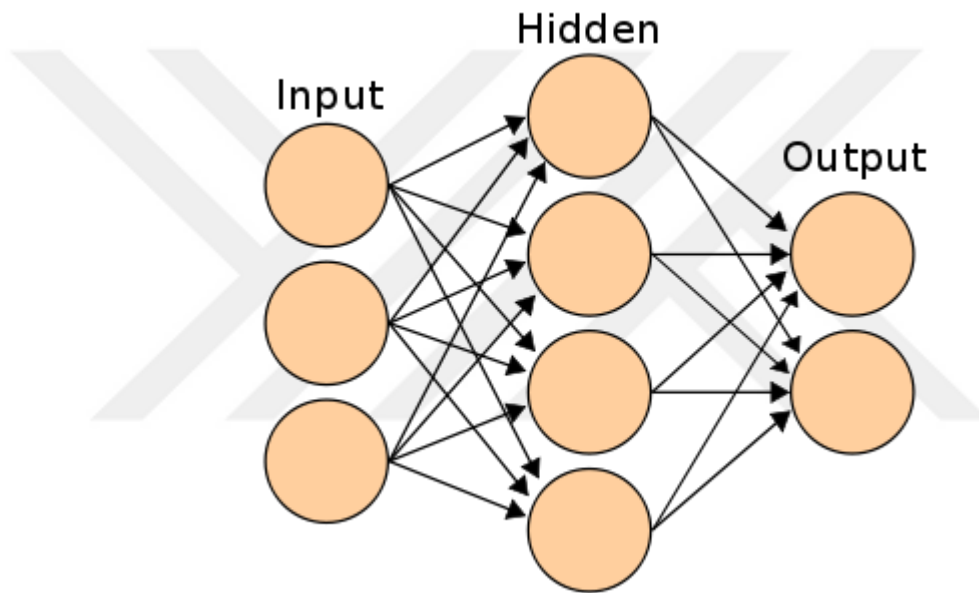


Figure 1: Basic ANN Design [15]

The other type of NN is a Convolutional Neural Network (CNN), which is commonly used in image and face recognition due to the power of a convolutional layer that can successfully recognize shapes by itself without prior preprocessing. The main element of CNN is the Convolutional Layer, an essential part of CNN that applies filters to the input to create activation; the other part of CNN is the rectified linear unit layer (ReLU). This activation function enhances feature extraction by introducing non-linearity. ReLU returns input if the input is positive; otherwise, return zero. ReLU solves the vanishing gradient issue, which helps the model to learn faster. The next component is the Pooling Layer, which reduces the feature dimension even more to focus on essential patterns. Key variants of pooling in CNN are average and max pooling. The last feature of CNN is that it has fully connected layers that use results

from the conventional layer and a pooling layer in the form of a hidden layer to predict the best outcome [16].

The final type of NN we will explore that many researchers talk about is Recurrent Neural Network (RNN); as the name implies, recurrent neural networks use backpropagation by saving the output of a layer and then feeding it back to the input layer so that the following learning have information (like memory) of the past round [15]. RNN is one of the most famous ML approaches used nowadays, though it has the power to handle time series modeling. Time series prediction is almost inherited in many domains like healthcare, economy, finance, energy, and telecommunication. The most popular RNN cells are LSTM and Gated Recurrent Unit (GRU) cells that use many hidden layers with gate mechanisms to memorize outcomes; LSTM uses three gates, while GRU uses two gates, with each gate presented as a hidden layer [17].

Many algorithms exist related to each type of ML mentioned above, and we highlight specific ones that we further used in our methodology part to understand each model fully. Firstly, we can explain the meaning of RF, a bagging-based ensemble method that uses random feature selection to lessen the relationship between base learners: “base learners are decision trees” [18]. RF is a vital ML algorithm that performs well in both classification and regression issues and is used in various domains. RF combines many decision trees to precisely evaluate the importance of each feature in the problem and select the most significant features that highly impact the forecasting task. RF was first presented by Leo Breiman in 2001, which merges the efficiency of decision trees and the power of ensemble learning to obtain remarkable achievements in ML. The main idea of RF is to predict based on building many decision trees, then take the predication from each tree and aggregate or take majority voting from all these predications to make the ending forecast. Each tree gets a random portion of the training data so that not all features assigned to one tree instead of the features divided between trees to lessen the effect of correlation that make RF immune to overfitting, this makes it an excellent choice for multi-feature datasets. Also, RF can handle absent values and anomalies. The other important aspect of RF feature importance assessment is to evaluate each feature contribution, which in turn affects the overall performance of the ML model. Feature importance is calculated for each feature in the dataset to assess the importance of each one in lessening the impurity of nodes in the decision tree. A standard equation used to calculate feature importance is called Mean Decrease Impurity (MDI), explained in Eq. 2.1, where n is

the total number of trees, original Error is the impurity before the features were used to split the trees, and displacement Error is the impurity after splitting the trees using feature j [19].

$$MDI_j = \text{original error} - \frac{1}{n} \sum_{i=1}^n (\text{displacement error}_i) \quad (2.1)$$

The other ML algorithm we want to review thoroughly is LSTM, an implication of RNN related to deep neural network learning with memory. It is appropriate for forecasting due to the connection between output and historical input. Thus, the LSTM uses the time sequence dependencies to learn. To achieve that, the LSTM model contains a memory cell with many conditions and gates; the cell state is the primary channel for knowledge movement as information goes forward without change. The forget gate chooses what data to save or remove. Data from the previous stage is directed through the sigmoid function and the current input state. The sigmoid function acts like a switch or activation function if a value near zero means neglect otherwise, if a value near one means store. Also, the cell state vector determines which component to erase. LSTM's basic design is shown in **Figure 2**. Due to the property of LSTM that handles sequence dependency, it is a considerable choice for financial fraud detection [20].

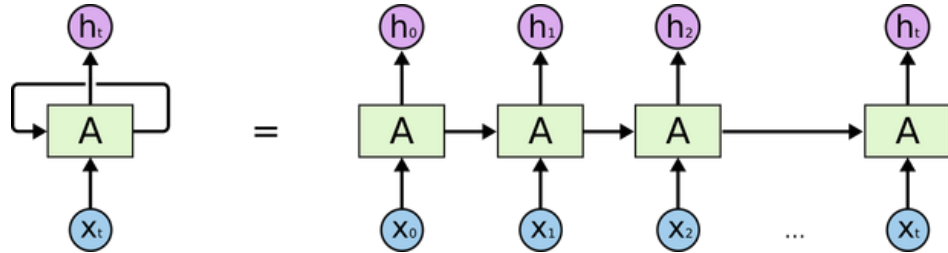


Figure 2: Basic LSTM Design [21]

Our research focuses on the detection of credit card fraud, a type of fraud in which a person or organization attempts to use a debit/credit card to gain profit without legal permission [5]. With the enormous increase in cashless transactions, whether using e-commerce or location stores, comes an increase in fraudulent rates [22]. According to the 2023 Federal Trade Commission report, fraud losses reached 10 billion US dollars, which is a 14% increase compared to 2022 [23]. It is essential to understand the challenges related to Credit Card Fraud Detection (CCFD) as fraudsters

always update their methods or ways that require banks to pay attention all the time to take appropriate actions. Prediction of fraud avoids the bank's profound loss or damage. Fraud detection is a binary classification problem where most of the transactions are normal and only very few are fraudulent transactions; due to the nature of this problem, datasets are highly imbalanced; when ML is directly applied to this problem, it leans more toward the majority class which imposes a serious difficulty for the ML model. Lastly, if the model misclassifies normal transactions as fraud, this will affect and harass the actual cardholder. To overcome the class imbalanced issue, a balancing technique needs to be applied to the dataset before feeding it to ML; methods like oversampling to increase the fraudulent class or under-sampling to remove the legitimate class are used in that manner [22].

Another challenge related to CCFD is data privacy and sensitivity concerns, as banks refuse to share their credit card transaction data with any central server for legal consideration [24] (According to the Data Protection Regulation Act [4]). ML models need massive amounts of data to train the model to generate appropriate outcomes. To overcome this challenge, an FL model was proposed to allow banks to train their data locally and share only weights to the central model [24].

FL process starts with each client downloading an initial global model for the following local training. Next, each client trains the model locally to enhance the global model for many rounds and sends the encrypted parameters to the central model. Then, these parameters would be averaged in the cloud to create a new global model that is retrieved back to local models. Lastly, this process is repeated many times until it reaches the desired performance or reaches the target time. With FL, no data is shared as each client trains the data locally, and that ensures data privacy [22]. The basic FL design is shown in **Figure 3**.

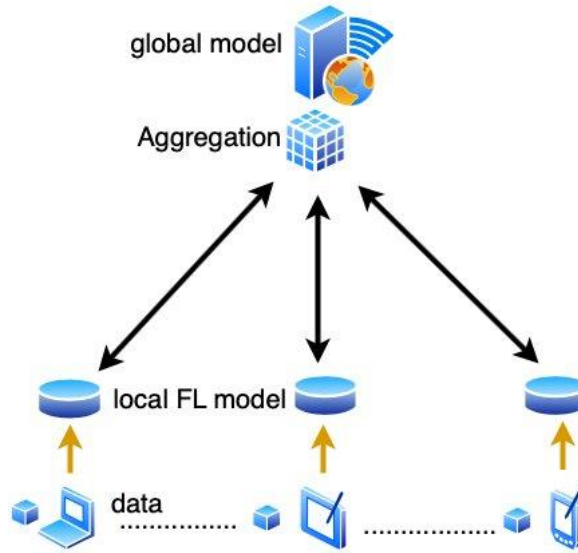


Figure 3: Basic FL Design [26]

To understand how the collaborative model functions, it's important to highlight the properties of Federated Learning (FL). One key aspect is its capability to connect different organizations that possess diverse data sources while safeguarding data privacy. For example, a bank that has data on client spending behavior can collaborate with a digital business platform that holds information about product features. This partnership can lead to personalized product suggestions based on the client's purchasing patterns. Therefore, we built a model that connects different businesses with different features without jeopardizing data privacy. Also, FL can deal with non-identically Independent Distribution (non-IID) and unbalanced data due to its heterogeneity property. Another aspect of FL is its equality state with each node in the system, which means larger or smaller data sources participated equally in the model, so FL does not lean toward massive data contributors. FL is divided into three categories: horizontal, vertical, and transfer learning.

Horizontal FL joins between nodes with common features and various data sizes; for instance, Google uses Horizontal FL on smartphones to predict text, as data have similar feature attributes. The other type is Vertical FL, which deals with the same clients but with different feature spaces. For example, a hospital predicts diabetes based on patient age, weight and other factors that can collaborate with fitness application that has different features of the same clients, like how many steps walked or diet type, to even enhance their prediction by considering other factors to predict diabetes. The last type of FL known as transfer learning FL, which unlike horizontal and vertical FL, has no connection to feature space nor client space. Transfer learning

share knowledge from one domain called the source domain to another one called the destination domain. A real-world application of transfer learning is FedHealth, which transfers information from one hospital to another to help predict diseases [25]. FL model faces many challenges; here, we focused on security risks and ways to mitigate them to create a robust FL model. Attacks are primarily divided into three types: Data poisoning, Model poisoning and Inferring attack. Firstly, data poisoning attacks by injecting malicious data into the training sets to jeopardize data integrity or create a bias that consequently affects the FL model. The other type of attack is a *model attack* that manipulates information (weights) sent to a global model by building a specific input trigger that produces wrong results. This attack also known as an adversarial attack directly affect the FL model. The final attack related to FL is an inferring attack, which uses model updates between client and server to know information about data used in training; this attack can be divided into a membership attack to know the client participated in the model and reconstruction attack to use updates to extract several features about the involved client.

To overcome security obstacles of FL, whether the attack is insider or outsider, different methods are used, such as differential privacy, that add noise whether on the client side or server side to hide data. Also, homomorphic encryption is used to encrypt updates to avoid leakage through the transmission process. This method imposes overhead on the encryption process that an alternative method called Locality-sensitive hashing uses a hash function to reduce overhead. A Secure Multi-Party Computation (SMC) is used to make sure that aggregation at server side is done securely as each client updates contributed in the model is not exposed by adding mask so that global model cannot learn sensitive information. In addition to the mentioned methods, many different hybrid methods are used to enhance FL security [25]. There are many open-source frameworks to use FL; famous one is TensorFlow Federated (TTF) by Google [27] and Federated AI Technology Enabler (FATE) produced by the Webank team [28].

CHAPTER III

RELATED WORK

3.1 OVERVIEW OF THE LITERATURE REVIEW

In this chapter, we explore recent literature related to advanced ML and the application of FL in credit card fraud detection. We use various databases to apply different search criteria directly related to this thesis's subject. The research focuses on papers from the last four years. We found many publications on the deep learning approach. In [29], different models evaluated, like ANN, Relational Graph Convolutional Networks (R-GCN), to detect fraud with results of 94.0% for ANN and 98.5% for R-GCN for accuracy. In [27], the paper suggests a DNN design of many hidden layers to enhance fraud detection. Matthews Correlation Coefficient (MCC) is used to determine the overall performance of the design. The paper achieved an MCC of 97.09%.

With paper [31], different ML and deep neural networks have been used to detect fraud and compare the results of each model using evaluation metrics like accuracy, precision, recall, and F1 score. The paper used different ML methods, such as Naïve Bayes, Logistic Regression, K-Nearest Neighbor (KNN), Random Forest, and Sequential Convolutional Neural Network. Results obtained for the models mentioned are 96.1%, 94.8%, 95.89%, 97.58%, and 92.3%, respectively.

Paper [32] suggests a GRU neural network that is computationally better than LSTM as it has fewer gates to inspect fraud patterns based on sequential dependencies. The proposed model achieved 98.72% accuracy, which is better than other models like LSTM and RNN.

Paper [33] used deep leaning Multilayer Perceptron (MLP) with four layers and a different set of neurons within each layer; the accuracy value was 93.996% on the test dataset. Paper [34] uses many ML models along with AdaBoost and compares results with and without ensemble learning; the model performs very well with 99% accuracy.

Another advanced method of ML is ensemble learning; in the paper [35], the authors used different models together like Decision Tree, Naive Bayes, Logistic Regression, and Support Vector Machines (SVMs) to enhance the model, the accuracy of this model was 99.9%. Another paper used ensemble learning [36] that combined three algorithms: Logistic Regression (LR), Decision Tree (DT), and Stochastic Gradient Descent (SGD), using the voting base technique to enhance evaluation performance. A recall of 97% was achieved compared to individual methods that evaluated 0.957198, 0.953521, and 0.953669 consecutively. Another paper [37] used bagging-based ensemble learning with RF as the base learner with an additional pipeline method to enhance this model; accuracy achieved 99.9999%, overcoming standalone models.

Many studies related to ML with FL application in the domain of credit card fraud detection are in the literature. In [35], an MLP deep neural network is used to train local banks, and then the gradients are aggregated by a global model that also uses MLP DNN with federated average to combine updates from different banks; the paper uses vertical FL to create a collaborative environment for banks to share knowledge without exposure of sensitive data. For model evaluation, accuracy reached 97% after more than 60 different rounds. Paper [39] presented FL with three banks that train data locally using different ML and deep neural networks RF, CNN, and LSTM, then send the updates to the global server. This model combines FL with blockchain to guarantee privacy and data protection. The model achieves an accuracy of 0.97%. Another paper [40] combined the power of ML with blockchain to create a decentralized global model to get rid of single-point failure in centralized FL. The model used LR for credit card fraud detection with FL ensemble aggregation methods based on the quality of the local model. Despite the innovative model, accuracy performance was poor, meaning more future improvement is needed.

[24] used Federated Average (FedAvg) with three deep neural networks, CNN, MLP, and LSTM, along with methods to combat the dataset's imbalanced nature. The model accomplished accuracy of 99.11, 98.28, and 95.78 for CNN, MLP, and LSTM consecutively. Another paper [41] used Graph Neural Network (GNN) with FL to create a collaborative environment to enhance CCFD. The paper suggested adjustable FL aggregation based on cosine similarity to elevate the aggregation stage efficiency further. Another method for federated learning aggregation is presented in the paper [39], which calculates the distance of weights between local and global models and

then applies aggregation. Papers [42] [43] used FL non-IID with FedAvg Epsilon Cluster Selection (ECS) method for robust aggregation where the model clusters the participated banks into a large cluster and a smaller one, then a Silhouette score is calculated by the global model to select whether the large cluster considered the benign banks and neglect small cluster as they considered malicious banks or use all results in aggregation. This model ensures that aggregation is robust, considering 100% label flip malicious banks. Paper [43] further expands the types of attacks to the FL model, assuming partially malicious banks label flip or round attacks and do experiments using the same filter-based method. In [42] the experiment was conducted on 10 participating banks applying different numbers of malicious banks and evaluating the F1 score each time, once with FedAvg and the other with FedAvg and ECS. While with [41], many experiments were conducted once with partial label flip with different percentages and others with partial flip and round base attack, the F1 score is used as evaluation metrics.

[44] used a method to detect corrupted clients by using historical updates consistency through training rounds to predict updates of the next round, then compared the expected results with those received to identify the difference. The results are stored in the suspicious scores. K-means with gap statistics are used to cluster clients based on score results calculated across multiple rounds for each client to spot noisy clients and then remove them from model aggregation. We inspire multiple models from the above-mentioned papers to build our framework by examining each selected paper thoroughly. A comprehensive review informed us of the development of our credit card fraud detection model and the existing machine-learning approaches in credit card fraud detection. Our investigation focused on two primary methodological streams: DNN [30] and ensemble ML [34]. Given the critical importance of data security and confidentiality in financial institutions, we examined federated learning (FL) approaches, which offered robust privacy preservation mechanisms [41],[45]. We proposed our FL method based on these two baseline approaches, FL cosine similarity aggregation and Federated Average Distance Weighted Aggregation (FedAvg DWA). Furthermore, we investigated an advanced post-processing technique for detecting noise in the FL model [44]. The review of existing methodologies provided crucial and valuable insights that shaped our research direction and model development.

3.2 DEEP LEARNING APPROACH TO CREDIT CARD FRAUD DETECTION

The research presented in [30] introduces an optimized DNN architecture, as shown in **Figure 4**, specifically designed for credit card fraud detection. The study addresses two fundamental challenges in fraud detection through a hybrid sampling methodology: dataset imbalance and model overfitting. This approach integrates under-sampling and over-sampling techniques to achieve dataset equilibrium. The empirical validation was conducted using the Kaggle *creditcard* dataset, comprising 284,807 transactions, on which only 492 (0.17%) represented fraudulent activities. Implementing this balanced sampling strategy yielded substantial improvements in model performance metrics. MCC achieved a remarkable 97.09%, demonstrating the method's robust performance in handling inherently imbalanced fraud detection scenarios.

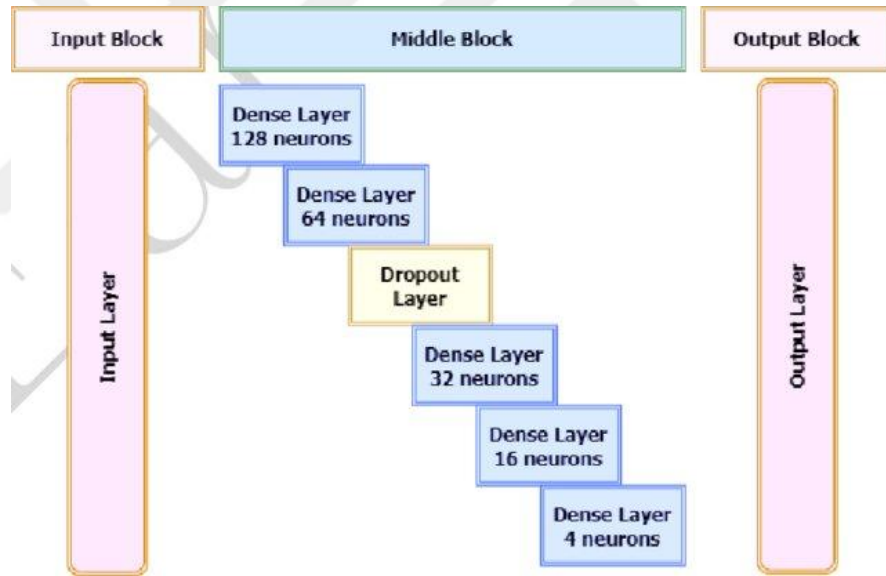


Figure 4: DNN Design from Paper [30]

3.3 ENSEMBLE LEARNING APPROACH TO CREDIT CARD FRAUD DETECTION

In [34], the researchers evaluate advanced ML methodologies for addressing fraud detection challenges in financial systems, particularly emphasizing highly imbalanced datasets. The study investigates multiple algorithmic approaches, including RF, Extreme Gradient Boosting (XGBoost), DT, and various methods enhanced through Adaptive Boosting (AdaBoost), utilizing the Kaggle *creditcard* dataset as its primary experimental framework. The researchers implemented the

Synthetic Minority Over-Sampling Technique (SMOTE) to address the severe class imbalance, where fraudulent transactions constituted merely 0.172% of the total observations. SMOTE's synthetic sample generation methodology proved instrumental in mitigating overfitting risks by creating non-duplicative synthetic data points for the minority class. The study employed a combination of ensemble and boosting techniques. The experimental framework encompassed multiple ML architectures:

- RF
- XGBoost
- LR
- DT
- Extra Trees (ET)

AdaBoost integration further enhanced these base classifiers, significantly improving performance across all models. Model assessment utilized a comprehensive metric suite including accuracy, recall, precision, Matthews Correlation Coefficient (MCC), and Area Under the Curve (AUC). The empirical results demonstrated exceptional performance for RF and ET models when integrated with AdaBoost, achieving:

- MCC scores approaching 0.99
- Overall accuracy of 99.95%

The research methodology incorporated dual dataset validation, utilizing both the Kaggle and synthetic datasets, thereby establishing robust model generalizability. This validation approach verified the adaptability and reliability of the methodology proposed in the study. The study's successful implementation of SMOTE combined with AdaBoost to address the class imbalance, particularly the high-performing SMOTE+RF+AdaBoost integration, provides a foundational framework for our study. The demonstrated robustness across multiple performance metrics positions this methodology as a promising candidate for federated learning applications in privacy-sensitive environments where secure fraud detection mechanisms are essential [46].

3.4 FEDERATED LEARNING WITH GRAPH NEURAL NETWORKS APPROACH TO CREDIT CARD FRAUD DETECTION

This paper [41] combines FL with Graph Neural Networks (GNN) to address critical challenges in credit card fraud detection (CCFD). The researchers benefit from the combination of graph neural networks and FL to detect fraud patterns through various banks while keeping the data private as each bank trains its data locally without sharing it with the global server. The main design of this model is the graph-based method, where the dataset is transformed into a graph with nodes representing each customer and edges connecting between nodes. Each edge is assigned a weight to show the relationship between edges. The transformation of the dataset to this form helps discover patterns and reveal hidden connections between transactions, which helps detectability. Different banks have various fraud patterns, as each has its customers. It is hard to transfer learning between each other due to privacy restrictions; the paper suggests an FL model based on Convolutional Feedforward Generative (CFG) design to tackle this issue and let banks train their data locally and only send weights so no raw data is shared. The FL model then aggregates the weights from local banks using cosine similarity-based aggregation that ensures the fair contribution of each model knowledge through a series of training stages while considering different sizes of datasets. The experiment result of this model done on *IEEE-CIS* and *credicard* real datasets showing notable performance enhancements:

- achieved a 33.9% enhancement in recall metrics compared to baseline methodologies,
- validated the efficacy of the GNN-FL hybrid architecture in maintaining robust fraud detection capabilities,
- demonstrated successful preservation of data privacy protocols in sensitive financial environments.

3.5 FEDERATED AVERAGING WITH DISTANCE-BASED WEIGHTED AGGREGATION APPROACH TO CREDIT CARD FRAUD DETECTION

In [45], the paper established a novel algorithm called Federated Averaging with Distance-Based Weighted Aggregation (FedAvg-DWA), which uses distance weighted to aggregate weights from local clients. This method produces an improvement compared to conventional standard aggregation in FL. The power of this method lies in its ability to handle the challenging nature of fraud datasets where

legitimate transactions compose the majority of the data, and only a small portion of transactions are fraud. The unfair ratio between real and fake transactions introduces a serious challenge to the standard federated learning models, such as FedAvg and Federated Proximal (FedProx), as these methods aggregate information without considering the imbalanced ratio of credit card fraud datasets. For that, the paper proposed an enhanced policy to deal with the restriction of the known federated learning models:

Local Training Enhancement:

- Implementation of dynamic weight assignment mechanisms for minority (fraudulent) samples
- Optimization of model sensitivity towards underrepresented classes
- Integration of adaptive learning parameters for improved minority class detection

Global Aggregation Refinement:

- Introduction of distance-based weighted aggregation methodology
- Implementation of intelligent down-weighting mechanisms for divergent clients
- Development of robust anomaly detection protocols within the aggregation framework

The model underwent testing using European cardholder transaction data, representing real-world class imbalance scenarios. Performance evaluation employed a comprehensive metric suite:

- Accuracy metrics for overall model performance
- Recall measurements for fraud detection sensitivity
- F1 Score for balanced performance assessment
- Matthews Correlation Coefficient (MCC) for binary classification robustness

The FedAvg-DWA model demonstrates several key architectural advantages that significantly enhance its effectiveness in addressing the challenges of imbalanced fraud detection scenarios. A detailed examination of these fundamental benefits is listed as follows:

Enhanced Minority Class Detection:

- Optimized sample weight adjustment during training phases
- Improved sensitivity to fraudulent transaction patterns
- Robust handling of extreme class imbalances

Sophisticated Data Heterogeneity Management:

- Advanced distance-based weighted aggregation protocols
- Effective integration of diverse client data distributions
- Robust handling of client-specific data characteristics

Enhanced Model Generalizability:

- Improved variance handling in client contributions
- Enhanced cross-dataset fraud detection capabilities
- Robust performance across heterogeneous data distributions

CHAPTER IV

METHODOLOGY

This chapter introduces an extensive conceptual basis for combining ML methods with FL to build a collaborative environment to share knowledge between banks. Many ML models were suggested, along with two aggregation methods for FL. This research focuses on two basic principles: improve cooperation between banks using the FL concept, investigate the effect of adding noise with various percentage levels on the federated learning model, and integrate a detection method to enhance the efficiency of the FL model. The methodology addresses a common challenge related to fraud detection datasets, such as imbalanced datasets. Moreover, a set of performance metrics is presented to evaluate the model effectively. The methodology process is presented as a flowchart shown in **Figure 5** below. The methodology starts by cleaning the dataset from missing or null values then change the categorical fields into numbers as ML only accepts numerical values. A features engineering is applied to select the appropriate features and remove redundant ones. The process of cleaning, changing text values and the application of feature engineering is called preprocessing. The next stage is partitioning the dataset into training and testing with particular percentages in order to feed ML the training set to identify classes while the test set used for evaluating the model performance. As the dataset related to fraud domain naturally imbalanced, different balancing techniques applied like SMOTE, hybrid balancing to balance the dataset to avoid bias toward majority class and create balance. After that, the balanced training part is used to train the ML models such as ANN, LSTM and RF+AdaBoost. Moreover, a federated learning collaborative environment with different aggregation methods is adopted to share knowledge between banks without breaking data privacy. Lastly, a post-processing stage is applied to add noise at various levels to investigate the effect on the FL model. Additionally, a detection method is used to detect the malicious client. The model evaluated on the test dataset to accurately assess the model performance.

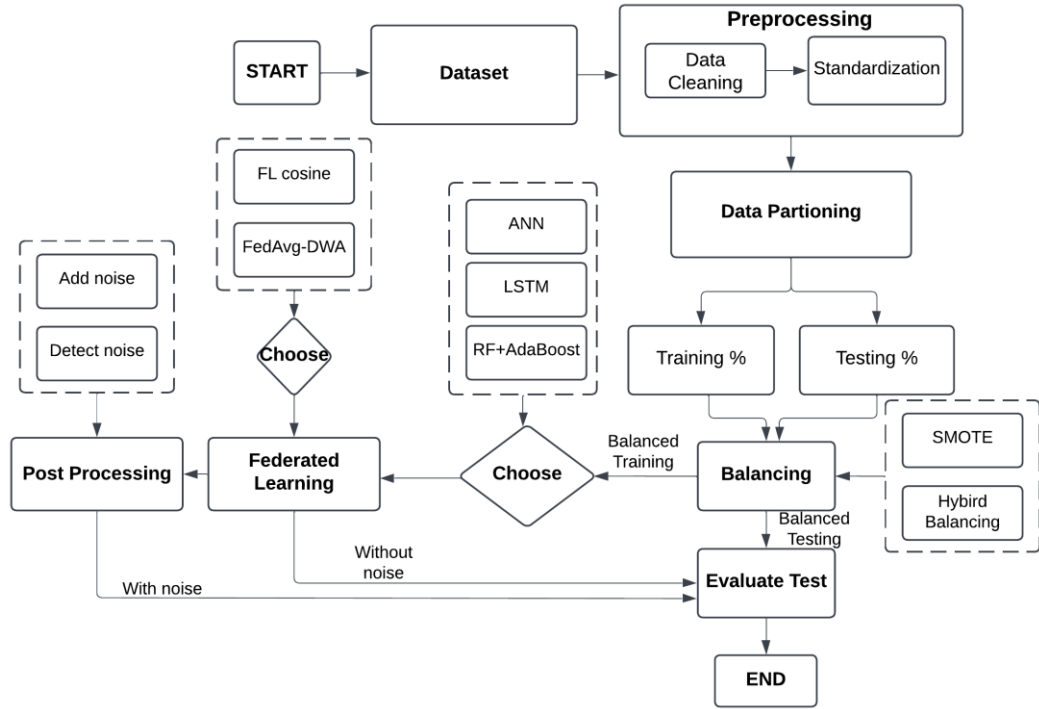


Figure 5: Methodology Process Flowchart

4.1 DEALING WITH THE IMBALANCED NATURE OF FRAUD DATA

One of the most complicated challenges that ML faces in the fraud detection domain is the imbalanced nature of datasets. This main obstacle limits ML models as the number of regular transactions far exceeds the number of fake ones. The disproportional ratio, where the financial dataset is highly skewed, affects the performance of ML drastically, and for that, many methods are used to handle this limitation. Standard ML algorithms are designed to optimize overall accuracy metrics, inadvertently introducing a critical limitation in fraud detection scenarios. When presented with an imbalanced dataset, these algorithms develop a systematic bias toward the majority class (legitimate transactions) during the training phase. This bias stems from the algorithms' fundamental objective of minimizing overall error rates. In a dataset where legitimate transactions might represent 99.9% of all cases, an algorithm could achieve seemingly impressive accuracy by simply classifying all transactions as legitimate while failing to identify fraudulent activities [47].

This algorithmic behavior becomes particularly problematic in fraud detection applications, where the minority class (fraudulent transactions) carries significantly higher importance from a business and security perspective. The cost of misclassifying a fraudulent transaction as legitimate (false negative) typically exceeds that of

misclassifying a legitimate transaction as fraudulent (false positive). For instance, failing to detect a large fraudulent transaction could result in substantial financial losses, while temporarily flagging a legitimate transaction for review causes minimal disruption [47].

To address these inherent challenges, our methodology implements various dataset-balancing techniques. These approaches aim to create a more representative training dataset that enables the model to learn effectively from both classes, ensuring robust fraud detection capabilities while maintaining reasonable false positive rates. The following sections detail the specific balancing methods employed in our research, including their theoretical foundations and practical implementations.

4.1.1 Synthetic Minority Over-sampling Technique (SMOTE)

SMOTE is considered one of the most important approaches used in research to handle the imbalanced nature of the dataset, especially in the fraud detection domain. The power of this method lies in its ability to produce synthetic samples in an effective way to increase the number of rare classes without affecting data integrity, unlike a straightforward oversampling method that just redundant minority class samples.

The basic concept of SMOTE is systematically producing new samples based on feature space. The method focuses on the minority class and finds the k -nearest neighbor near the similar classes. A formula is then used to calculate the difference between the class and its nearest sample to introduce an artificial sample between them. This ensures that the generated samples distributed across the dataset are meaningful. The synthetic sample generation process occurs through feature-space interpolation. Mathematically, SMOTE creates new instances by selecting a point along the line segment that connects a minority class instance to one of its k -nearest neighbors. This can be visualized as choosing a random point between two existing minority class samples in the feature space [48]. The process can be represented as:

$$Z = X_0 + W(X - X_0) \quad (4.1)$$

Where Z is the new sample, X_0 is the minority instance, X is a neighbor, and W is a random value between 0 and 1 [48].

This interpolation approach offers several key advantages:

- **Feature Value Coherence:** The synthetic samples maintain realistic relationships between features since they are derived from actual instances
- **Pattern Preservation:** The generated samples reflect genuine patterns present in the minority class
- **Variance Introduction:** The random interpolation adds beneficial variance to the dataset without introducing noise

The application of SMOTE achieved the desirable balanced dataset to empower ML to learn various attributes of fraudulent transactions. The generated instances extend the capability of ML by exposing it to more minority class samples, which helps elevate decision boundaries to make better and more reliable detections [48]. Applying SMOTE balances the dataset, by increasing rare samples presentation within the dataset, provides a practical base for ML to enhance their ability to detect minority classes while maintaining the patterns and relationships between transaction features. The systematic approach to adding synthetic samples helps models to effectively learn the fraudulent behavior during the training stage [34].

4.1.2 Hybrid Balancing (SMOTE+RUS)

[49] Combines SMOTE (Synthetic Minority Oversampling Technique) with RUS (Random Under Sampling). This approach introduces an improved merged method to handle the class imbalances presented in credit card detection models. This method integrates the power of both models (oversampling and under-sampling terminologies) to build effective and well-distributed samples while keeping the critical structure from both classes. The composite steady process is done through a two-phase approach. In the first phase, SMOTE generates a synthetic minority class to balance its rare appearance in the dataset using a systematic approach using a k-nearest neighbor to create samples in an organized way. SMOTE produces new samples using a method that assists the ML model in recognizing the features of fraudulent schemes. The added samples keep the characteristics and connections between different features while imposing a well-balanced representation of minority classes in the dataset. For the second phase of this approach, an RUS is applied to lessen the dominant class (legit transactions). The reduction was implemented by selecting a random sample of the majority class. The goal of reducing the number of legitimate transactions to create a balance in the dataset is so that the ML model does not lean toward the most presented

class in the dataset, which results in overfitting the majority class while neglecting the minority class. The RUS guarantees that the chosen dominant class maintains a well-distributed representation within the dataset.

The combination of these two approaches gives the opportunity to use the advantages proposed by each one and construct an effective and well-distributed dataset to support the ML process. The combination of SMOTE and RUS helps solve the restriction involved within each method. SMOTE-generating synthetic instances may lead to noise that affects the overall dataset, and here comes the role of RUS, which reduces the effect of noise by random under-sampling. While with RUS, applied by itself may lead to the loss of precious information of legitimate class, the SMOTE application that generates samples of minority classes creates a balanced dataset that helps the ML model to understand the patterns well and produce an efficient output. The effectiveness of this hybrid approach has been demonstrated through significant improvements in multiple performance metrics [9]. Specifically, the methodology achieves:

- Enhanced recall rates, indicating better detection of actual fraud cases
- Improved precision, reducing false fraud alerts
- Better model generalization to new, unseen transactions
- Reduced overfitting through the balanced representation of both classes

This balanced approach to dataset preparation has proven particularly valuable in real-world fraud detection scenarios, where the cost of missing fraudulent transactions must be weighed against the operational impact of false positives. The methodology creates an optimal learning environment that enables models to develop robust fraud detection capabilities while maintaining practical applicability in production environments.

4.2 DATASETS USED IN THIS STUDY

Our methodology was validated using our *PB_dataset* and the benchmark Kaggle *creditcard* fraud dataset.

4.2.1 Original Iraqi Bank Dataset

This study obtained a dataset of actual credit card transactions from an Iraqi bank for the second half of 2023. The dataset contained 27,230 records with only 2145 fraud transactions (the dataset is imbalanced), as shown in **Figure 6**, with 38 features

and the alert field (fraud yes or no). The data were obtained from the bank by anonymizing the identity information.

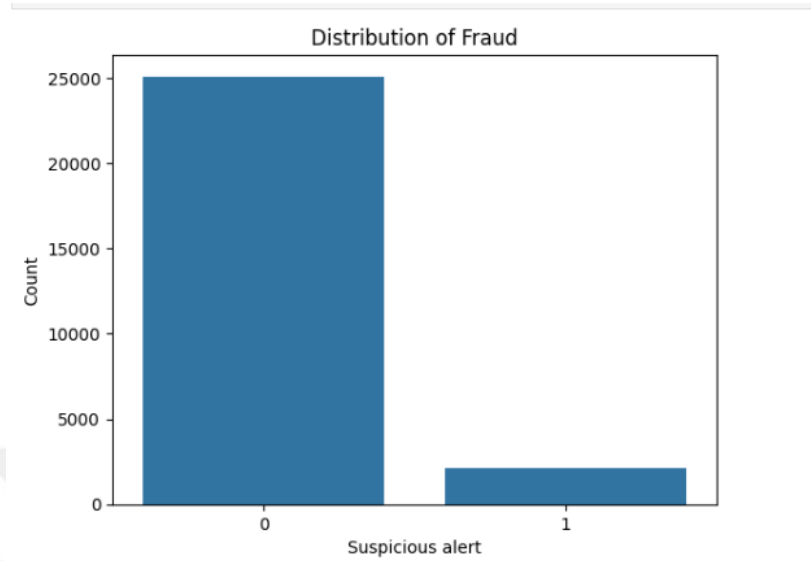


Figure 6: Distribution of Normal and Fraudulent Transactions

As the dataset is original and related to actual cardholders, this imposes challenges of privacy and confidentiality as no data should be shared or published. The Central Bank of Iraq (CBI) and its Anti-Money Laundering (AML) and compliance teams have implemented strict monitoring systems to combat fraudulent transactions, particularly those exploiting currency arbitrage opportunities. A standard scheme involves newly issued payment cards loaded with Iraqi Dinar (IQD) and distributed to surrounding districts where perpetrators attempt to convert the funds to US Dollars (USD) at the official exchange rate. The significant disparity between the official and market exchange rates incentivizes this exploitation [50]. As illustrated in **Figure 7**, analysis of transaction data reveals that these fraudulent activities typically involve multiple bulk point-of-sale (POS) transactions with no corresponding exchange of goods or services. The bank dataset underwent a comprehensive preprocessing pipeline, employing domain expertise to ensure data quality and optimize feature representation for fraud detection analysis. The preprocessing methodology encompassed several critical stages to transform raw financial data into a structured format suitable for ML applications.



Figure 7: Collect USD by doing Transactions with no Product

The initial data cleaning phase focused on handling missing values through systematic identification and treatment while simultaneously validating the data types of each feature to ensure consistency and accuracy. A thorough feature assessment was conducted to identify and remove redundant fields, streamlining the dataset while preserving essential information. The original dataset contained various financial indicators, including transaction amounts, merchant identification details (such as merchant names, location, and type of the business), capture method of the transaction, currency of the transaction, type of the transaction (purchase or ATM withdrawal), time, date of the transaction and account-specific information. To standardize the data representation, categorical variables were transformed into numerical formats using label encoding, making them compatible with ML algorithms. A particularly important aspect of the feature engineering process involved the temporal dimension. The original timestamp feature ("Time") was decomposed into three distinct temporal components - hours, minutes, and seconds - enabling the capture of potential temporal patterns in fraudulent activities. After completing all preprocessing steps, the final dataset was distilled to 14 carefully selected features plus the target variable. The features are ACCT_NUMBER, MERCHANT_COUNTRY_CODE, TRANSACTION_TYPE_CODE, SERVICE_CONTRACT_ID, TRAN_CURRENCY_CODE, SETTLEMENT_AMOUNT_GR, MCC_CODE, CAPUTRE_METHOD, MERCHANT_NAME_ENCODED, trans_day, trans_month, hour, minute, and second. This refined dataset balances maintaining comprehensive

transaction information and ensuring computational efficiency for subsequent modeling stages.

4.2.2 Kaggle Dataset

In addition to our original bank dataset, the experimental validation of our methodology leverages the widely recognized Kaggle *creditcard* dataset [51]. The dataset contains 284,807 transactions characterized by 31 features, with only 492 transactions (0.17%) labeled fraudulent, representing a significant class imbalance. To preserve privacy, 28 original features (V1-V28) were transformed using Principal Component Analysis (PCA) as these features were anonymized by the source of the dataset for privacy concerns, while two features—transaction time and amount—remained unmodified. The dataset is checked for missing or null values. After transformation we kept the animalized 28 features as it is and transform time feature into hour, minute, and second of the transaction. Moreover, the amount feature is standardized to avoid bias. The resulted dataset has 30 features and a target field (fraud yes or no). This publicly available dataset has become a standard benchmark in fraud detection research due to its realistic representation of transaction patterns and class distribution typically encountered in financial fraud detection.

4.3 MODEL ARCHITECTURE

This section extensively investigates our model's architectural designs and their fundamental components.

4.3.1 Artificial Neural Network (ANN) Model

4.3.1.1 Overview of the Model

ANN model design [30] aims to create an effective method to handle classification problems to differentiate between normal and abnormal transactions. The model takes advantage of the neural network's ability to detect fraud and the powerful capabilities of hidden layers for deep neural networks to extract features that help understand the nature of rare classes. Due to the nature of the dataset, where the presentation of legitimate transactions is far more compared to fraud, this imposes a challenging factor for ML to forecast the minority class. To tackle this issue, the researchers suggest a hybrid approach of using SMOTE and RUS to balance the

dataset while keeping the structure and essential features within the dataset. The balancing of the dataset was done in two phases:

- Applying SMOTE: produce a synthetic sample effectively to increase the number of rare classes without affecting data integrity, unlike a straightforward oversampling method that just redundant minority class samples. The basic concept of SMOTE is to systematically produce a new sample based on feature space. The method focuses on the minority class and finds the k-nearest neighbor near the similar classes. This assists the ML model in understanding rare classes presented in the dataset.
- Applying RUS: lessen the dominant class (legit transactions). The reduction is done by selecting a random sample of the majority class to take off. The goal of reducing the number of legitimate transactions is to create a balance in the dataset without removing much information. The merge between SMOTE and RUS produced a well distributed presentation of both classes in the dataset while keeping the crucial patterns that identify each class type.

The design of ANN architecture was created based on many factors like the size of the dataset and the number of features to comply with the characteristics of the dataset. Depending on these factors, the design constructed and the number of layers used are decided. Also, the number of neurons within each layer is crafted to adapt to the needs of the dataset used. The model ensures the best design to overcome underfitting problems where the design is too simple and cannot catch complex relationships. It also addresses the issue of overfitting, where the model becomes too specialized to the training data, resulting in unrealistic predictions. The balancing effect of applying a hybrid method composed of SMOTE and RUS ensures fair distribution of both classes in the dataset, which in turn helps ML to make better decisions without affecting one class over the other. The sequential deep neural network layers ensure the model analyzes the patterns closely. As the first layer detects the signs of fraudulent activities, the inner layers extract features and learn from them to reach the optimal result. This learning through many stages ensures that even complicated or complex behaviors are detected through various and deep understanding through many phases.

4.3.1.2 Data Preprocessing and Balancing

The dataset followed a structural, logical approach for partitioning to maximize model training and performance assessment. A random split is applied to partition the dataset into training, validation, and test dataset using stratified sampling to ensure fair distribution across the dataset. Firstly, 10% of the dataset was kept for the final evaluation, while the rest of the dataset, 90% portion, was further split up into training (70%) and validation (20%) subsets. An enhanced balancing technique was followed through all dataset subsets to handle the class imbalance commonly faced in fraud detection datasets due to the small presentation of fraudulent transactions rather than legitimate ones. This approach combines the oversampling of minority fraud cases with the under-sampling of most legitimate transactions, ensuring balanced exposure during the model's learning process. For optimization of model performance and consistency in training, feature standardization was implemented using the *StandardScaler* method. This transformation normalizes variables of different scales to a common scale, facilitating efficient learning of model parameters and promoting stable convergence during training. The balanced dataset is fed to the ANN model in a form of vectors with dimensions equivalent to the number of features in the dataset (14 features).

4.3.1.3 Hyperparameter Tunning

To select the best parameters for the ANN design, we used the *hyperopt* library. We tried various values to decide the number of neurons in each dense layer, such as dropout value, batch size, learning rate, optimizer, and batch normalization. Evaluation is carried out based on the best validation loss and accuracy value. The design uses the best parameters to achieve the best performance.

4.3.1.4 Architecture Design

The architecture of the proposed DNN model consists of an input layer, five hidden dense layers with 64, 32, 16, 8, and 4 neurons, respectively, and an output layer with a sigmoid activation function. The model is designed to analyze features through dense layers. Each dense layer is activated using the ReLU activation function to introduce non-linearity, which aids in capturing complex patterns in the data. A dropout layer of rate 0.2 is included in the design to avoid overfitting; the value of this layer is decided using hyperparameter tuning. The benefit of this layer is that it selects 20% of

neurons and sets them to zero randomly. The final output layer contains a single neuron, as fraud is a classification issue activated using the sigmoid function to select either zero or one. The design is illustrated in **Table 1**, showing the detailed structure of the suggested DNN design, containing the input, hidden layers, and output shapes. Also, the learnable parameters obtained by each layer are presented.

Table 1: Design of ANN Model

Sl.No.	Layer (type)	Input Shape	Output Shape	Learnable Parameters
1	Input Layer	(None, 14)	(None,14)	0
2	Dense Layer_1 ReLU Activation	(None, 14)	(None,64)	960
3	Dense Layer_2 ReLU Activation	(None,64)	(None, 32)	2080
4	Dropout Layer Rate: 0.2	(None, 32)	(None, 32)	0
5	Dense Layer_3 ReLU Activation	(None, 32)	(None, 16)	528
6	Dense Layer_4 ReLU Activation	(None, 16)	(None, 8)	136
7	Dense Layer_5 ReLU Activation	(None, 8)	(None, 4)	36
8	Output Layer Sigmoid	(None, 4)	(None, 1)	5

4.3.1.5 Model Training

The model's training configuration was optimized for robust learning and generalization capabilities. The assembled phase incorporates binary cross-entropy as the loss function and the Adam optimizer, which dynamically adjusts learning rates for efficient parameter updates and stable convergence. Training dynamics were controlled through several key mechanisms:

- the early stopping protocol monitored validation performance metrics, automatically terminating training when improvement plateaued to prevent overfitting,
- a batch size of 32 was selected to balance computational efficiency with gradient estimation accuracy,
- the training process extended up to 100 epochs, with the model's performance continuously evaluated on the validation subset.

The data employment approach followed a structural division by assigning 75% of the preprocessed dataset for model training while keeping the rest of the dataset, which is 25%, to be kept aside to use in model evaluation. This followed distribution guarantees enough datasets to be used by the ML model for training while preserving enough data to assess performance further. This training framework was

designed to maximize the model's learning of meaningful patterns while retaining generalization capability for real-world fraud detection situations. The data utilization strategy followed a structured partition, allocating 75% of the preprocessed dataset for model training while reserving 25% for ongoing validation. This distribution ensures robust performance assessment during training while maintaining sufficient data for effective parameter optimization.

4.3.2 Long Short-term Memory (LSTM)-based Deep Learning Model

4.3.2.1 Overview of the Model

We adopted enhanced long-term memory (LSTM) architecture and a deep neural network learning application with different layers to improve forecasting abilities. The LSTM is an RNN model that uses historical dependencies in a dataset to catch patterns that evolve over time. The design ensures that the model can capture the relationship between patterns, especially complex ones leveraging time sequences. The proposed model uses this facility granted by this type of neural network to consider time dependencies. The design can detect the complex relationship between features over sequential time through a series of layers. The powerful aspect of LSTM that differentiates it from the traditional neural network is the fact that it contains three gates: input, output, and forget gates; these gates help the model to consider the time dependency between patterns as each input will have feedback from the previous one to help the model make better analysis than the decisions done based on historical dependencies which are crucial for fraud detection in the financial sector.

4.3.2.2 Data Preprocessing and Balancing

The preprocessing stage is done in a similar structural way that is applied to the ANN model, which combines SMOTE and RUS to balance the dataset. The dataset underwent identical splitting ratios for test, training, and validation subsets. Similarly, the hybrid balancing technique was applied to address the class imbalance, ensuring balanced data distribution throughout the model's learning process and maintaining methodological consistency across different neural network architectures. The dataset is reshaped into three-dimensional structure (samples, timesteps, features) allowing LSTM to process temporal dependencies.

4.3.2.3 Hyperparameter Tunning

Hyperopt is used to investigate the best parameters for the LSTM model, the number of neurons for each LSTM model layer, the number of neurons in dense layers, and other parameters like learning rate, batch size, dropout values, and batch normalization. The decision is made depending on the best validation loss value. The best parameters are further used in the architecture design.

4.3.2.4 Architecture Design

The LSTM model architecture implements a multi-layered structure designed explicitly for sequential pattern recognition in financial transaction data. The architectural framework encompasses several specialized layers, each optimized for specific temporal feature processing and fraud detection. The core architectural components are structured as follows:

The foundational input layer transforms feature vectors into sequence format, feeding into the primary LSTM layer configured with 64 neurons. This layer employs a *tanh* activation function, normalizing input and hidden state values within the -1 and 1 intervals for optimal network regularization. The "return sequences" parameter is enabled to maintain temporal information flow through the network. The secondary processing stage incorporates the following:

- A dropout layer (20% rate) for initial regularization
- A second LSTM layer with 32 neurons and tanh activation for deeper temporal feature extraction
- Another dropout layer (12% rate) for additional regularization against overfitting

The architecture then transitions to a dimensionality reduction and feature abstraction phase through a cascade of dense layers:

- First dense layer: 16 neurons with ReLU activation
- Second dense layer: 8 neurons with ReLU activation
- Third dense layer: 4 neurons with ReLU activation

The architecture reaches the highest point in an output layer comprising a single neuron with sigmoid activation, producing probability scores for binary classification of transactions. The comprehensive structural specifications, including layer-wise dimensions and learnable parameters, are detailed in **Table 2**.

This architectural design optimizes the model's capability to capture immediate and long-term temporal patterns in transaction sequences while maintaining robust generalization through strategic regularization mechanisms.

Table 2: Design of LSTM Model

Sl.No.	Layer (type)	Input Shape	Output Shape	Learnable Parameters
1	Input Layer	(None, 1, 14)	(None, 1, 14)	0
2	(LSTM Layer 1)	(None, 1, 14)	(None, 1, 64)	20224
3	Dropout Layer Rate: 0.2	(None, 1, 64)	(None, 1, 64)	0
4	LSTM Layer 2	(None, 1, 64)	(None, 32)	12416
5	Dropout Layer Rate: 0.12	(None, 32)	(None, 32)	0
6	Dense Layer 1 ReLU Activation	(None, 32)	(None, 16)	528
7	Dense Layer 2 ReLU Activation	(None, 16)	(None, 8)	136
8	Dense Layer 3 ReLU Activation	(None, 8)	(None, 4)	36
9	Output Layer Sigmoid Activation	(None, 4)	(None, 1)	5

4.3.2.5 Model Training

The training phase was enhanced using two key factors that helped the model to improve the overall performance. Beginning with implementing the Adams optimizer, which acts as an activation function for the neural network layer, the Adams optimizer ensures that the adjustment is adapted to the input dataset to make the learning process more appropriate. The other important factor that elevated the model is the early stopping technique. This approach ensures the model does not overfit, leading to unrealistic results. This is done by continuously checking the validation metrics and instinctively stopping the model when the value loss remains unchanged to avoid overfitting. Applying these approaches provides a powerful model that ensures training is done correctly. As a result, the prediction of unseen data would be reliable.

4.3.3 RF+AdaBoost Model

4.3.3.1 Overview of the Model

This model integrates Random Forest (RF) with AdaBoost for ensemble learning. The model's basic design is shown in **Figure 8**.

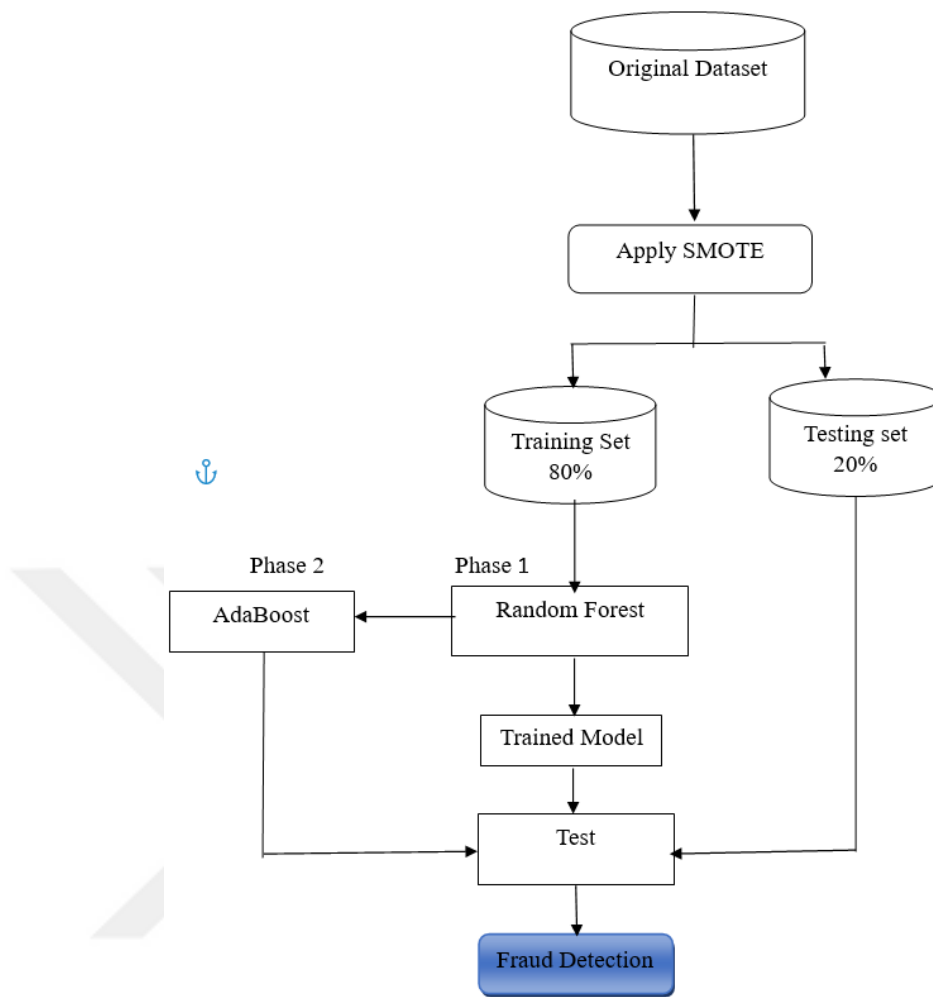


Figure 8: RF+AdaBoost Design [34]

4.3.3.2 Data Preprocessing and Balancing

To handle the naturally imbalanced dataset, we applied SMOTE, which stands for Synthetic Minority Over-sampling Technique, as the primary solution to balance the dataset. This method produced a synthetic sample effectively to increase the number of rare classes without affecting data integrity and keeping the critical features of the dataset without distortion. SMOTE ensures that both classes are well distributed in the dataset; after achieving that, a methodological division is done on the dataset into training and testing parts. The split was on an 80 to 20 ratio. This partitioning allows the ML model to learn enough from the training set and then validate the learning process on unseen test datasets to evaluate the model further.

4.3.3.3 Architecture Design

The design of this model combines Random Forest (RF) as a base learner with the AdaBoost method as the enhanced learner. The goal is to design a well-structured model that integrates two models. In addition, the base learner is precisely shaped to determine the optimal hyperparameters like *n_estimator* to manage ensemble size, *max_depth* to handle the depth and hierarchy of the tree, and *class_weight* to deal with class weight examinations. Using the AdaBoost method, which was carefully crafted to select appropriate parameters, we enhanced the RF model even more with the advantages of AdaBoost, which changes the weak learner into a strong one by changing sample weights depending on prior classification results. This model achieves a powerful combination by taking advantage of RF's ability to select the best features along with the AdaBoost adaptable learning method.

4.3.3.4 Model Training

To reach the best-performed model, a K-Fold Cross-Validation method was carried out to ensure we selected the best parameters to achieve better results. The training part of the dataset goes through a well-planned division into five folds to ensure that the model reaches the best result through many rounds. To validate the model in each round, one-fold is preserved to evaluate the model while the remaining four participated in the data training. The assessment process conforms to a careful protocol:

- Each iteration utilized a unique fold as the validation set
- Performance metrics were calculated independently for each fold
- The aggregate performance was determined through statistical analysis of metrics across all folds
- Mean values of performance indicators were recorded to provide an extensive assessment of model stability and generalization capability

This cross-validation strategy ensures a rigorous evaluation of the model's performance consistency across various data subsets, reducing the impact of data partitioning bias and providing more reliable estimates of the model's generalization capabilities. The implementation of 5-fold cross-validation strikes an optimal balance between computational efficiency and statistical reliability in performance assessment.

4.4 APPLICATION OF FEDERATED LEARNING

In this study, we utilized different FL aggregation methods; the overall design of the FL model is shown in **Figure 9**.

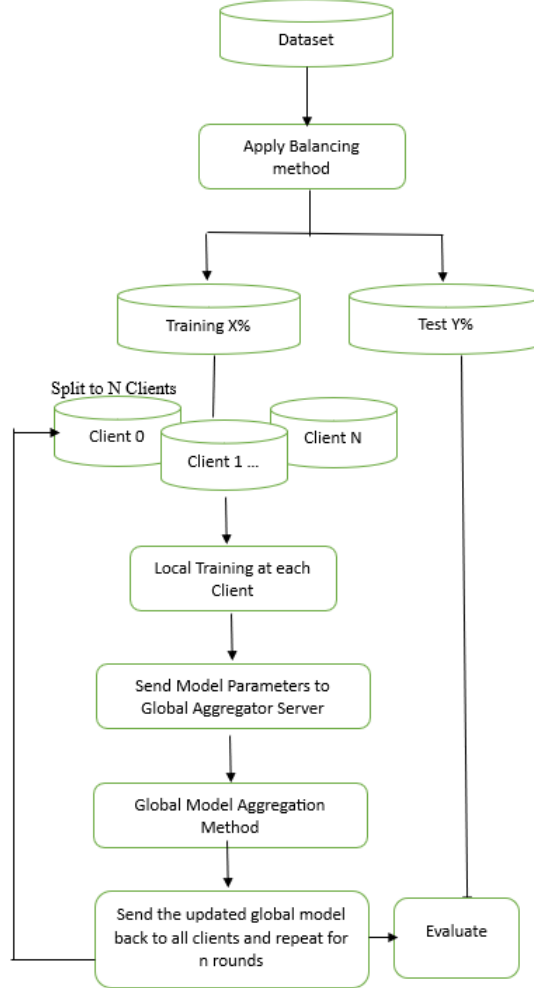


Figure 9: FL Design

4.4.1 An Integration of Federated Learning and Graph Neural Networks for Privacy-preserving Financial Security

4.4.1.1 Overview of the model

The proposed model architecture integrates ML with Federated Learning methodology through a cosine similarity mechanism, establishing a privacy-preserving distributed learning framework. This integration is crucial for banking systems where data confidentiality and cardholder privacy are paramount considerations. The key architectural components function as follows.

The model implements a distributed learning approach where:

- Training processes are executed locally at individual banking nodes
- Model weights, rather than raw transaction data, are shared within the collaborative network
- Cosine similarity metrics guide the weight aggregation process
- Privacy preservation is maintained through the elimination of direct data sharing

This architectural design addresses several critical requirements in financial fraud detection:

- Data Privacy: sensitive transaction information remains secure within local banking environments
- Regulatory Compliance: the framework aligns with data protection regulations and banking security standards
- Collaborative Learning: knowledge sharing occurs without compromising individual institution's data sovereignty
- Model Performance: the system benefits from diverse training data while maintaining strict privacy protocols

The union of federated learning with ML approaches allows financial institutions to maximize the learning of various fraud patterns from each frame while maintaining strong security for cardholders' data safety. This methodology represents a significant advancement in balancing model performance with privacy requirements in sensitive financial applications.

4.4.1.2 Data Preprocessing

The dataset went through a structured approach, which divided the data into various parts to mimic the environment of the federated learning model. After that, a further split is done to ensure that the training is performed on the training set only while the evaluation is applied to unseen datasets for reliable and fair assessment. This split is accomplished to make sure that the model provides a realistic value while achieving the goal of activating a collaborative learning environment. To explain how the partitioning is done to support the FL collaborated model, we present the process as the following points:

- The training dataset was systematically divided into N equal partitions
- Each partition was assigned to a simulated client node
- Individual clients implemented localized data balancing procedures
- Balance methods were applied consistently across all partitions

This hierarchical partitioning strategy serves multiple critical objectives:

- Ensures fair and comprehensive model evaluation through isolated test data
- Facilitates realistic simulation of distributed learning environments
- Enables consistent performance comparison across different architectural configurations
- Maintains data independence while supporting collaborative learning

The method applied to partition the dataset is carefully used to evaluate federated learning model performance to closely see how the model affects overall performance while keeping the test dataset separated from the learning process to observe the effect of this fairly collaborated model on the detection process. Each client trains the balanced dataset locally and sends weights to the federated learning model to participate in the exchange learning environment while keeping test data isolated to compute evaluation metrics without bias.

4.4.1.3 Architecture Design

The suggested federated architecture constructs a distributed learning framework that preserves data privacy while enabling collaborative model improvement. The system operates through a carefully organized interaction between local client nodes and a global aggregation mechanism.

The structural workflow proceeds as follows:

Local Training Phase:

- Individual clients conduct training on balanced local datasets
- Privacy preservation is maintained through strict local data containment
- Model parameters are optimized independently at each client node
- No raw transaction data leaves the local environment

Global Aggregation Phase:

- Client-derived model weights are transmitted to the global model
- Cosine similarity metrics guide the weight aggregation process
- Aggregated parameters are redistributed to client nodes
- This synchronization process iterates through multiple rounds

The federated cycle continues through consecutive rounds of local training and global summation, gradually improving the model's performance while preserving robust data privacy protocol. This recursive process facilitates:

- Collaborative knowledge sharing without data exposure
- Continuous model improvement through distributed learning
- Maintenance of client data sovereignty
- Enhanced model robustness through diverse training exposure

This architectural design effectively balances the requirements for collaborative learning with the stringent privacy demands of financial fraud detection systems.

4.4.1.4 Model Training

4.4.1.4.1 Local Training

The suggested method grants applying changes in the parameters used by the local clients to train their dataset to investigate federated learning models with different model variations. To achieve this, we set some changes in the configuration of ML parameters to mimic real-world scenarios where each client has its features, like adjusting the value for the number of estimators or model depth. In other instances, a uniform model configuration is applied across all clients to establish a controlled experimental baseline. This approach enables a comprehensive evaluation of both heterogeneous and homogeneous federated learning setups.

4.4.1.4.2 Global Training

Cosine similarity is used to evaluate the alignment of each client's model with the global model, as the parameter vectors of the client models are aggregated using weights proportional to their cosine similarity scores. The aggregated parameters are used to update the global model, which is then redistributed to the clients for the next round of training. The process is repeated over and over for several rounds (here, we

use five rounds as the dataset size is relatively small), where in each round, the global model aggregates the updated client models to produce an improved global model.

4.4.2 Federated Averaging with Distance-based Weighted Aggregation

Approach

4.4.2.1 Overview of the Model

The model employs an enhanced variant of traditional Federated Learning, specifically the FedAvg-DWA (Federated Averaging with Distance-based Weighted Aggregation) methodology. This aggregation strategy implements a dynamic weighting mechanism that optimizes the integration of model updates from distributed clients. The FedAvg-DWA approach enhances the conventional federated averaging process by incorporating adaptive weight adjustments based on client contribution quality and reliability metrics. Key architectural components include:

- Distance weight assessment for client contributions
- Adaptive aggregation strategies based on performance metrics
- Optimized convergence through intelligent weight distribution
- Enhanced resilience against client-side heterogeneity

This aggregation methodology offers several advantages over traditional federated averaging:

- Improved model convergence rates
- Enhanced robustness against data heterogeneity
- More efficient utilization of client contributions
- Optimized balance between local and global model performance

Implementing FedAvg-DWA represents a significant advancement in federated learning architectures, particularly for financial fraud detection systems where model performance and training efficiency are crucial considerations.

4.4.2.2 Data Preprocessing

The data preprocessing and partitioning methodology maintains methodological consistency with the previously implemented models. The dataset underwent identical preprocessing protocols and structural partitioning strategies to ensure comparable evaluation frameworks.

4.4.2.3 Architecture Design

The distributed learning architecture implements a systematic n-client partitioning approach, where each client node operates autonomously while contributing to a unified global model. The framework operates through a sophisticated orchestration of local and global learning processes:

Local Training Dynamics:

- Dataset undergoes strategic partitioning across n distributed clients
- Each client implements independent model training procedures
- Local model parameters are optimized within isolated environments
- Weight updates are computed through iterative local training cycles

Global Aggregation Mechanism:

- Global model parameters initialized and distributed to each local client
- A distance-based weighted aggregation methodology orchestrates global model synthesis
- Client contributions are weighted according to their relative significance
- Global model parameters are optimized through intelligent weight integration
- The aggregation process preserves model coherence while leveraging distributed insights
- The server can sense if there is a large distance between the global and local models, which gives insights into whether there is an outlier or bias presented in the training stage.

This architectural framework enables:

- Efficient utilization of distributed computational resources
- Preservation of data privacy through localized training
- Enhanced model robustness through diverse training exposure
- Optimized convergence through intelligent weight aggregation

The integration of distance-based weighted aggregation represents a significant advancement in federated learning architectures, enabling more nuanced and effective model synthesis while maintaining the integrity of local training processes.

4.4.2.4 Model Training

4.4.2.4.1 Local Training

Each client is trained locally, and weights are produced whether all clients use the same parameters or different ones sent to the global model.

4.4.2.4.2 Global Training

The global model needs to be initialized first (with random parameters, or we can use any weights related to one of the Clients, N). Then, the distance between the parameter vectors of the global model and each local model is computed using the Euclidean norm, and weights are derived as the inverse of the distances to ensure that closer models have more influence. This ensures that the used weights dynamically adjust the influence of local models based on their similarity to the global model, which guarantees that models that are too far from the global parameters are assigned to lower weights. The global model repeats the step for n rounds; then, evaluation is done.

4.5 EVALUATION METRICS

In evaluating fraud detection model performance, we employ a comprehensive suite of evaluation metrics that provide deep insights into the model's effectiveness across different aspects of fraud detection [52].

4.5.1 Confusion Matrix

The Confusion Matrix categorizes model predictions into four fundamental outcomes, each carrying specific significance in fraud detection:

True Positives (TP) represent cases where the model correctly identifies fraudulent transactions. In practical terms, these are actual fraud attempts that our system successfully flags, enabling financial institutions to prevent potential losses. Maximizing true positives is crucial as each correctly identified fraud case represents protected customer assets.

True Negatives (TN) occur when legitimate transactions are correctly classified as non-fraudulent. While seemingly straightforward, maintaining a high true negative rate is essential for customer satisfaction and operational efficiency. Each correctly cleared legitimate transaction means a smooth, uninterrupted banking experience for customers.

False Positives (FP), or Type I errors, represent legitimate transactions incorrectly flagged as fraudulent. These misclassifications can lead to significant operational overhead, customer frustration, and unnecessary investigation costs. For instance, a customer's legitimate purchase being declined can result in immediate inconvenience and potentially damage customer trust.

False Negatives (FN), perhaps the most critical metric in fraud detection and known as Type II errors, represent fraudulent transactions that slip through undetected. These misclassifications can result in direct financial losses and compromised security. Each false negative potentially represents a successful fraud attempt that could have been prevented.

4.5.2 Classification Metrics

- **Accuracy** is a fundamental metric for evaluating the performance of classification models in ML. It represents the proportion of correct predictions made by the model across all predictions. We can express this mathematically as:

$$Accuracy = 100 \times \frac{(TP+TN)}{N} \quad (4.2)$$

- **Precision** is a crucial metric that helps us understand how reliable a model's positive predictions are. Think of precision as measuring the model's ability to avoid false alarms. The mathematical formula for precision is:

$$Precision = \frac{TP}{(TP+FP)} \quad (4.3)$$

- **Recall**, also known as the True Positive Rate (TPR) or sensitivity, measures how well a model can identify actual positive cases from all the existing positive cases. In the context of fraud detection, it tells us what percentage of all actual fraudulent transactions our model successfully catches. We can express this mathematically as:

$$Recall = \frac{TP}{(TP+FN)} \quad (4.4)$$

- **The False Positive Rate (FPR)**, also known as the fall-out rate, measures the proportion of actual negative cases (legitimate transactions) that are incorrectly classified as positive (fraudulent). We can express this mathematically as:

$$FPR = \frac{FP}{(FP+TN)} \quad (4.5)$$

- **F1-Score** is a vital metric that helps us evaluate classification models by combining precision and recall into a balanced measure. The F1-Score is the harmonic mean of precision and recall, expressed mathematically as:

$$F1 - Score = 2 \times \left(\frac{(Precision \times Recall)}{(Precision + Recall)} \right) \quad (4.6)$$

4.5.3 Visual Representations

The Area Under the Curve (AUC), also known as AUC-ROC (Area Under the Receiver Operating Characteristic Curve), is a powerful metric for evaluating classification model performance across all possible thresholds. The ROC curve plots the True Positive Rate (TPR) against the False Positive Rate (FPR) as we vary the classification threshold from 0 to 1. The AUC represents the entire two-dimensional area underneath this ROC curve. A perfect model would have an AUC of 1.0, meaning it achieves 100% TPR (catching all fraud) with 0% FPR (no false alarms). In contrast, a model that guesses randomly would have an AUC of 0.5, creating a diagonal line on the plot.

4.5.4 Statistical Metrics

Matthews Correlation Coefficient (MCC) is a comprehensive metric that provides a balanced measure of classification performance even when classes are imbalanced [10]. MCC generates a correlation coefficient between the observed and predicted binary classifications, producing a value between -1 and +1. The mathematical formula for MCC is:

$$MCC = \frac{(TP \times TN) - (FP \times FN)}{\sqrt{(TP+FP)(TP+FN)(TN+FP)(TN+FN)}} \quad (4.7)$$

4.6 POST-PROCESSING

4.6.1 Adding and Detecting Noise in FL Systems

4.6.1.1 Overview of the Model

We mentioned the idea of Federated Learning and their collaborative environment, especially in a fraud detection system where each bank trains the data locally and sends only weights or updates to a global model; this keeps the privacy of the dataset, which is crucial in banking system and getting the advantage of improving fraud detection model using others bank knowledge, but this imposes a challenge that some of the clients (banks) that participated in the learning model may try to sabotage the process by manipulating the model by sending poisoning updates (noisy bank to contaminate the model) to the global model which when adding updates from all clients including malicious one create an inadequate global model. This might cause fraudulent transactions not to be detected and legitimate transactions to be considered fraud due to the wrong updates sent by the global model to all clients. Different solutions are introduced to mitigate this and enhance the global model against malicious model updates [42].

4.6.1.2 Architecture Design

In this model, we incorporate a noise (malicious) bank into the system by contaminating one of the participating clients through label flipping of the target feature. To address this, we employ a method for noise detection inspired by FLDetector [44].

In [44], the researchers adapted to identify noise at varying levels by introducing label flipping on the target feature. The detection mechanism operates using a predefined threshold, where noise levels below the threshold remain undetected, while noise levels at or above the threshold trigger detection. This approach identifies malicious clients within the federated learning (FL) environment. FLDetector detects malicious clients by evaluating model update consistency. It leverages historical model updates and computes deviation metrics to assign scores to client updates. Once a malicious act is detected, the method removes the suspected

client or clients and resumes the process [44]. Since noise influences client updates, it causes deviations from the expected norm. We adopt this idea by applying a label flip attack on the target feature that done randomly with various noise percentages to investigate the effect of malicious act on the performance metrics of the FL model. The model works normally with certain noise levels and complete all rounds while with specific noise level (25% or higher) triggers a detection method to identify the noisy client, as a result of this trigger an alert message showed on output terminal specifies the corrupted client number, and the model stops the learning process immediately. Without noise detection, the FL process runs for all specified rounds under normal conditions. Following the final round, evaluation metrics are computed using the last round's updates. However, in cases where noise is detected (at or above the threshold), only one round is completed before the model stops, and evaluation is performed using the model state at that point. This addition of noise let us see the effect of noise on the performance of the global model and how the metrics got effected. Additionally, detection mechanism integrated within the global server to make sure that the model can detect noise at certain threshold and stop the learning process to construct a robust FL model that takes into consideration malicious act by one of the participated banks. This approach ensures prompt identification and isolation of malicious clients and enhances the integrity of the federated learning process.

CHAPTER V

EXPERIMENTS

The experimental structure was well defined based on a careful set of performance evaluation metrics like accuracy, precision, recall, F1-score, Area Under Curve (AUC), and Matthews Correlation Coefficient (MCC) to ensure that the validation cover the overall aspect of the results. The use of various metrics facilitates reliable analysis across different perspectives. The experimental examination involves different crucial criteria. Model efficiency is tested under different conditions to cover the overall aspect of the evaluation process. Using various metrics, we can assess the model performance from many points of view and not focus on one aspect. The framework consists of necessary steps to replicate the model by future researchers or any further enhanced work, such as hardware environment used, dataset characteristics, parameter configuration, and design of the models. The results were analyzed carefully for each constructed model to recognize performance metrics, which model performs the best, and possible restrictions. Extensive research empowers a deep understanding of each model's powerful points as well as its limitations, which in turn provides an essential understanding of each model's potential in fraud detection models. The experimental analysis considers each model's computational requirements and functional productivity, presenting a fair comparison of each model's efficiency. Through this structured evaluation framework, we clearly understood each model's performance characteristics, enabling informed decisions about their deployment in real-world fraud detection scenarios. The results offered a necessary inference about each approach 's relative strengths and possible restrictions, supporting evidence-based selection of appropriate models for specific operational contexts.

5.1 HARDWARE

The experiments were conducted on a laptop using OS Windows 11 Pro with the 64-bit operating system and 16.0 GB RAM. *Jupyter Notebook* is used to run code on the local environment for the Iraqi bank dataset. In addition, *Google Colab Pro* edition was used [53] to run codes for the Kaggle *creditcard* dataset. Colab specification high-RAM of 51.0 GB and L4 GPU processor used to run all experiments tests.

5.2 EXPERIMENTAL ANALYSIS OF FRAUD DETECTION IN IRAQI BANKING TRANSACTION

We perform many experiments using models explained in the methodology part, and results are realized using Metrics to evaluate each model, once with the Original Dataset and the other with the public Kaggle dataset.

5.2.1 ANN Model

The model was evaluated with the most common metrics for fraud detection, and the evaluation was done on the Test dataset set (unseen). The model shows good results, especially recall with relatively missed legitimate transactions. Results are shown in **Table 3**, and the prediction in Confusion Matrix is shown in **Figure 10**.

Table 3: ANN Model Metrics

ANN model					
Accuracy	Precision	Recall	F1 Score	AUC Score	MCC
0.9567	0.9452	0.9697	0.9573	0.9834	0.9138

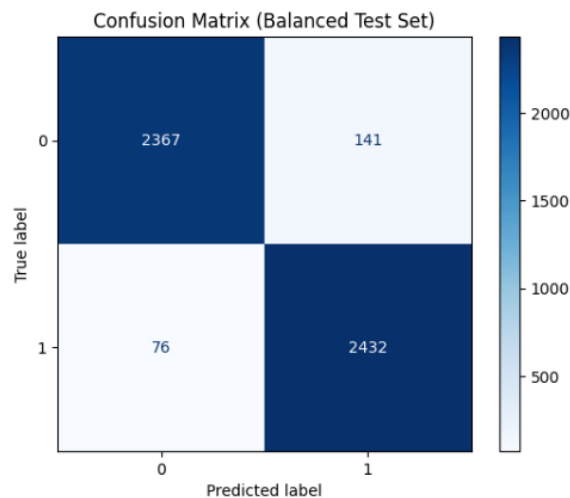


Figure 10: Confusion Matrix of ANN Model

5.2.2 LSTM Model

The model evaluation was conducted on Test data, and the result is shown in **Table 4** below. Also, the Confusion matrix is shown in **Figure 11**. We can see that from a Recall perspective, the LSTM model works slightly better; this metric is critical in fraud detection systems as it catches the true positive (fraud).

Table 4: LSTM Model Metrics

LSTM Model					
Accuracy	Precision	Recall	F1 Score	AUC Score	MCC
0.9522	0.9362	0.9705	0.9530	0.9813	0.9049

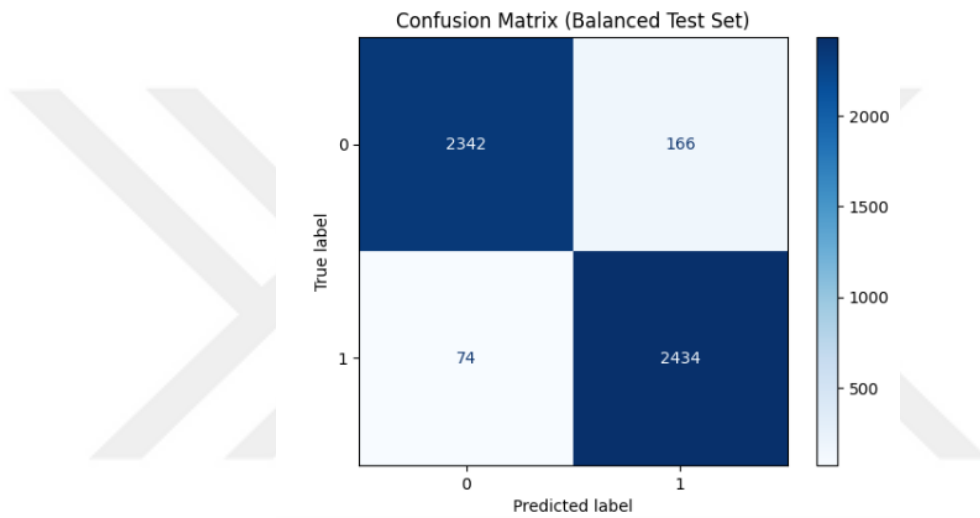


Figure 11: Confusion Matrix of LSTM Model

5.2.3 RF+AdaBoost Model

The model evaluation was conducted on Test data and compared with standalone SMOTE+RF model along with the pipelined model as shown in **Table 5** below. We ensure that the standalone model uses the same parameters as the RF-enhanced model for a fair comparison. Confusion matrix is shown in **Figure 12**. We can see that the enhanced model achieves superior performance compared to other models.

Table 5: Comparison Results between RF and RF+AdaBoost

Metrics	SMOTE+RF	SMOTE+RF+AdaBoost
Accuracy	0.9516	0.9918
Precision	0.9252	0.9865
Recall	0.9815	0.9972
F1 Score	0.9525	0.9918
AUC Score	0.9769	0.9997
MCC	0.9048	0.9837

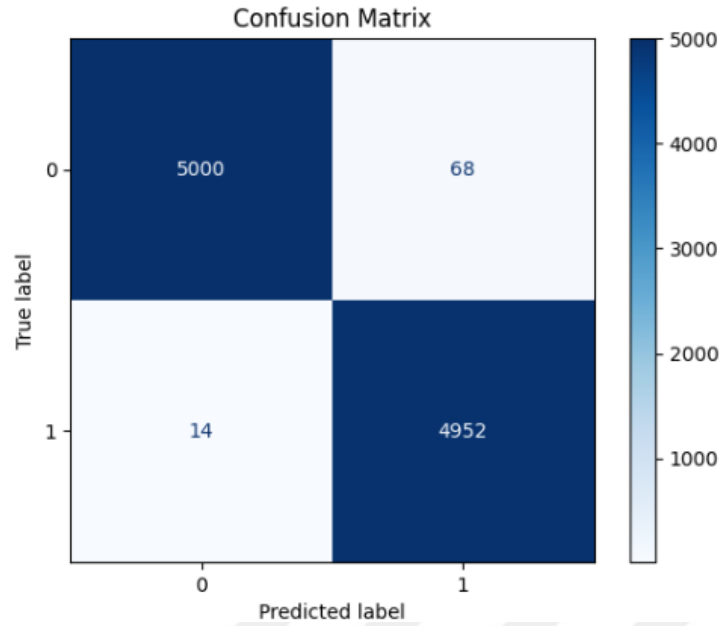


Figure 12: Confusion Matrix of SMOTE+RF+AdaBoost Model

5.2.4 Summary of Model Performance

The evaluation of the three models, ANN, LSTM, and RF + AdaBoost, clearly indicates that the RF + AdaBoost model outperforms the others. Also, the model achieves higher performance across other metrics, including accuracy, recall, precision, and AUC, with its MCC score being notably higher than the others, as shown in **Figure 13**. This demonstrates the model's robustness in handling the class imbalance problem. Based on these results, the RF + AdaBoost model is determined to be the best-performing model in this study.

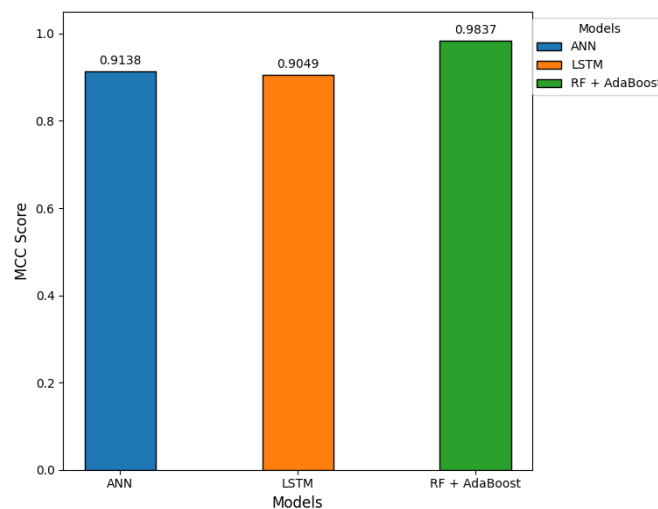


Figure 13: Comparison of MCC between Various ML Models

5.2.5 Graph-based Neural Network with FL Cosine Model

The split of the dataset followed a structural approach to dividing the repressed dataset into 90% training and 10% testing. The training dataset is divided into three even splits to mimic the FL environment. Each client trains their portion of the dataset locally using a graph conventional neural network that converts transactions into nodes and connects nodes through edges to capture connections between nodes. The configuration uses hyperparameter tuning (Grid Search) to select an optimal number of neurons in the hidden layer and learning rate. The Graph Conventional Network (GCN) was constructed using PyTorch Geometric library. The local banks send their weights to the global model. The model runs for five rounds, and in the last one, the parameters produced by the global model are used to evaluate the reserved balanced test dataset to void bias prediction. The evaluation was made using a set of metrics as explained in **Table 6**. Results showed that the model leaned toward class zero rather than class one.

Table 6: Results of Graph-based FL Cosine-based Model

Metrics	Graph-based GCN FL
Accuracy	0.8621
Precision	0.9060
Recall	0.8081
F1 Score	0.8543
AUC Score	0.8621

5.2.6 SMOTE+RF+AdaBoost with FL Cosine Model

The dataset is split into 80% training and 20% test sets. The training dataset is further split evenly between three banks. Each client applied RF+AdaBoost to train data locally using different hyperparameters to mimic real-world applications and weights sent to the global model. After completing all five rounds, the last parameters produced by the global model are used to evaluate the balanced test dataset for a fair comparison with the standalone model. We can notice that metrics are lower with FL trade-off performance against security and privacy preservation. The comparison of the metrics is explained in **Table 7**. Confusion Matrix for the FL model is shown in **Table 8**.

Table 7: Comparison results between RF+AdaBoost and RF+AdaBoost+FL

Metrics	SMOTE+RF+AdaBoost	SMOTE+RF+AdaBoost with FL
Accuracy	0.9918	0.9647
Precision	0.9865	0.9830
Recall	0.9972	0.9458
F1 Score	0.9918	0.9640
AUC Score	0.9997	0.9963
MCC	0.9837	0.9301

Table 8: Confusion Matrix of RF+AdaBoost with FL Cosine-based

		Predicted Labels	
		Legit	Fraud
Actual Labels	Legit	4934	82
	Fraud	272	4744

5.2.7 SMOTE+RF+AdaBoost with FL Model

FedAvg-DWA balances between contributions from multiple clients while assigning a small weight for farther clients' distance to remove bias, so the farther the distance, the less influence on the global model. The same division method is applied, evenly assigning training data between three clients. Each client is trained locally with exact parameters and sent to the global model. We use five rounds in this model, and in the last round, parameters were used to evaluate a balanced test dataset (20%) for a fair comparison with the standalone model, as explained in **Table 9**. Confusion Matrix for the FL FedAvg-DWA model is shown in **Table 10**.

Table 9: Comparison Results between RF+AdaBoost and RF+AdaBoost+FL DWA

Metrics	SMOTE+RF+AdaBoost	SMOTE+RF+AdaBoost with FedAvg-DWA
Accuracy	0.9918	0.9637
Precision	0.9865	0.9868
Recall	0.9972	0.9400
F1 Score	0.9918	0.9628
AUC Score	0.9997	0.9970
MCC	0.9837	0.9285

Table 10: Confusion Matrix of RF+AdaBoost with FL DWA

		Predicted Labels	
		Legit	Fraud
Actual Labels	Legit	4953	63
	Fraud	301	4715

5.2.8 Execution Time Analysis

To further differentiate the performance of FL with Cosine and FedAvg-DWA, we measured the execution time of both models. The execution time reflects the total time to complete one training run for each method, including model updates and aggregation. The run time was measured over five repeated runs to minimize the

influence of system fluctuations. For FL with Cosine, the execution time was (620.01) seconds, while for FL with FedAvg-DWA was (1064.13) seconds, as shown in **Figure 14**.

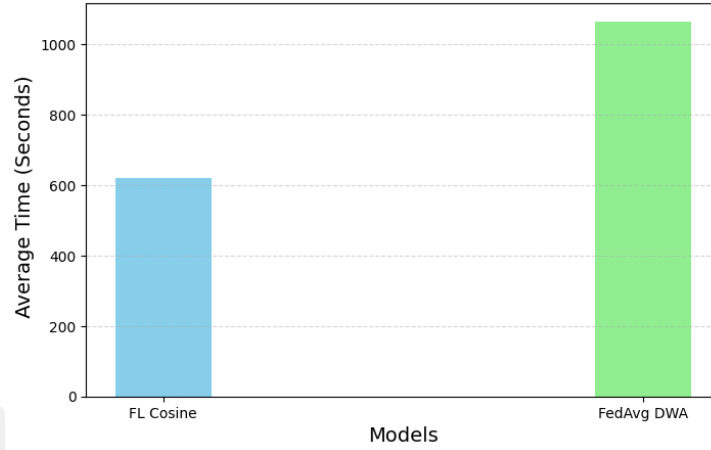


Figure 14: Time Analysis of Both Aggregation Methods

These results show that the faster model is FL with Cosine, with a time difference of 444.12 seconds. This analysis highlights the model's effectiveness in terms of better performance metrics (accuracy, precision, recall, and MCC) than FL DWA and its computational efficiency.

5.2.9 Using ANN Model with FL Cosine Similarity

We divide data into three clients and apply deep neural network learning for local model training. The result of the ANN+FL model shows a significant improvement in Recall value when FL with cosine is applied compared to the standalone model, as shown in **Table 11**, which emphasizes the importance of using Neural Networks along with the FL model between banks to empower fraud detection systems and enhance it using knowledge from other banks without breaking data sensitivity.

Table 11: Comparison Results between ANN and ANN+ FL Cosine-based

Metrics	ANN Model	ANN Model with FL Cosine
Accuracy	0.9567	0.9299
Precision	0.9452	0.8859
Recall	0.9697	0.9869
F1 Score	0.9573	0.9337
AUC Score	0.9834	0.9299
MCC	0.9138	0.8654

5.2.10 Adding and Detecting Noise in FL with Cosine System

We use the best model RF+AdaBoost along with FL cosine as the optimal aggregation method according to our time analysis between two aggregation methods to detect and test the performance at various noise levels, as shown in **Table 12**.

The experiment uses a label flip attached to the target feature with different noise level percentages (10, 15, 20, 25). The experiment randomly applies label flip of target feature to flip fraud into legit and vice versa at different percentages. The FL model should be able to detect noisy clients when the percentage level is equal to or bigger than 25% and triggers an output message showing the number of malicious clients. Also, we immediately stopped the learning process rounds and calculated the metrics. The results show that the performance is getting lower with each noise level. There is something exciting about the noise level of 10%; we see that recall is higher compared to the result with a clean dataset, which is misleading and imposes the importance of implementing noise detection and handling methods within FL systems, especially since the recall factor is critical in fraud detection in banking systems as it catches true positive. We can observe that the higher the noise level the more negatively affect the performance of the model.

Table 12: Evaluation Metrics at Different Noise Levels

Metrics	Noise Levels			
	10%	15%	20%	25%
Accuracy	0.9046	0.8703	0.8334	0.8334
Precision	0.8561	0.8250	0.8307	0.8307
Recall	0.9727	0.9400	0.8375	0.8375
F1 Score	0.9107	0.8788	0.8341	0.8341
AUC Score	0.9748	0.9552	0.9282	0.9282
MCC	0.8168	0.7479	0.6669	0.6669

5.2.11 Adding and Detecting Noise in FL with FedAVG-DWA System

Another experiment evaluated the impact of adding different noise levels while employing FLDetector with the FedAvg-DWA method and RF+AdaBoost model. However, the experiment failed to detect noise effectively due to several key factors. Firstly, FedAvg-DWA assigns smaller weights to clients with more considerable distances from the global model, which reduces the influence of these clients during the aggregation process. This approach is intended to minimize the impact of outliers. Still, in the context of noisy updates, it limits the model's capacity to identify noise as the more considerable distance from the global model is handled by assigning smaller

weights. Secondly, the distance-based weighted aggregation (DWA) mechanism assigns weights based on the distance between the client's local and global models. Since noisy clients produce updates with similar magnitudes but different directions than legitimate clients, the algorithm struggles to differentiate between noise and legitimate client updates. Consequently, this limits the model's ability to detect and identify noisy clients.

5.3 EXPERIMENTAL ANALYSIS OF FRAUD DETECTION IN KAGGLE DATASET

5.3.1 Using ANN Model

We test the ANN proposed model with the same dataset preprocessing and architecture design but this time with the public *creditcard* dataset. Results using the public dataset are shown in **Table 13**. We can see that the model leans toward the legit class more than the fraud due to the high imbalance of *creditcard* class, so the class imbalanced needs to be addressed with more robust methods, also the design plays an essential role in the overall performance (number of layers, hidden layers and number of neurons). We only change the number of neurons in each layer according to the proposed model and evaluate the performance of the model again, and right away, the metrics improved, as shown in **Table 14**.

Table 13: Results of ANN Model

Metrics	ANN model Kaggle
Accuracy	0.9498
Precision	0.9990
Recall	0.9005
F1 Score	0.9472
AUC Score	0.9789
MCC	0.9040

Table 14: Results of ANN Model using Exact Model Design

ANN model with exact design proposed by [30]					
Accuracy	Precision	Recall	F1 Score	AUC Score	MCC
0.9579	0.9985	0.9171	0.9561	0.9752	0.9189

5.3.2 Using the LSTM Model

We tested the LSTM proposed model with the same dataset preprocessing and architecture design but this time with the public *creditcard* dataset. Results are shown in **Table 15**. We can see that the model leans toward the non-fraud class more than fraud due to the high imbalance of *creditcard* class. For that, class imbalance needs to

be addressed with more powerful methods; also, the size and features of the Kaggle dataset should be taken into consideration.

Table 15: Results of the LSTM Model

Metrics	LSTM Model Kaggle
Accuracy	0.9453
Precision	0.9971
Recall	0.8931
F1 Score	0.9423
AUC Score	0.9705
MCC	0.8554

5.3.3 Using SMOTE+ RF+AdaBoost Model

The ensemble learning SMOTE+RF+AdaBoost approach was applied with same design followed with *PB_dataset* to observe the effect. The evaluation done on test dataset is shown in **Table 16**. showing very good overall performance.

Table 16: Results of RF+AdaBoost Model

Metrics	RF+AdaBoost Kaggle DS
Accuracy	0.9907
Precision	0.9938
Recall	0.9876
F1 Score	0.9907
AUC Score	0.9997
MCC	0.9814

5.3.4 Using Graph-based Neural Network with FL Cosine Model

The framework combines GNN with FL cosine-based aggregation. We applied the same design that was followed with *PB_dataset* to identify the effect while using the Kaggle dataset. The dataset was split into 10 clients to adhere to the FL environment, and each client locally trained their portion using GCN that consists of an input, hidden layer, and output layer. The process repeated for ten rounds. Performance metrics were conducted on the last round on the reserved test dataset; results are shown in **Table 17**. We can observe from the results that an advanced balancing technique needs to be applied. Also, more sophisticated feature importance approaches need to be considered in the design of graph-based design.

Table 17: Results of Graph-based FL Cosine-based Model

Metrics	Graph-based FL Cosine
Accuracy	0.8642
Precision	0.9983
Recall	0.7296
F1 Score	0.8431
AUC Score	0.8642

5.3.5 Using SMOTE+ RF+AdaBoost+FL Cosine Model

We divide training data into ten even parts and then experiment with RF+AdaBoost along with FL “cosine similarity,” where each client trains data locally using the ensemble method RF+Adaboost and sends weights to the global model for summing using cosine-similarity-based aggregation. The process was repeated for 10 rounds, and the last round weights were used to assess the model on an unseen dataset. Evaluation metrics are shown in **Table 18**.

Table 18: Results of RF+AdaBoost+FL Cosine-based Model

Metrics	RF+AdaBoost+FL Kaggle DS
Accuracy	0.9380
Precision	0.9997
Recall	0.8762
F1 Score	0.9339
AUC Score	0.9901
MCC	0.8827

5.3.6 Using ANN with FL Cosine Model

In this experiment, we applied the Kaggle *creditcard* dataset with the proposed ANN and the FL cosine similarity method for a collaborative environment. The model design and steps used are the same as with the original dataset. Only clients split into 10 banks, and FL rounds are set to 10 due to size differences. The results are presented in **Table 19**. We can see that the model works very well in not detecting legit transactions as fraud as the precision value is almost perfect but misses fraud transactions presented by recall value. The result is also affected by the design of the ANN model, the size of the dataset, and how many features and fraud patterns are inhabited. The Kaggle dataset is larger with different feature numbers; the highly imbalanced ratio challenge needs to be addressed carefully.

Table 19: Results of ANN with FL Cosine-based Model

Metrics	ANN+FL Kaggle DS
Accuracy	0.9211
Precision	0.9940
Recall	0.8473
F1 Score	0.9148
AUC Score	0.9211
MCC	0.8515

5.3.7 Using FL Cosine and Noise Detection

We use the Kaggle public dataset with RF+AdaBoost and FL cosine similarity enhanced by applying the noise detection method. The same experiment is used, which is applying corruption to one client by flipping the label target field at different level percentages (10, 15, 20, 25); with 10,15 and 20 percent the model would not detect noise while with 25%, a threshold is set to detect noisy client numbers by showing a message displayed on the output terminal and the code breaks immediately then calculates metrics. The evaluation result at each level is shown in **Table 20**. Results showed that noise negatively affected performance, especially on MCC.

Table 20: Evaluation Metrics at Different Noise Levels

Metrics	Noise Levels			
	10%	15%	20%	25%
Accuracy	0.9581	0.9487	0.9488	0.9456
Precision	0.9997	0.9996	0.9996	0.9998
Recall	0.9165	0.8979	0.8981	0.8914
F1 Score	0.9563	0.9460	0.9461	0.9425
AUC Score	0.9909	0.9934	0.9934	0.9924
MCC	0.9194	0.9022	0.9024	0.8965

CHAPTER VI

DISCUSSION AND CONCLUSION

A comparative analysis was conducted to evaluate the performance of multiple ML architectures across two distinct datasets: our *PB_dataset* and the larger public Kaggle benchmark dataset. The empirical results, visualized in **Figure 15**, demonstrate that the RF+AdaBoost ensemble configuration consistently achieved superior accuracy metrics across both datasets. This cross-dataset performance validation provided robust evidence for the model's robustness and generalization capabilities, independent of dataset scale and feature characteristics.

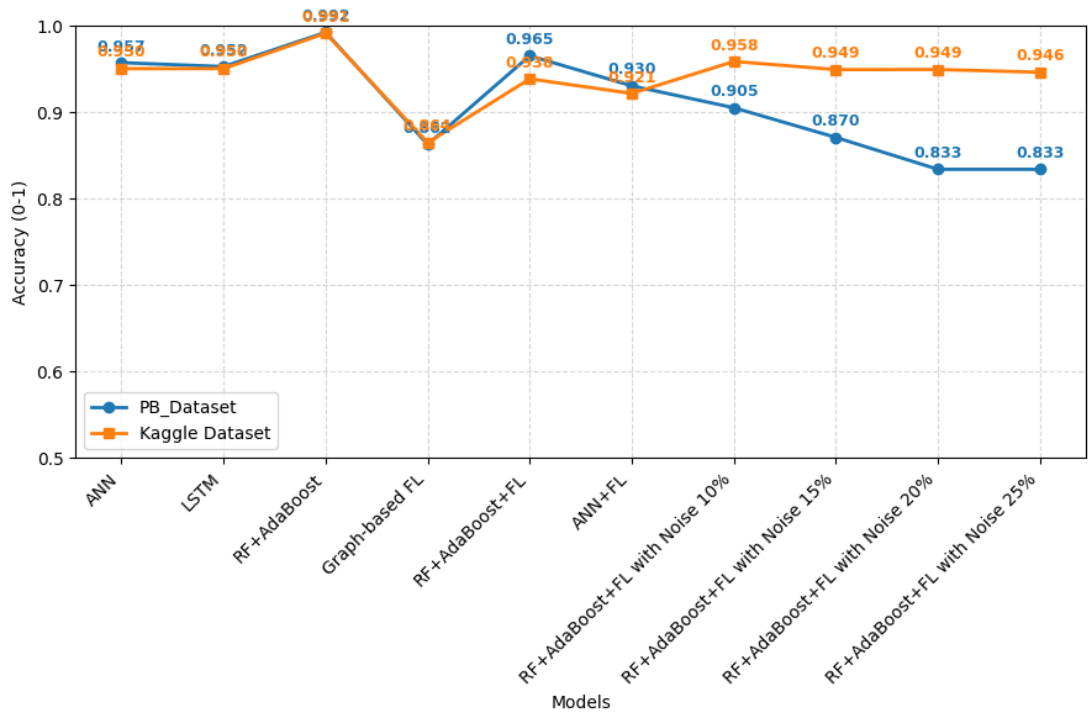


Figure 15: Models Comparison between PB_dataset and Kaggle Against *Accuracy*

Analysis of the results reveals that the RF+AdaBoost model, when integrated with the federated learning framework, exhibits higher accuracy metrics on our *PB_dataset* than those achieved on a larger Kaggle dataset. This finding challenges the conventional assumption that federated learning frameworks primarily benefit large-

scale datasets, Alternatively, it suggests that the architecture can effectively enhance performance across varying dataset scales. Given the critical importance of recall metrics in CCFD, a comprehensive comparison of model performance across both datasets concerning recall is presented in **Figure 16**, showing that RF+Adaboost integrated with FL accomplished the best recall compared to other tested models.

Furthermore, the visualization in both figures demonstrates that noise disorder has a more pronounced impact on smaller datasets, indicating an inverse relationship between dataset size and noise sensitivity. The analysis further reveals that while noise injection reduces model performance, larger datasets are more resistant to such disturbance.

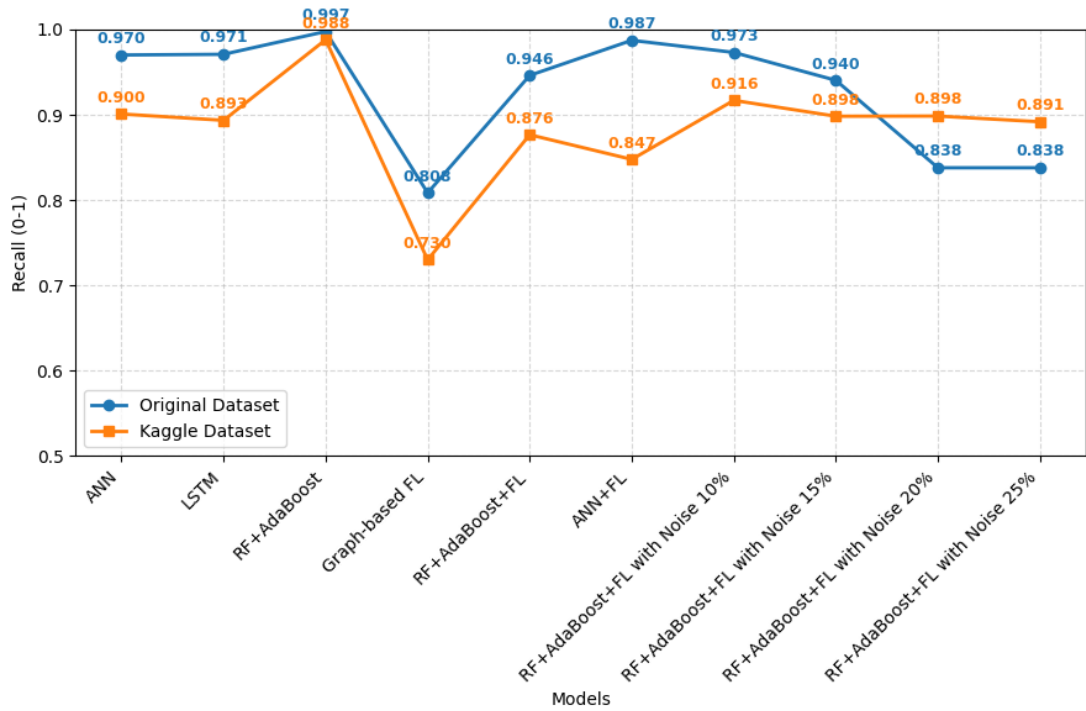


Figure 16: Models Comparison between PB_dataset and Kaggle Against *Recall*

The RF+AdaBoost ensemble model demonstrated consistently superior performance across both datasets, substantiating its robustness and computational efficiency. DNN architecture exhibited enhanced performance metrics on the *PB_dataset*, potentially attributable to factors, including the pronounced class imbalance ratio and distinct feature space characteristics. A particularly substantial finding is the improved recall achieved by the ANN when implemented within the federated learning framework compared to the equivalent standalone model on the

PB_dataset. This result suggests that FL ensures data privacy and enhances the detection of true positive cases.

On the other hand, Graph-based FL performed almost identically for accuracy metric in both datasets (*PB_dataset*: 86.21%, Kaggle: 86.42%). However, recall achieved a higher value on *PB_dataset* (80.81%) than on the Kaggle dataset (72.96%). Moreover, the results showed average performance compared to other ML models with FL, indicating that GCN may not be suitable for a small dataset with limited of features. Even with a larger dataset (Kaggle), the results showed moderate performance, suggesting a need for advanced methods to enhance the model, such as feature importance.

The observed results proved the efficiency and reliability of ensemble learning through various metrics conducted. The RF+AdaBoost model achieved very good results in both datasets despite their different sizes and features, ensuring the generalization capability of this model regardless of the dataset size. This favorable result reveals the capabilities of ensemble learning techniques to handle the complexity of different financial datasets. In addition, the combination of federated learning models holds huge opportunities and enhancements for the fraud detection domain due to its ability to pass information from various banks to each other without breaking regulatory rules or jeopardizing the privacy of the datasets. The merge between the DNN model and federated learning, particularly the ANN model with federated learning, showed notably better recall values with collaborated information compared to the recall of the standalone model on the *PB_dataset*. This observation is crucial as banks always seek to catch fraudulent transactions as much as possible, and missing true positives can lead to serious problems for the bank.

The empirical results suggest that deep learning architectures implemented within a federated framework may be optimally suited for high-stakes financial environments where comprehensive fraud detection is paramount for loss prevention. This finding has critical implications for banking institutions where the cost of missed fraudulent transactions significantly outweighs the impact of false positives.

One of the experiments applied involved implementing a label flip attack with various noise levels on two FL aggregation methods: FL cosine similarity and FL DWA using FLDetector. The trial failed to detect noise with FL DWA-based aggregation because its internal structure inherently handles outliers. We performed a comparison

between the results generated by our model on both datasets and literature papers inspired our methodology. We found out that:

- Application of ensemble learning (RF+AdaBoost): The results of the model using the *PB_dataset* reached 99.18% accuracy and 99.72 recall. The results of the Kaggle *creditcard* dataset were 99.07% accuracy and 98.76% recall, showing minor differences compared to the existing literature [34], which achieved 99.95% accuracy and 99.86% recall on the Kaggle dataset. This proves the efficiency and reliability of this model, as explained above.
- Application of ANN design: results on the *PB_dataset* reached 95.67% accuracy and 96.97% recall, while on the Kaggle dataset, results were 94.98% accuracy and 90.05% recall. These results were lower results compared to the literature paper [30], which reported 98.54% accuracy and 97.96% recall. The better results that appeared in the existing literature are potentially attributed to many factors like balancing methods or feature importance techniques.
- Application of Graph-based with FL cosine similarity aggregation: results were 86.21% accuracy and 80.81% recall on the *PB_dataset*, while with the Kaggle dataset, they were 86.42% accuracy and 72.96% recall. The results were lower than the existing literature, which achieved 95.99% accuracy and 91.10% recall. The relatively weak performance suggests the for more advanced balancing techniques and integration of feature importance methods to enhance GNN performance. Furthermore, complex GNN design may not be suitable for small datasets with limited features.
- Application RF+AdaBoost with FL cosine-based aggregation: our model achieved 96.47% accuracy and 94.58% recall on the *PB_dataset*, outperforming the FL graph-based method on the Kaggle dataset [41], which reached 95.99% accuracy and 91.10% recall. However, results obtained by RF+AdaBoost and FL on the Kaggle dataset were lower than [41], with 94.58% accuracy and 87.62% recall. These results

highlight the potential of ensemble learning integrated within the FL model, particularly in real-world scenarios.

The summary of the comparison with existing literature is illustrated in **Table 21**.

Table 21: Performance Comparison with Existing Literatures

Model	Dataset	Accuracy (%)	Recall (%)	Existing Literature	Notes
Ensemble Learning (RF+AdaBoos)	PB_dataset	99.18	99.72	[31] 99.95% accuracy 99.86% recall	Strong performance
	Kaggle	99.07	98.76		
ANN DNN	PB_dataset	95.67	96.97	[27] 98.54% accuracy 97.96% recall	Efficiency of DNN design
	Kaggle	94.98	90.05		
Graph-Based FL with Cosine Similarity Aggregation	PB_dataset	86.21	80.81	[38] 95.99% accuracy 91.10% recall	Poor results need complex methods
	Kaggle	86.42	72.96		
RF+AdaBoost FL Cosine Aggregation	PB_dataset	96.47	94.58	[38] 95.99% accuracy 91.10% recall	Potentials of Ensemble learning
	Kaggle	93.80	87.62		

In conclusion, insights from comparisons between our *PB_dataset* and existing literature demonstrate the efficiency and reliability of the RF+AdaBoost model for fraud forecasting across various datasets. Furthermore, this study reveals the model's flexibility to adopt a federated learning collaborative approach. Despite the differences between datasets, which restrict straightforward comparisons between metrics, our project successfully provided valuable insights into integrating ensemble ML and FL to tackle real-world fraud detection problems. Given the lack of such datasets in the existing literature, the use of a real bank dataset significantly influenced the research domain.

CHAPTER VII

FUTURE WORK

Our suggestion for further work is to extend the experiment and collaborate with different local Iraqi banks to build an FL environment with non-IID to share knowledge of different fraud patterns. Moreover, we should use various attack types to make sure that the FL model is secure enough. Also, incorporating methods suitable for Iraqi society to raise awareness about fraud and how to improve cardholders' ability to protect their credentials as the client represents the first line of defense against fraud is crucial. Banks need to pay attention not only to using sophisticated tools for forecasting but also focusing on prevention by taking advantage of many existing terminals like social media to raise awareness through various channels as proposed in the paper [54].

REFERENCES

- [1] WORLD BANK (2025), International Bank for Reconstruction and Development, *A world Bank Group Flagship Report: Global Economic Prospects January 2025*, Washington, DC.
- [2] ALI Abdulalem, ABD RAZAK Shukor, OTHMAN Siti Hajar, EISA Taiseer Abdalla Elfadil, AL-DHAQM Arafat, NASSER Maged, ELHASSAN Tusneem, ELSHAFIE Hashim and SAIF Abdu (2022), “Financial Fraud Detection Based on Machine Learning: A Systematic Literature Review”, *Applied Sciences*, Vol. 12, No 9637, pp. 2–24, DOI: 10.3390/app12199637.
- [3] GANDHAR Akash, GUPTA Kapil, PANDEY Aman Kumar and RAJ Dharm (2024), "Fraud Detection Using Machine Learning and Deep Learning", *SN Computer Science*, Vol. 5, No 453, pp. 2–10, DOI: 10.1007/s42979-024-02772-x.
- [4] GDPR (2024), GENERAL DATA PROTECTION REGULATION, *Legal Text*, <https://gdpr-info.eu/>, DoA. 15.01.2024.
- [5] GHAI V. and KANG S. S. (2021), “Role of Machine Learning in Credit Card Fraud Detection”, In, *Proceedings of the 2021 3rd International Conference on Advances in Computing, Communication, Control, and Networking (ICAC3N 2021)*, pp. 939–943, IEEE, United States, DOI: 10.1109/ICAC3N53548.2021.9725540.
- [6] JHA B. K., SIVASANKARI G. G. and VENUGOPAL K. R. (2020), “Fraud Detection and Prevention by using Big Data Analytics”, In, *Proceedings of the 4th International Conference on Computing Methodologies and Communication, ICCMC 2020*, pp. 267–274, IEEE, United States, DOI: 10.1109/ICCMC48092.2020.ICCMC-00050.
- [7] ASMAR M. and AQEL B. Y. (2023), “Analysis of Credit Cards Fraud Detection: Process and Techniques Perspective”, In, *Artificial Intelligence (AI) and Finance*, Eds. Alareeni B.A.M., Elgedawy I., pp. 899–911, Springer, Cham, Switzerland, DOI: 10.1007/978-3-031-39158-3_84.

- [8] VYAWAHARE H. R. (2022), “Machine Learning: A Solution Approach for Complex Problems”, *International Journal of Scientific Research in Engineering and Management*, Vol. 6, No 04, pp. 1-3, DOI: 10.55041/ijjsrem16123.
- [9] KAUR C., PAL SINGH BRAR T., CHANDEL R. and SHARMA S. (2023), *Machine Learning and its Applications-A Review Study*, www.cgcijectr.com, DoA. 15.01.2024.
- [10] NAEEM S., ALI A., ANAM S. and AHMED M. M. (2023), “An Unsupervised Machine Learning Algorithms: Comprehensive Review”, *International Journal of Computing and Digital Systems*, Vol. 13, No 1, pp. 911–921, DOI: 10.12785/ijcds/130172.
- [11] DHAMELIA H. and MORADIYA R. (2023), “Unlocking Hidden Insights: Unleashing the Strength of Semi-Supervised Learning in Machine Learning”, *International Journal for Research in Applied Science & Engineering Technology (IJRASET)*, Vol. 11, No 8, pp. 2049–2057, DOI: 10.22214/ijraset.2023.55468.
- [12] BHARATHI B., SHAREEFA P., UMA MAHESHWARI P., LAHARI B., DAVID DONALD A. and ADITYA SAI SRINIVAS T. (2023), “Exploring the Possibilities: Reinforcement Learning and AI Innovation”, *International Journal of Advanced Research in Science, Communication and Technology*, Vol. 3, No 2, pp. 356–362, DOI: 10.48175/ijarsct-8837.
- [13] SHAH M., KANTAWALA H., GANDHI K., PATEL R., PATEL K. A. and KOTHARI A. (2023), “Theoretical Evaluation of Ensemble Machine Learning Techniques”, In, *Proceedings - 5th International Conference on Smart Systems and Inventive Technology, ICSSIT 2023*, pp. 829–837, IEEE, United States, DOI: 10.1109/ICSSIT55814.2023.10061139.
- [14] SAYAL A., JHA J., CHAITHRA N., GUPTA V., GUPTA A., GUPTA O. and MEMORIA M. (2023), “Neural Networks and Machine Learning”, *2023 IEEE 5th International Conference on Cybernetics, Cognition and Machine Learning Applications (ICCCMLA)*, pp. 58–63, Hamburg.
- [15] ISLAM M., CHEN G. and JIN S. (2019), “An Overview of Neural Network”, *American Journal of Neural Networks and Applications*, Vol. 5, No 1, pp. 7, DOI: 10.11648/j.ajnna.20190501.12.

- [16] SHRUTI P. and REKHA R. (2023), “A Review of Convolutional Neural Networks, its Variants and Applications”, *In, Proceedings of the 2023 International Conference on Intelligent Systems for Communication, IoT and Security (ICISCoIS)*, pp. 31–36, IEEE, Coimbatore, DOI: 10.1109/ICISCoIS56541.2023.10100412.
- [17] KHALDI R., EL AFIA A., CHIHEB R. and TABIK, S. (2023), “What is the Best RNN-cell Structure to Forecast Each Time Series Behavior?”, *Expert Systems with Applications*, Vol. 215, p. 119140, DOI: 10.1016/j.eswa.2022.119140.
- [18] DASTIDAR K. G., CAELEN O. and GRANITZER M. (2024), “Machine Learning Methods for Credit Card Fraud Detection: A Survey”, *IEEE Access*, Vol. 12, pp. 158939–158965, DOI: 10.1109/ACCESS.2024.3487298.
- [19] MIAO Y. and XU Y. (2024), “Random Forest-Based Analysis of Variability in Feature Impacts”, *2024 IEEE 2nd International Conference on Image Processing and Computer Applications*, pp. 1130–1135, Shenyang.
- [20] ALGHOFAILI Y., ALBATTAH A. and RASSAM M. A. (2020), “A Financial Fraud Detection Model Based on LSTM Deep Learning Technique”, *Journal of Applied Security Research*, Vol. 15, No 4, pp. 498–516, DOI: 10.1080/19361610.2020.1815491.
- [21] JAKUB KVITA (2016), *Visualizations of RNN Units*, <https://kvitajakub.github.io/2016/04/14/rnn-diagrams/>, DoA. 24.01.2024.
- [22] PRISCILLA C.V., PARABHA D.P. (2020), “Credit Card Fraud Detection: A Systematic Review”, *International Conference on Information, Communication and Computing Technology. ICICCT 2020*, pp. 290–303, Istanbul.
- [23] FEDERATION TRADE COMMISSION (2023), *As Nationwide Fraud Losses Top \$10 Billion in 2023, FTC Steps Up Efforts to Protect the Public*, <https://www.ftc.gov/news-events/news/press-releases/2024/02/nationwide-fraud-losses-top-10-billion-2023-ftc-steps-efforts-protect-public>, DoA. 24.01.2024.

- [24] AURNA N. F., HOSSAIN M. D., TAENAKA Y. and KADOBAYASHI Y. (2023), “Federated Learning-Based Credit Card Fraud Detection: Performance Analysis with Sampling Methods and Deep Learning Algorithms”, *In, Proceedings of the 2023 IEEE International Conference on Cyber Security and Resilience (CSR)*, pp. 180–186, IEEE, Venice, DOI: 10.1109/CSR57506.2023.10224978.
- [25] LI L., FAN Y., TSE M. and LIN K. Y. (2020), “A Review of Applications in Federated Learning”, *Computers & Industrial Engineering*, Vol. 149, pp. 1-15, DOI: 10.1016/j.cie.2020.106854.
- [26] Jiang J. C., Kantarci B., Oktug S., and Soyata T. (2020), “Federated Learning in Smart City Sensing: Challenges and Opportunities”, *Sensors*, Vol. 20, No 21, pp. 1-29, DOI: 10.3390/s20216230.
- [27] TENSOR FLOW (2024), *Federated Learning*, <https://www.tensorflow.org/federated>, DoA. 25.12.2024.
- [28] FEDAI (2024), *Federated AI Ecosystem*, <https://www.fedai.org/>, DoA. 25.12.2024.
- [29] SHANBHOG N. R., TOTAD K. S., HANCHINAL A. R. and BIDARGADDI A. P. (2024), “Fraud Detection in Financial Transactions Using Deep Learning Approach: A Comparative Study”, *2024 5th International Conference for Emerging Technology, INCET 2024*, pp. 1-7, Karnataka.
- [30] SHOME N., SARKAR D. D., KASHYAP R. and LASKAR R. H. (2024), “Detection of Credit Card Fraud with Optimized Deep Neural Network in Balanced Data Condition”, *Computer Science*, Vol. 25, No 2, pp. 1–24, DOI: 10.7494/csci.2024.25.2.5967.
- [31] HINDAWI (2023), “Retracted: Financial Fraud Detection in Healthcare Using Machine Learning and Deep Learning Techniques”, *Security and Communication Networks*, Vol. 2023, pp. 1–1, DOI: 10.1155/2023/9758612.
- [32] SUGANTHI V. and JEBATHANGAM J. (2024), “A Novel Approach for Credit Card Fraud Detection using Gated Recurrent Unit (GRU) Networks”, *In, Proceedings of the 2024 8th International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC)*, pp. 1716–1721, IEEE, Kirtipur, DOI: 10.1109/I-SMAC61858.2024.10714795.

- [33] SHUKLA P., AGGARWAL M., JAIN P., KHANNA P. and RANA M. K. (2023), “Financial Fraud Detection and Comparison Using Different Machine Learning Techniques”, In, *Proceedings of the 2023 3rd International Conference on Technological Advancements in Computational Sciences (ICTACS)*, pp. 1205–1210, IEEE, Tashkent, DOI: 10.1109/ICTACS59847.2023.10390165.
- [34] ILEBERI E., SUN Y. and WANG Z. (2021), “Performance Evaluation of Machine Learning Methods for Credit Card Fraud Detection Using SMOTE and AdaBoost”, *IEEE Access*, Vol. 9, pp. 165286–165294, DOI: 10.1109/ACCESS.2021.3134330.
- [35] ALI N. T., HASAN S. J., GHANDOUR A. and AL-HCHIMY Z. S. (2024), “Improving Credit Card Fraud Detection Using Machine Learning and GAN Technology”, *Fifth International Scientific Conference of Alkafeel University (ISCKU 2024)*, pp. 1-18, Najaf.
- [36] WANG J. and YANG C. (2022), “Financial Fraud Detection Based on Ensemble Machine Learning”, *2022 IEEE Intl Conf on Dependable, Autonomic and Secure Computing, Intl Conf on Pervasive Intelligence and Computing, Intl Conf on Cloud and Big Data Computing, Intl Conf on Cyber Science and Technology Congress (DASC/PiCom/CBDCoM/CyberSciTech)*, pp. 1–6, Falerna.
- [37] BAGGA S., GOYAL A., GUPTA N. and GOYAL A. (2020), “Credit Card Fraud Detection using Pipelining and Ensemble Learning”, *Procedia Computer Science*, Vol. 173, pp. 104–112, DOI: 10.1016/j.procs.2020.06.014.
- [38] DANG T. K. and HA T. (2024), “A Comprehensive Fraud Detection for Credit Card Transactions in Federated Averaging”, *SN Computer Science*, Vol. 5, No. 5, pp. 1-10, DOI: 10.1007/s42979-024-02898-y.
- [39] BAABDULLAH T., ALZHRANI A., RAWAT D. B. and LIU C. (2024), “Efficiency of Federated Learning and Blockchain in Preserving Privacy and Enhancing the Performance of Credit Card Fraud Detection (CCFD) Systems”, *Future Internet*, Vol. 16, No 6, pp. 1-22, DOI: 10.3390/fi16060196.
- [40] DJOLEV D., LAZAROVA M. and NAKOV O. (2024), “FBLearn: Decentralized Platform for Federated Learning on Blockchain”, *Electronics*, Vol. 13, No 18, pp. 1-29, DOI: 10.3390/electronics13183672.

- [41] TANG Y. and LIANG Y. (2024), “Credit Card Fraud Detection Based on Federated Graph Learning”, *Expert Systems with Applications*, Vol. 256, pp.1-13, DOI: 10.1016/j.eswa.2024.124979.
- [42] BIAN K. and ZHENG H. (2023), “FedAvg-DWA: A Novel Algorithm for Enhanced Fraud Detection in Federated Learning Environment”, *2023 4th International Conference on Big Data, Artificial Intelligence and Internet of Things Engineering, ICBAIE 2023*, pp. 13–17, Hangzhou.
- [43] MYALIL D., RAJAN M. A., APTE M. and LODHA S. (2021), “Robust Collaborative Fraudulent Transaction Detection using Federated Learning”, *Proceedings - 20th IEEE International Conference on Machine Learning and Applications, ICMLA 2021*, pp. 373–378, Pasadena.
- [44] MYALIL D., RAJAN M. A., APTE M. and LODHA S. (2023), “Robust Cross-Silo Federated Fraudulent Transaction Detection in Banks Using Epsilon Cluster Selection”, *SN Computer Science*, Vol. 4, No 4, pp. 2-11, DOI: 10.1007/s42979-023-01873-3.
- [45] ZHANG Z., CAO X., JIA J. and GONG N. Z. (2022), “FLDetector: Defending Federated Learning Against Model Poisoning Attacks via Detecting Malicious Clients”, In, *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '22)*, pp. 2545–2555, Association for Computing Machinery, New York, DOI: 10.1145/3534678.3539231.
- [46] SHI Y., SONG H. and XU J. (2023), “Responsible and Effective Federated Learning in Financial Services: A Comprehensive Survey”, In, *Proceedings of the 2023 62nd IEEE Conference on Decision and Control (CDC)*, pp. 4229–4236, IEEE, Singapore, DOI: 10.1109/CDC49753.2023.10384119.
- [47] SUN Y., WONG A. K. C. and KAMEL M. S. (2009), “Classification of Imbalanced Data: A Review”, *International Journal of Pattern Recognition Artificial Intelligence*, Vol. 23, No 4, pp. 687–719, DOI: 10.1142/S0218001409007326.
- [48] ELREEDY D. and ATIYA A. F. (2019), “A Comprehensive Analysis of Synthetic Minority Oversampling Technique (SMOTE) for Handling Class Imbalance”, *Information Sciences*, Vol. 505, pp. 32–64, DOI: 10.1016/j.ins.2019.07.070.

- [49] MOHAMMED R. A., WONG K. W., SHIRATUDDIN M. F. and WANG X. (2018), “Scalable Machine Learning Techniques for Highly Imbalanced Credit Card Fraud Detection: A Comparative Study”, In, *PRICAI 2018: Trends in Artificial Intelligence*, Eds. Xin Geng and Byeong-Ho Kang, pp. 237–246, Springer, Cham, DOI: 10.1007/978-3-319-97310-4_27.
- [50] CHERIF A., BADHIB A., AMMAR H., ALSHEHRI S., KALKATAWI M. and IMINE A. (2023), “Credit Card Fraud Detection in the Era of Disruptive Technologies: A Systematic Review”, *Journal of King Saud University-Computer and Information Science*, Vol. 35, No 1, pp. 45–174, DOI: 10.1016/j.jksuci.2022.11.008.
- [51] CLOUD D. S. and ADNAN G. (2023), “Iraqis Fly to Nearby Countries to Get Dollars as U.S. Cracks Down on Money Laundering”, *The Wall Street Journal*, <https://www.wsj.com/articles/iraqis-fly-to-nearby-countries-to-get-dollars-as-u-s-cracks-down-on-money-laundering-ec366494>, DoA. 25.12.2024.
- [52] DATASET Kaggle (2024), *Credit Card Fraud Detection*, <https://www.kaggle.com/datasets/mlg-ulb/creditcardfraud>, DoA. 25.12.2024.
- [53] COLAB (2024), *Welcome to Colab*, <https://colab.research.google.com/>, DoA. 25.12.2024.
- [54] SKULA I., BOHACIK J. and ZABOVSKY M. (2020), “Use of Different Channels for User Awareness and Education Related to Fraud and Phishing in a Banking Institution”, In, *Proceedings of the 2020 18th IEEE International Conference on Emerging eLearning Technologies and Applications (ICETA)*, pp. 606–612, IEEE, Košice, DOI: 10.1109/ICETA51985.2020.9379220.