INSTITUTE FOR HOMELAND SECURITY

Sam Houston State University

Enhancing Fraud Detection in Financial Transactions Using Random Forest and Graph Convolutional Neural Networks

Institute for Homeland Security

Sam Houston State University

Qingzhong Liu

Abstract

Global financial crimes in 2023 resulted in an estimated $31 trillion in losses, according to the latest report from the Organized Crime and Corruption Reporting Project (OCCRP). These crimes, including money laundering, embezzlement, tax evasion, and illicit trading, pose a significant threat to the stability and fairness of the global economy. Despite efforts by governments and financial institutions to curb these activities, criminals continue to exploit complex financial networks and emerging technologies, such as cryptocurrencies, to evade detection.

In this study, we employ both Random Forest and Graph Convolutional Neural Network (GCN) models to detect fraudulent financial transactions. Our findings indicate that the Random Forest model consistently outperforms the GCN across various metrics in two distinct datasets. Specifically, Random Forest achieves higher and more stable ROC-AUC scores, precision, and recall, making it a more dependable model for these datasets.

While the GCN demonstrates potential in capturing intricate relationships within graph-structured data, its performance is less consistent, showing a greater tendency towards false positives, particularly at certain values of k. This variability suggests that GCN may benefit from more refined tuning and alternative approaches to feature engineering or graph construction to enhance its performance.

Based on the experimental results, Random Forest emerges as the preferred model for these datasets, offering superior overall performance, stability, and reliability. However, GCN remains a viable option in scenarios where complex interrelationships between data points are crucial, albeit with the need for further optimization.

Keywords: Money laundering, Financial Crime, Fraud detection, Random Forest, Graph Convolutional Neural Network (GCN), ROC-AUC, Precision, Recall

I. Financial Fraud and the Detection

Financial crime is a global epidemic that transcends borders and legal boundaries. From individual fraudsters to transnational criminal organizations, perpetrators quickly adapt to new defenses, exploiting emerging technologies and sharing illicit strategies for profit. Their activities are

estimated to cause over $485 billion in global fraud losses and fuel more than $3.1 trillion in money laundering and terrorist financing worldwide [1].

The scale of this problem is staggering. In 2023, banks globally faced projected losses of $442 billion from payments, check, and credit card fraud. Consumer scams affected every region, resulting in an estimated $43.6 billion in losses borne by individuals and businesses, often with life-altering consequences [1].

These fraud schemes, along with heinous crimes such as human trafficking, drug trafficking, and terrorist activity, have facilitated the flow of trillions of dollars in illicit funds through the global financial system over the past year. This enormous scale of financial crime threatens the integrity of our financial system and undermines communities worldwide.

In USA, according to the FBI's Internet Crime Report 2023, 880,418 complaints were reported to the FBI by the public, a 10 percent increase from 2022. The potential total loss increased to $12.5 billion in 2023, up from $10.3 billion in 2022, wherein the investment fraud losses rose from $3.31 billion in 2022 to $4.57 billion in 2023, a 38% increase [2]. The Federal Trade Commission's (FTC) data shows that consumers reported losing more than $10 billion to fraud in 2023, marking the first time that fraud losses have reached that benchmark. This marks a 14% increase over reported losses in 2022. US families were scammed out of more than $4.6 billion, which is higher than any other category in 2023. That amount represents a 21% increase over 2022 [3].

Financial fraud poses significant risks to individuals and organizations. Traditional fraud detection methods are often limited in their ability to handle large, complex datasets and identify sophisticated fraud patterns. AI and machine learning (ML) offer a promising solution by leveraging their capabilities to analyze vast amounts of data and learn from patterns that may indicate fraudulent activity.

A common challenge in financial fraud detection is the issue of class imbalance. This occurs when the number of instances in one class (e.g., fraudulent transactions) is significantly smaller than the number of instances in the other class (e.g., legitimate transactions). This imbalance can hinder the performance of ML models, as they may become biased towards the majority class [4].

II. Our Study for Financial Fraud Detection

Based on our previous experience in financial data analysis, machine learning, and deep learning, in this study, we focus on the detection of financial fraudulent transactions by using Random Forest [5] and KNN-Graphic Convolution Neural Network [6].

1) Data Sets

The first financial fraud transaction dataset used in our experiments is available on Kaggle [7]. This dataset provides comprehensive information about transactions, with a particular focus on identifying fraudulent activities. With over 6 million entries, it offers a rich and diverse collection of transactional data for analysis and modeling. step: Represents a unit of time in the transaction process, though the specific time unit is not specified in the dataset. It could denote hours, days, or another unit, depending on the context. The data set contains the following attributes:

type: Describes the type of transaction, such as transfer, payment, etc. This categorical variable allows for the classification of different transaction behaviors.
amount: Indicates the monetary value of the transaction, providing insight into the financial magnitude of each transaction.
nameOrig: Serves as the identifier for the origin account or entity initiating the transaction. This helps trace the source of funds in each transaction.
oldbalanceOrg: Represents the balance in the origin account before the transaction occurred, offering a reference point for understanding changes in account balances.
newbalanceOrig: Reflects the balance in the origin account after the transaction has been processed, providing insight into how the transaction affects the account balance.
nameDest: Functions as the identifier for the destination account or entity receiving the funds in each transaction. It helps track where the money is being transferred to.
oldbalanceDest: Indicates the balance in the destination account before the transaction, offering a baseline for assessing changes in account balances due to incoming funds.
newbalanceDest: Represents the balance in the destination account after the transaction has been completed, providing insight into the impact of incoming funds on the account balance.
isFraud: A binary indicator (0 or 1) denoting whether the transaction is fraudulent (1) or legitimate (0). This is the target variable for fraud detection modeling.
i**sFlaggedFraud**: Another binary indicator (0 or 1) which may signal whether a transaction has been flagged as potentially fraudulent. This could serve as an additional feature for fraud detection algorithms.

Below please find the first fifteen rows of the data.

step	type	amount	nameOrig	oldbalanceOrg	newbalanceOrig	nameDest	oldbalanceDest	newbalanceDest	isFraud	isFlaggedFraud
1	PAYMENT	9839.64	C1231006815	170136.00	160296.36	M1979787155	0.0	0.00	0	0
1	PAYMENT	1864.28	C1666544295	21249.00	19384.72	M2044282225	0.0	0.00	0	0
1	TRANSFER	181.00	C1305486145	181.00	0.00	C553264065	0.0	0.00	1	0
1	CASH_OUT	181.00	C840083671	181.00	0.00	C38997010	21182.0	0.00	1	0
1	PAYMENT	11668.14	C2048537720	41554.00	29885.86	M1230701703	0.0	0.00	0	0
1	PAYMENT	7817.71	C90045638	53860.00	46042.29	M573487274	0.0	0.00	0	0
1	PAYMENT	7107.77	C154988899	183195.00	176087.23	M408069119	0.0	0.00	0	0
1	PAYMENT	7861.64	C1912850431	176087.23	168225.59	M633326333	0.0	0.00	0	0
1	PAYMENT	4024.36	C1265012928	2671.00	0.00	M1176932104	0.0	0.00	0	0
1	DEBIT	5337.77	C712410124	41720.00	36382.23	C195600860	41898.0	40348.79	0	0
1	DEBIT	9644.94	C1900366749	4465.00	0.00	C997608398	10845.0	157982.12	0	0
1	PAYMENT	3099.97	C249177573	20771.00	17671.03	M2096539129	0.0	0.00	0	0
1	PAYMENT	2560.74	C1648232591	5070.00	2509.26	M972865270	0.0	0.00	0	0
1	PAYMENT	11633.76	C1716932897	10127.00	0.00	M801569151	0.0	0.00	0	0
1	PAYMENT	4098.78	C1026483832	503264.00	499165.22	M1635378213	0.0	0.00	0	0

The data set contains 6354407 normal transaction records and 8213 fraud records. The ratio of normal to fraud is 773.7:1.

The data set is highly imbalanced. Synthetic Minority Over-sampling Technique (SMOTE) [8] is a popular oversampling technique used to address the issue of class imbalance in machine learning datasets. It's particularly effective when dealing with datasets where one class (the minority class) is significantly smaller than the other (the majority class). The steps using SMOTE are listed below.

1. **Identify Minority Class Instances:** SMOTE first locates the instances belonging to the minority class.

2. **Compute K-Nearest Neighbors:** For each minority class instance, it finds its k nearest neighbors within the minority class.
3. **Generate Synthetic Instances:** New synthetic instances are generated along the line segments connecting the minority class instance to its k nearest neighbors. These synthetic instances are created by interpolating between the feature values of the original instance and its neighbors.

The benefits with SMOT include:

- **Increases Minority Class Samples:** By creating new synthetic instances, SMOTE effectively increases the number of samples in the minority class, making the dataset more balanced.
- **Preserves Class Distribution:** SMOTE generates synthetic instances that are similar to existing minority class instances, helping to maintain the original class distribution.
- **Improves Classifier Performance:** By addressing class imbalance, SMOTE can improve the performance of machine learning classifiers, especially those that are sensitive to imbalanced datasets.

The second data file, aml_syn_data.csv, was provided by JPMorgan Chase bank [9]. It contains 1173586 good transactions and 310950 bad transactions. The ratio of good to bad is 3.77:1. The first few records are shown below.

Time_step	Label	Transaction_Id	Sender_Id	Sender_Account	Sender_Institution	Sender_Country	USD_amount	Bene_Country	Transaction_Type
2022-01-07 00:02:00	GOOD	T-174791-02	JPMC-CLIENT-174571-02	CHECKING-174582-02	JPMORGANCHASE	USA	0.0000	USA	KYC-ADD-ACCOUNT-OWNER
2022-01-07 00:02:00	GOOD	T-105637-03	JPMC-CLIENT-105413-03	CHECKING-105426-03	JPMORGANCHASE	USA	0.0000	USA	KYC-ADD-ACCOUNT-OWNER
2022-01-07 00:02:00	GOOD	T-235858-04	COMPANY-235572-04	CHECKING-235577-04	JPMORGANCHASE	USA	0.0000	USA	KYC-ADD-ACCOUNT-OWNER
2022-01-07 00:02:00	GOOD	T-379320-02	JPMC-CLIENT-379064-02	CHECKING-379071-02	BANK-379167-02	USA	68.3500	USA	WIRE
2022-01-07 00:02:00	GOOD	T-413316-04	JPMC-CLIENT-413218-04	CHECKING-413226-04	JPMORGANCHASE	USA	0.0000	USA	KYC-ADD-ACCOUNT-OWNER
2022-01-07 00:02:00	GOOD	T-256459-02	COMPANY-256174-02	CHECKING-256180-02	JPMORGANCHASE	USA	0.0000	USA	KYC-ADD-ACCOUNT-OWNER
2022-01-07 00:02:00	GOOD	T-483373-04	COMPANY-483228-04	CHECKING-483229-04	JPMORGANCHASE	USA	869.0600	USA	WIRE
2022-01-07 00:02:00	GOOD	T-1718316-04	BILLING-COMPANY-1718175-04	CHECKING-1718173-04	BANK-1718317-04	USA	5249.9700	USA	WIRE
2022-01-07 00:02:00	GOOD	T-291469-06	JPMC-CLIENT-291195-06	CHECKING-291203-06	JPMORGANCHASE	USA	5224.6700	NaN	WIRE
2022-01-07 00:02:00	GOOD	T-1130131-06	COMPANY-1129922-06	CHECKING-1129929-06	JPMORGANCHASE	USA	0.0000	USA	KYC-ADD-ACCOUNT-OWNER
2022-01-07 00:02:00	GOOD	T-824056-10	NaN	NaN	NaN	NaN	0.0000	CANADA	KYC-CREATE-ACCOUNT
2022-01-07 00:02:00	GOOD	T-498146-04	BILLING-COMPANY-497986-04	CHECKING-497984-04	BANK-498147-04	USA	2356.7900	USA	WIRE
2022-01-07 00:06:00	GOOD	T-1913283-04	JPMC-CLIENT-1913045-04	CHECKING-1913053-04	BANK-1913155-04	USA	262.5400	USA	PAYMENT
2022-01-07 00:06:00	GOOD	T-623763-06	CUSTOMER-623641-06	CHECKING-623644-06	JPMORGANCHASE	USA	0.0000	USA	KYC-ADD-ACCOUNT-OWNER
2022-01-07 00:06:00	GOOD	T-63616-02	BILLING-COMPANY-63463-02	CHECKING-63461-02	BANK-63617-02	USA	2757.0200	USA	WIRE
2022-01-07 00:06:00	GOOD	T-78647-04	JPMC-CLIENT-78525-04	CHECKING-78531-04	JPMORGANCHASE	USA	43.1000	USA	WIRE
2022-01-07 00:06:00	GOOD	T-108370-04	BILLING-COMPANY-108229-04	CHECKING-108227-04	BANK-108371-04	USA	3813.1702	USA	WIRE
2022-01-07 00:06:00	GOOD	T-1177331-04	BILLING-COMPANY-1177173-04	CHECKING-1177171-04	JPMORGANCHASE	USA	3660.2603	USA	WIRE
2022-01-07 00:06:00	GOOD	T-2534994-04	JPMC-CLIENT-2534930-04	CHECKING-2534936-04	JPMORGANCHASE	USA	605.2200	USA	WIRE
2022-01-07 00:06:00	GOOD	T-1220401-06	COMPANY-1220283-06	CHECKING-1220284-06	JPMORGANCHASE	USA	4448.1200	UNITED-KINGDOM	WIRE

2) Fraud Detection with Random Forest

Random Forest is an ensemble learning method that combines multiple decision trees to improve classification or regression accuracy. It is based on the principles of "bagging" (Bootstrap Aggregating) and "random feature selection."

1. **Decision Tree Basics:** A decision tree is a tree-like model used for decision making. Each internal node represents a "test" or "decision" on an attribute, each branch represents the outcome of the test, and each leaf node represents a class label (in classification) or a continuous value (in regression).
2. **Bootstrap Sampling:** For each tree in the forest, a bootstrap sample D_i is created by randomly sampling n examples with replacement from the training set D of size n.

$$\mathcal{D}_i = \{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\} \quad (1)$$

3. **Random Feature Selection:** At each node of the tree, instead of considering all features p, a random subset of m features ($m<p$) is selected to determine the best split. This ensures that the trees are diverse.
4. **Tree Construction:** Each decision tree T_i is grown using the bootstrap sample D_i and random feature selection. The tree continues to split until the maximum depth is reached or another stopping criterion (like minimum number of samples in a node) is met.
5. **Prediction Aggregation:**
   - For classification: Each tree T_i provides a class prediction . The final class prediction is made by majority voting.

$$\hat{y} = \text{mode}(\{\hat{y}_1, \hat{y}_2, \ldots, \hat{y}_k\}) \quad (2)$$

   - For regression: Each tree T_i provides a predicted value. The final prediction is the average of all tree predictions.

$$\hat{y} = \frac{1}{k} \sum_{i=1}^{k} \hat{y}_i \quad (3)$$

Applying Random Forest, our detection outcomes are shown below, wherein, 0 means normal transaction (negative), 1 indicates fraud transactions (positive). In the processing of the original data, we dropped the attributes 'isFraud', 'isFlaggedFraud', 'nameOrig', 'nameDest', 'oldbalanceDest', 'newbalanceDest'.

We did 10 experiments on the aml_syn_data.csv, we randomly selected 60% for training and the remaining 40% for testing in each experiment. The mean value and standard deviation of the 10 experiments on testing data are listed below.

Table 1. The mean and standard deviation of the 10 experiments on testing data on the JPMorgan Chase data file

Testing metric	With SMOTE		Without SMOTE	
	Mean	Standard deviation	Mean	Standard deviation
ROC-AUC	0.999	6.4E-05	0.999	4.7E-05
True Positives	123518.9	63.7	123545.7	45.5
False Positives	972.6	51.8	1016.7	49.7
True Negatives	468462.4	51.8	468418.3	49.7
False Negatives	861.1	63.7	834.3	45.5
Negative Precision	0.998	1.4E-04	0.998	9.7E-05
Negative Recall	0.998	1.1E-04	0.998	1.1E-04
Negative F1-Score	0.998	2.5E-05	0.998	3.3E-05
Positive Precision	0.993	4.1E-04	0.992	3.9E-04
Positive Recall	0.993	5.1E-04	0.993	3.7E-04
Positive F1-Score	0.993	9.8E-05	0.993	1.2E-04

Table 2. The mean and standard deviation of the 10 experiments on the Kaggle data file

Testing metric	With SMOTE		Without SMOTE	
	Mean	Standard deviation	Mean	Standard deviation
ROC-AUC	0.998	3.7E-04	0.996	9.5E-04
True Positives	2820.2	42.6	2785.5	25.0
False Positives	353.4	37.2	387.4	34.3
True Negatives	2541409.6	37.2	2541375.6	34.3
False Negatives	464.8	42.6	499.5	25.0
Negative Precision	0.9998	1.7E-05	0.9998	9.8E-06
Negative Recall	0.9998	1.5E-05	0.9998	1.4E-05
Negative F1-Score	0.9998	4.1E-06	0.9998	4.5E-06
Positive Precision	0.8889	9.2E-03	0.8780	8.8E-03
Positive Recall	0.8585	1.3E-02	0.8480	7.6E-03
Positive F1-Score	0.8733	3.9E-03	0.8627	3.2E-03

3) Fraud Detection with KNN Graphic Convolutional Network

A Graph Convolutional Network (GCN) is a neural network that operates directly on graph structures, capturing dependencies between nodes based on the graph topology.

K-Nearest Neighbors (KNN) graph creation is a widely used technique in machine learning, particularly in the context of graph-based models like Graph Neural Networks (GNNs). The KNN graph is constructed by connecting each data point (node) to its k nearest neighbors based on some distance metric, typically Euclidean distance.

Advantages of KNN Graph Creation:

1. **Local Structure Preservation**:
    - **Capture Local Relationships**: KNN graphs preserve the local structure of the data by ensuring that each node is connected to its closest neighbors. This is particularly important in scenarios where local interactions are more relevant than global ones, such as in community detection, clustering, or when analyzing spatial data.
2. **Simplicity and Interpretability**:
    - **Easy to Understand**: KNN graph construction is straightforward and intuitive. It is easy to interpret the connections between nodes because they are based on proximity in the feature space.
    - **Parameter Control**: The only major parameter to tune is k, which controls the graph's sparsity and density, making it simpler to experiment with compared to more complex graph construction methods.
3. **Scalability**:
    - **Efficient Computation**: For moderate-sized datasets, KNN graph creation is computationally efficient, especially when using optimized algorithms like Ball Tree or KD-Tree for nearest neighbor search.
4. **Versatility**:

- **Applicable Across Domains**: KNN graphs can be applied to various types of data, including images, text, and tabular data. It is versatile and can be used in tasks ranging from classification and regression to clustering and dimensionality reduction.

Disadvantages of KNN Graph Creation:

1. **Choice of k**:
   - **Sensitivity to k**: The performance of models using KNN graphs can be sensitive to the choice of k. A small k might not capture enough relationships, leading to disconnected or sparse graphs, while a large k could introduce noise by connecting unrelated nodes.
   - **No Universal Optimal k**: There is no universally optimal k value; it often requires empirical tuning and cross-validation, which can be time-consuming.
2. **Computational Complexity**:
   - **High Complexity for Large Datasets**: Although efficient algorithms exist, the computation of the KNN graph can still be expensive for very large datasets, especially in high-dimensional spaces where the curse of dimensionality makes nearest neighbor search more challenging.
3. **Limited Global Structure**:
   - **Focus on Local Relationships**: While KNN graphs are excellent at preserving local structure, they may fail to capture important global relationships in the data. This limitation can be problematic in tasks where global context is crucial.
4. **Potential for Noise**:
   - **Inclusion of Irrelevant Neighbors**: Especially when using a larger k, there is a risk of including neighbors that are not semantically similar, which can introduce noise into the model and reduce performance.

KNN graph creation is a powerful tool for graph-based modeling, particularly when local structure is important. It offers simplicity and scalability, but the choice of k and the computational cost can be challenging, especially for large or high-dimensional datasets. The decision to use a KNN graph should be based on the specific requirements of the task at hand, with careful consideration of the advantages and limitations.

a. K-Nearest Neighbors (KNN) Graph Creation

The first part of the script creates a KNN graph, where each node represents a data point and edges connect nodes to their k nearest neighbors.

- **KNN Graph**: First, we need to create a function to build a sparse matrix where each row represents a node and contains binary values indicating connections to the k nearest neighbors.
- **Edge Index & Weight**: Then we build a function to converts the sparse matrix into an edge list and edge weights.

b. Graph Convolutional Network (GCN)

A GCN is designed to operate on graph-structured data. Here we define a simple GCN with two graph convolutional layers.

- **Graph Convolution Operation**: The GCN layers perform graph convolutions, which aggregate features from neighboring nodes.

Mathematically, the output feature $\mathbf{H}^{(l+1)}$ of layer l+1 in a GCN can be expressed as:

$$\mathbf{H}^{(l+1)} = \sigma\left(\hat{\mathbf{A}}\mathbf{H}^{(l)}\mathbf{W}^{(l)}\right) \tag{4}$$

where:

- $\mathbf{H}^{(l)}$ is the input feature matrix at layer l,
- $\hat{\mathbf{A}}$ is the normalized adjacency matrix of the graph,
- $\mathbf{W}^{(l)}$ is the weight matrix of the layer, and
- σ(·) is a non-linear activation function, like Rectified Linear Unit (ReLU) activation function [10].
- **Normalization of Adjacency Matrix**: is defined as:

$$\hat{\mathbf{A}} = \mathbf{D}^{-\frac{1}{2}}\mathbf{A}\mathbf{D}^{-\frac{1}{2}} \tag{5}$$

where A is the adjacency matrix of the graph and D is the degree matrix with

$$D_{ii} = \sum_j A_{ij} \tag{6}$$

c. Forward Propagation in GCN

- **First Layer**: The first convolutional layer aggregates features from neighboring nodes to learn hidden representations:

$$\mathbf{H}^{(1)} = \mathrm{ReLU}\left(\hat{\mathbf{A}}\mathbf{X}\mathbf{W}^{(0)}\right) \tag{7}$$

where X is the input feature matrix and Rectified Linear Unit (ReLU) activation function introduces the property of nonlinearity to a deep learning model and solves the vanishing gradients issue [10].

- **Second Layer**: The second convolutional layer (`conv2`) performs another aggregation, transforming the hidden representations into the final output space:

$$\mathbf{Z} = \hat{\mathbf{A}}\mathbf{H}^{(1)}\mathbf{W}^{(1)} \tag{8}$$

Then we apply a softmax function to the output to obtain class probabilities. The softmax function, also known as softargmax [11], converts a vector of K real numbers into a probability distribution of K possible outcomes.

$$Output=Softmax(Z) \tag{9}$$

d. Final Output

The final output of the GCN is a log-softmax of the predicted class scores for each node:

$$Y=log\_softmax(Z, dim=1) \tag{10}$$

This output can be used for node classification tasks, where each node is assigned to a class based on the learned features. Here,

$$log\_softmax(x) = log(softmax(x)) \tag{11}$$

The GCN's power lies in its ability to capture the structure of the graph, allowing it to effectively model the relationships between data points in non-Euclidean space.

Table 3 and Table 4 show the mean and standard deviations by using Graphic Convolution Neural Network with different k to the two data sets.

Table 3. The mean and standard deviation of the 10 experiments on testing data on the JPMorgan Chase data file with Graphic Convolution Neural Network

Testing metric	K = 5		K = 10		K = 20	
	Mean	Standard deviation	Mean	Standard deviation	Mean	Standard deviation
ROC-AUC	0.888	0.053	0.912	0.014	0.919	0.015
True Positives	56051.6	5186.2	53571.9	3855.6	56155.5	4059.6
False Positives	29365.0	35780.9	8729.6	10006.7	15199.9	11536.5
True Negatives	205353.0	35780.9	225988.4	10006.7	219518.1	11536.5
False Negatives	6138.4	5186.2	8618.1	3855.6	6034.5	4059.6
Negative Precision	0.975	0.022	0.962	0.015	0.974	0.017
Negative Recall	0.875	0.151	0.963	0.043	0.936	0.051
Negative F1-Score	0.910	0.092	0.964	0.017	0.953	0.020
Positive Precision	0.751	0.210	0.882	0.108	0.812	0.122
Positive Recall	0.903	0.083	0.861	0.064	0.903	0.065
Positive F1-Score	0.791	0.118	0.865	0.036	0.845	0.042

Table 4. The mean and standard deviation of the detection on the Kaggle data file with Graphic Convolution Neural Network

Testing metric	K = 5		K = 10		K = 20	
	Mean	Standard deviation	Mean	Standard deviation	Mean	Standard deviation
ROC-AUC	0.872	0.020	0.799	0.115	0.833	0.065
True Positives	1356	116	1386	202	1418	174
False Positives	103281	76336	313443	397423	252465	269196
True Negatives	1167600	76336	957438	397423	1018416	269196

False Negatives	287	116	257	202	224	174
Negative Precision	1.000	0.000	1.000	0.000	1.000	0.000
Negative Recall	0.919	0.060	0.753	0.313	0.803	0.211
Negative F1-Score	0.957	0.032	0.813	0.253	0.873	0.149
Positive Precision	0.030	0.026	0.024	0.024	0.024	0.030
Positive Recall	0.825	0.071	0.841	0.123	0.864	0.106
Positive F1-Score	0.055	0.050	0.043	0.042	0.047	0.048

III. Analysis and Discussion

The experimental results in Tables 1 and 2 are from applying a Random Forest classifier to two datasets (`aml_syn_data.csv` and a Kaggle dataset) for fraud detection. The results are summarized in two tables, showing the mean and standard deviation of various metrics across 10 experiments, comparing the performance with and without applying the SMOTE (Synthetic Minority Over-sampling Technique) method.

Table 1 shows that the model performed extremely well in distinguishing between normal and fraudulent transactions, with nearly perfect ROC-AUC scores, both with and without SMOTE. The very low standard deviation indicates consistent performance across the experiments. The model successfully identified fraudulent transactions (99.3%) and normal transactions (99.8%), with similar performance with and without SMOTE. The standard deviations are low, indicating stable results across different experiments. The model made slightly fewer false positive errors with SMOTE compared to without, showing a minor improvement in this aspect with SMOTE. The mean values are extremely high for Precision (0.998 for normal and 0.993 for fraud), Recall (0.998 for normal and 0.993 for fraud), and F1-Score (0.998 for normal and 0.993 for fraud), indicating that the model is highly accurate in identifying normal transactions and fraudulent transactions. The differences between the two scenarios with/without SMOTE are negligible.

The results in Table 2 show that the ROC-AUC is slightly better with SMOTE (0.998 vs 0.996), and the smaller standard deviation indicates more consistent performance across experiments. The model detected more fraudulent transactions with SMOTE (2820.2 vs 2785.5), though the differences are relatively small. Negative Precision, Negative Recall, and Negative F1-Score on normal transactions are all 0.9998, which means that the detection of normal transaction is almost perfectly accurate. With SMOTE, Positive Precision, Positive Recall, and Positive F1-Score on fraudulent transactions are 0.8889, 0.8585, and 0.8733, respectively, which means that the detection of fraudulent transaction is not perfect as the normal transaction, but still nice. Without SMOTE, the detection of normal transaction is the same and the detection of fraudulent transaction is slightly lower, Positive Precision, Positive Recall, and Positive F1-Score on fraudulent transactions are 0.8780, 0.8480, and 0.8627, respectively.

Overall, by applying Random Forest with SMOTE generally improved the detection of fraudulent transactions, particularly in reducing false positives and improving the precision of the model. The differences between SMOTE and non-SMOTE are more pronounced in the Kaggle dataset compared to the JPMorgan Chase data file.

The experimental results in Tables 3 and 4 involve the application of a Graph Convolutional Network (GCN) to two datasets (JPMorgan Chase data and a Kaggle dataset) using different values of `k`, where `k` represents the number of nearest neighbors considered in the graph structure. The

results provide insights into how well the GCN can model relationships in the data and how varying k affects performance.

In Table 3, the ROC-AUC improves as k increases, indicating that the model becomes better at distinguishing between normal and fraudulent transactions. The improvement from k=5 (0.888) to k=10 (0.912) and k=20 (0.919) suggests that considering more neighbors in the graph improves the model's ability to capture complex relationships. The number of true positives is highest at k=20 (56155.5), indicating that the model detects more fraudulent transactions with more neighbors. The number of false positives decreases significantly when moving from k=5 (29365.0) to k=10 (8729.6), but then increases again at k=20 (15199.9). This suggests that while increasing k can improve TP (True Positive), it may also lead to more FP (False Positive) errors at higher values. The highest TN (True Negative) is achieved with k=10 (225988.4), indicating that this value of k strikes a balance in correctly identifying normal transactions.

In Table 4, the ROC-AUC is highest at k=5 (0.872) and decreases at k=10 (0.799), before slightly improving at k=20 (0.833). This suggests that smaller values of k might be better suited for this dataset. The number of true positives increases as k increases, 1356 (k=5), 1386 (k=10), and 1418 (k=20), indicating better detection of fraudulent transactions with more neighbors. The TN count is highest at k=5 (1167600), showing that smaller values of k might be more effective at correctly identifying normal transactions.

It should be noted that the Positive Precision values in Table 4 are 0.030 (k=5), 0.024 (k=10), and 0.030 (K=20), which means that the GCN performs poorly in detecting fraudulent transactions on the Kaggle financial crime data. The ratio of normal transaction to fraudulent transaction is 773.7:1, and the fraud category contains only 8213 records. From our standpoint, such a high imbalance ratio and small number of fraud records lead to the failure of the GCN model in exposing the fraudulent activities.

For JPMorgan Chase Data, increasing k generally improves the model's performance in detecting both normal and fraudulent transactions. However, there is a trade-off as false positives can increase with larger k.

For Kaggle Data, the GCN model shows more complex behavior, with smaller k generally yielding better results in ROC-AUC and true negative detection, though larger k improves the detection of true positives.

Comparing the results across Tables 1 to 4, it clearly demonstrates that Random Forest performs better than KNN-Graphic Convolution Neural Network. Either on Kaggle highly imbalance data or JPMorgan synthetic data, Random Forest delivers reliable and accurate detection performance. Although KNN-Graphic Convolution Neural Network performs well in detecting the fraudulent transactions on JPMorgan synthetic data, however, the accuracy and reliability dramatically deteriorates while detecting the Kaggle fraud data due to the high imbalance ratio and small number of fraud records, resulting in the incapability of representing the local structure of the fraudulent activities by using the KNN Graph.

We also observed that SMOTE may slightly improve the detection performance but not so well.

IV. Conclusion

In our study in applying Random Forest and graphic neural network for financial fraudulent transaction detection, Random Forest outperforms the GCN across most metrics in both datasets. It achieves higher and more consistent ROC-AUC scores, precision, and recall, with lower standard deviations, making it a more reliable model for these specific datasets.

GCN shows promise, particularly in capturing complex relationships in graph-structured data. However, its performance is less consistent, and it's more prone to false positives, especially when k is small or large. The variability in results suggests that GCN might require more careful tuning and potentially different approaches to feature engineering or graph construction to match or surpass Random Forest performance.

Given the experimental results, Random Forest is the preferred model for these datasets, as it offers better overall performance, stability, and reliability. GCN may still be useful in contexts where the relationships between data points are more complex or critical, but it would require further optimization.

Acknowledgements

The support for this study from SHSU Institute of Homeland Security is highly appreciated.

References

1. https://nd.nasdaq.com/rs/303-QKM-463/images/2024-Global-Financial-Crime-Report-Nasdaq-Verafin-20240119.pdf
2. Federal Bureau of Investigation Internet Crime Report 2023, Internet Crime Complaint Center. https://www.ic3.gov/media/pdf/annualreport/2023_ic3report.pdf
3. https://www.ftc.gov/system/files/ftc_gov/pdf/CSN-Annual-Data-Book-2023.pdf
4. Kamuangu, Paulin & K K, Paul. (2024). A Review on Financial Fraud Detection using AI and Machine Learning. Journal of Economics Finance and Accounting Studies. 6. 67-77. 10.32996/jefas.6.1.7.
5. Breiman, L. Random Forests. *Machine Learning* **45**, 5–32 (2001). https://doi.org/10.1023/A:1010933404324
6. Kang, Seokho. (2021). k-Nearest Neighbor Learning with Graph Neural Networks. Mathematics. 9. 830. 10.3390/math9080830.
7. https://www.kaggle.com/datasets/chitwanmanchanda/fraudulent-transactions-data
8. N. V. Chawla, K. W. Bowyer, L. O.Hall, W. P. Kegelmeyer, "SMOTE: synthetic minority over-sampling technique," Journal of artificial intelligence research, 321-357, 2002.
9. https://www.jpmorgan.com/technology/artificial-intelligence/initiatives/synthetic-data/request-synthetic-data
10. K. Fukushima (1969), "Visual Feature Extraction by a Multilayered Network of Analog Threshold Elements", in IEEE Transactions on Systems Science and Cybernetics, vol. 5, no. 4, pp. 322-333, Oct. 1969, doi: 10.1109/TSSC.1969.300225.
11. Goodfellow, Ian; Bengio, Yoshua; Courville, Aaron (2016). "6.2.2.3 Softmax Units for Multinoulli Output Distributions". Deep Learning. MIT Press. pp. 180–184. ISBN 978-0-26203561-3.

INSTITUTE FOR HOMELAND SECURITY

Sam Houston State University

The Institute for Homeland Security at Sam Houston State University is focused on building strategic partnerships between public and private organizations through education and applied research ventures in the critical infrastructure sectors of Transportation, Energy, Chemical, Healthcare, and Public Health.

The Institute is a center for strategic thought with the goal of contributing to the security, resilience, and business continuity of these sectors from a Texas Homeland Security perspective. This is accomplished by facilitating collaboration activities, offering education programs, and conducting research to enhance the skills of practitioners specific to natural and human caused Homeland Security events.

Institute for Homeland Security
Sam Houston State University

Liu, Q. (2024). Enhancing fraud detection in financial transactions using random forest and graph convolutional neural networks (Report No. IHS/CR-2024-1036). Sam Houston State University, Institute for Homeland Security.
https://doi.org/10.17605/OSF.IO/2W639