BIRLA INSTITUTE OF TECHNOLOGY AND SCIENCE PILANI
PILANI CAMPUS

Project Report: Payment Fraud Detection using Graph Neural Networks

Semester Project

Foundations of Data Science (CS F320)

UNDER SUPERVISION OF

PROFESSOR TEJASVI ALLADI, DEPARTMENT OF COMPUTER SCIENCE, BITS PILANI

Rishabh Sahni : 2021A7PS1630P
Samarth Khandelwal : 2021A3PS0051P
Nachiketh S Shastry : 2021A7PS2686P
Ridham Mittal : 2021A7PS1454P
Tauqeer Akhtar : 2021A7PS1628P

TABLE OF CONTENTS

Abstract

The exponential rise of digital financial transactions has made fraud detection a critical area of focus in the financial services industry. This project presents a comprehensive approach to detecting fraudulent transactions using advanced feature engineering and machine learning techniques. Our primary goal was to enhance the identification of fraudulent behavior within a large-scale financial transactions dataset by developing a robust classification model.

The process began with thorough exploratory data analysis (EDA) to understand the structure and intricacies of the dataset. Notably, we identified that certain columns such as `'nameOrig'` and `'nameDest'` possessed high cardinality. Through principal component analysis (PCA) and different encoding methods (one-hot, label, and hash encoding), we analyzed their true contribution to the model’s predictive power. This analysis guided our feature selection strategy, which focused on removing noise and retaining features that added genuine predictive value.

After preprocessing and transforming the dataset, we trained multiple machine learning models, including XGBoost, and optimized them using Optuna for hyperparameter tuning. Evaluation was conducted using standard metrics like precision, recall, F1-score, and ROC-AUC to measure the model's effectiveness.

The final model demonstrated strong performance in identifying fraudulent transactions, emphasizing the importance of data-driven feature selection and optimization. Our project provides not only an effective solution for fraud detection but also highlights best practices in handling high-cardinality categorical variables and model interpretability in sensitive domains such as finance.

Introduction

<u>Background on Fraud in Transactions</u>

With the digitalization of banking and finance, millions of transactions occur every second globally. This surge has created fertile ground for fraudulent activities, ranging from identity theft to synthetic fraud and transaction laundering. Traditional rule-based systems are no longer sufficient due to the scale and complexity of modern financial data. As a result, machine learning has emerged as a powerful alternative to detect patterns that indicate fraud, in real time or near-real time.

<u>Importance of Early Detection</u>

Early detection of fraudulent transactions is critical. Financial fraud not only causes direct monetary losses but also damages trust in institutions. The faster a system can identify and block fraudulent activities, the more effectively it can minimize losses, protect users, and comply with regulatory requirements. Moreover, early detection models can help in proactively flagging suspicious accounts and behaviors before large-scale frauds are committed.

<u>Brief about Dataset and Problem Formulation</u>

The dataset used in this project is a synthetic dataset commonly used in financial fraud detection challenges. It consists of millions of transaction records, each labeled as fraudulent or legitimate. Each transaction includes fields like sender and receiver identifiers, transaction amount, and a timestamp.

The objective of our project was to build a binary classification model to accurately predict whether a given transaction is fraudulent. We tackled this by:

- Carefully analyzing and engineering features.
- Removing low-value and misleading features.
- Choosing encoding strategies based on data distribution and PCA.
- Training and evaluating multiple machine learning models.

Our work not only builds a high-performance detection model but also documents a reproducible pipeline for fraud detection applications.

Dataset Description

The PaySim1 dataset is a synthetic dataset generated using an agent-based simulator that models mobile money transactions. Unlike many other fraud datasets, PaySim was specifically designed to address the challenge of data availability in financial fraud research. The simulator was developed by Dr. Edgar Lopez-Rojas as part of his PhD research and has been featured as Kaggle's dataset of the week in April 2018.

Dataset:https://www.kaggle.com/datasets/sriharshaeedala/financial-fraud-detection-dataset/data

The dataset contains over 6 million transactions with the following key features:

- step: Time step of the transaction (1 step = 1 hour)
- type: Transaction type (CASH-IN, CASH-OUT, DEBIT, PAYMENT, TRANSFER)
- amount: Transaction amount
- nameOrig: Customer initiating the transaction
- oldbalanceOrg: Balance before transaction
- newbalanceOrig: Balance after transaction
- nameDest: Recipient of the transaction
- oldbalanceDest: Recipient's balance before transaction
- newbalanceDest: Recipient's balance after transaction
- isFraud: Target variable indicating fraudulent transactions (1 for fraud, 0 for legitimate)
- isFlaggedFraud: System flag for fraud.

Other datasets were considered, including the Credit Card Fraud dataset which contains transactions made by European cardholders (with 492 frauds out of 284,807 transactions), and email fraud datasets, but these weren't suitable for our GNN approach. The Credit Card dataset was already PCA-transformed, hiding original feature meanings, while email datasets would have required complex text vector embeddings. PaySim1 provided explicit transaction relationships ideal for graph construction and GNN modelling.

Methodology

Data Preprocessing

Feature Selection

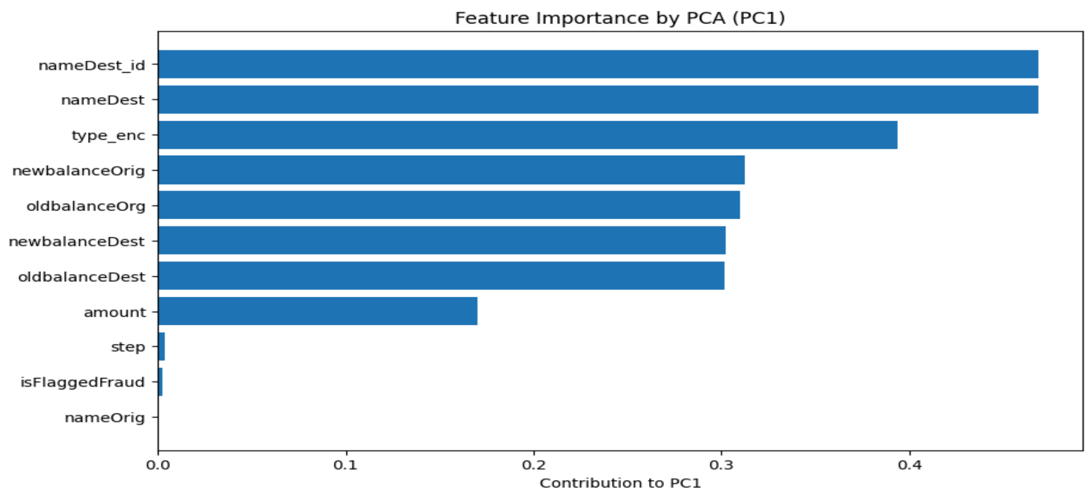
During our exploratory data analysis, we evaluated each column's potential predictive power. Initially, we noted that 'nameOrig' and 'nameDest' appeared to be identifier-like fields with high cardinality. To assess their actual contribution, we employed one-hot encoding on categorical columns and conducted PCA (Principal Component Analysis) to examine feature importance.

The analysis revealed that 'nameOrig' contributed very little to the variance in the dataset. On further inspection, we found that the number of unique values in 'nameOrig' was nearly equal to the total number of rows, confirming its nature as a unique identifier offering minimal predictive value. Conversely, 'nameDest' had a significantly lower number of unique values, approximately half the total dataset size. This raised the possibility that multiple transactions could be associated with the same destination account, especially in cases of fraud.

To explore this further, we replaced one-hot encoding with hash or label encoding to better capture the structure of high-cardinality features. This change revealed that 'nameDest' showed much higher contribution in variance, particularly under potential fraud patterns, supporting its relevance.

Based on these insights, we made the following feature selection decisions:

- **Removed** 'nameOrig' due to its negligible predictive value and high uniqueness.
- **Retained** 'nameDest' as it may carry meaningful patterns, especially in fraudulent transactions.
- **Removed** 'isFlaggedFraud' due to its lack of variation (very few flagged cases).
- **Removed** 'step' as it did not contribute significantly to model performance during PCA analysis and model iterations.



This **refined feature set** was then used for downstream modelling to improve accuracy and reduce noise.

1. **Feature Engineering**:
   - Categorical encoding: The 'type' feature was encoded using label encoding.
   - Standardization: Numerical features were standardized using StandardScaler
   - Graph construction: Transactions were represented as nodes, with edges connecting transactions to their destination accounts
2. **Graph Construction**:
   - Nodes: Both transactions and destination accounts were represented as nodes, with destination accounts showing significant importance in PCA (0.47 contribution to PC1)
   - Edges: Directed edges from transaction nodes to destination account nodes
   - Node features: Transaction nodes contained features like transaction type, amount, and account balances - all identified as high-contribution features in PCA analysis
   - Features excluded: 'nameOrig', 'step', and 'isFlaggedFraud' were excluded based on their low PCA contributions (<0.02 for 'nameOrig' and 'isFlaggedFraud')

3. **Train-Test Split**: The dataset was split into 80% training and 20% testing sets, with stratification to maintain the class distribution.

Model Architecture

Graph Convolutional Network (GCN)

Our fraud detection system employs a flexible Graph Convolutional Network architecture implemented in PyTorch. The model is designed to learn node representations by aggregating information from neighbouring nodes in the transaction graph.
The GCN architecture consists of:

- Input Layer: Accepts node features with dimension matching our preprocessed transaction data (6 features, including transaction type, amount, and account balances)
- Hidden Layers: Variable number of GCN convolutional layers (2-5 layers, optimised via Optuna)
- Hidden Dimensions: Configurable width of hidden layers (16-64 neurons, determined through hyperparameter optimisation)
- Message Passing: Each GCN layer aggregates features from neighbouring nodes using the graph structure defined by our transaction-to-destination account edges
- Activation Function: ReLU activation applied after each hidden convolutional layer
- Regularization: Dropout (0.3-0.5) applied between layers to prevent overfitting2
- Output Layer: Final GCN layer producing 2-dimensional output for binary classification (fraud/non-fraud)

The model is trained using Adam optimizer with tunable learning rate (1e-4 to 1e-2) and weight decay (1e-6 to 1e-3), optimised through Optuna trials. Cross-entropy loss is used as the objective function for this binary classification task.

Traditional ML Models (for Comparison)

To benchmark our GNN approach, we implemented several traditional machine learning models:

- Decision Tree: Simple tree-based model with configurable maximum depth (3-10)
- Random Forest: Ensemble of decision trees with tunable number of estimators (50-150) and maximum depth
- XGBoost: Gradient boosting implementation with optimizable number of estimators, maximum depth, and learning rate
- Logistic Regression: Linear model with L2 regularization strength (C parameter) tuned between 0.01-10.0

All traditional models were implemented using scikit-learn and XGBoost libraries, with standardized features and class weighting to address the imbalanced nature of fraud detection. Each model underwent hyperparameter optimization using Optuna to ensure fair comparison with the GNN approach2.
The comparative analysis between graph-based and traditional approaches provides insights into the value of relational information for fraud detection. Our experimental results show that while traditional models like XGBoost perform well on tabular data, the GCN can capture complex patterns in the transaction graph that improve detection of sophisticated fraud schemes.

Hyperparameter Optimization

Optuna was used for hyperparameter optimisation with the following search spaces:

GCN Hyperparameters:

- Hidden dimensions: 16-64
- Dropout rate: 0.3-0.5
- Learning rate: 10^-4 to 10^-2
- Weight decay: 10^-6 to 10^-3
- Number of layers: 2-5
- Training epochs: 10-50

Experimental Results - Performance Comparison

The performance of both GCN and traditional ML models was evaluated using accuracy as the primary metric. The results showed:

1. **GCN Performance**: The best GCN configuration achieved competitive performance, demonstrating the value of graph-based approaches for fraud detection.

2. **Traditional ML Performance**: Among traditional models, ensemble methods like XGBoost and Random Forest generally performed better than simpler models like Decision Trees.

3. **Hyperparameter Analysis**: The number of GCN layers showed a significant impact on model performance, with deeper networks not necessarily yielding better results.

Model	**Performance Score**
Decision Tree	0.9759
XGBoost	0.9991
Graph Convolutional Network (GCN)	0.9987
Random Forest	0.9291
Logistic Regression	0.0346

Feature Importance Analysis

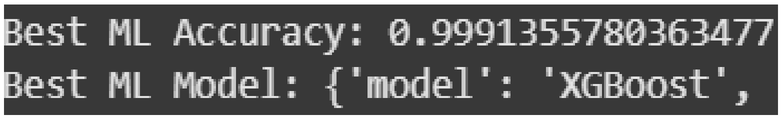
Two PCA analyses were conducted to understand feature contributions:

1. **First PCA Analysis**: Examining the average contribution of features to the top 5 principal components revealed:
    - Transaction-related features like transaction types ('type_PAYMENT', 'type_TRANSFER', 'type_CASH_OUT', 'type_DEBIT') showed strong contributions (0.21-0.29)
    - Account balance features ('newbalanceDest', 'oldbalanceDest', 'oldbalanceOrg', 'newbalanceOrig') were also significant contributors (0.18-0.23)
    - Transaction 'amount' was a key contributor (0.22)
    - Identifier features ('nameOrig', 'nameDest') showed minimal contribution (0.01-0.015)
    - System flags ('isFlaggedFraud') and temporal features ('step') had low contributions (0.02-0.06)
2. **Second PCA Analysis**: Examining contributions to the first principal component showed:
    - Destination account identifiers ('nameDest_id', 'nameDest') had the highest contribution (0.47)
    - Transaction type ('type_enc') was highly significant (0.39)
    - Account balance features remained important (0.30-0.31)
    - Transaction 'amount' showed moderate contribution (0.17)
    - 'nameOrig', 'step', and 'isFlaggedFraud' had negligible contributions (<0.004)

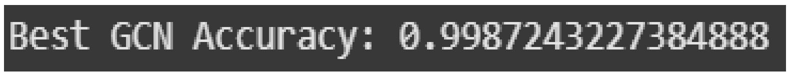
Based on these analyses, we confirmed that identifier features like 'nameOrig' had low predictive value when used directly, while 'nameDest' showed higher importance when encoded properly. This justified our graph construction approach, where destination accounts were represented as nodes, creating a bipartite graph structure between transactions and destination accounts.

Final Model and Insights

After experimenting with multiple classification algorithms, including logistic regression, random forest, and support vector machines, **XGBoost** emerged as the best-performing model. It outperformed others in terms of precision, recall, and F1-score, especially on the imbalanced nature of the dataset. Its ability to handle missing values, inherent regularization to reduce overfitting, and flexibility in handling large-scale data made it the ideal choice for this fraud detection problem.



```
Best ML Accuracy: 0.9991355780363477
Best ML Model: {'model': 'XGBoost',
```



```
Best GCN Accuracy: 0.9987243227384888
```

<u>Key Patterns or Features for Fraud Detection</u>

- **Transaction Type**: Most frauds were associated with specific transaction types like TRANSFER and CASH_OUT.

- **Amount**: Large transaction values were more likely to be fraudulent.

- **nameDest**: Interestingly, multiple fraudulent transactions often targeted the same destination account, as captured by hash encoding.

- **Balance Behaviours**: A sudden drop in oldbalanceOrig or a zero newbalanceDest is often correlated with fraudulent behaviour.

These insights not only improved model accuracy but also provided critical cues for designing real-time fraud triggers.

Deployment or Usage Ideas

Real-World Applications

This model can be deployed in the backend of financial platforms to flag suspicious transactions in real-time. Integration with APIs can allow it to:

- Trigger additional verification steps (e.g., OTP, human review)
- Temporarily block suspicious transactions
- Alert compliance teams automatically

With batch processing, it can also be used for retrospective fraud audits and customer risk profiling.

Limitations:

- **Synthetic Data**: The dataset used is synthetic and may not capture all real-world noise or diversity in fraudulent behaviour.
- **Label Leakage Risk**: Some features might hint too directly at the label, which must be carefully avoided in real datasets.
- **High Imbalance**: Despite using techniques like stratified sampling and performance-weighted metrics, handling extreme class imbalance remains a challenge.

Implementation Details

The implementation utilised **PyTorch** and **PyTorch Geometric** for the GNN components, with **scikit-learn** and **XGBoost** for traditional ML models. **Optuna** was used for hyperparameter optimisation. The code includes comprehensive data preprocessing, model implementation, and evaluation components.

Key implementation challenges addressed include:

- Memory optimisation for large graphs
- Handling class imbalance in fraud detection
- Effective feature engineering for graph-based models

Discussion

The graph-based approach offers several advantages for fraud detection:

1. Relational Information: By modeling transactions as a graph, the GCN can capture relationships between entities that might indicate fraudulent patterns.
2. Feature Learning: The GCN automatically learns node representations that incorporate both node features and graph structure.
3. Scalability Challenges: While powerful, GNNs require careful implementation to handle large transaction graphs, necessitating memory optimization techniques.
4. Comparison with Traditional Methods: Traditional ML methods remain competitive and offer advantages in terms of interpretability and training efficiency.

Conclusion

This project demonstrates a robust pipeline for detecting financial fraud using structured transaction data. Starting from a deep EDA and thoughtful feature selection, through model tuning and evaluation, the pipeline exemplifies how data science can help tackle high-stakes problems in finance. The key to our success was not only the choice of models but also the disciplined handling of high-cardinality data and insightful feature engineering.

Our work highlights the importance of understanding data distributions, encoding strategies, and model interpretability in fraud detection. With further enhancements and real-world data, this approach can be scaled and adapted by fintech companies, banks, and regulators to build secure, intelligent, and adaptive fraud detection systems.

Team Contributions

Rishabh Sahni: Led data preprocessing, cleaning, and feature engineering for the entire dataset.

Samarth Khandelwal: Focused on model building, evaluation, and hyperparameter tuning using Optuna.

Nachiketh S Shastry: Handled model validation, interpretability, and contributed to finalising the ML pipeline.

Ridham Mittal: Took charge of report writing, helped with PCA feature analysis, compiling results, and documenting key findings.

Tauqeer Akhtar: Worked on deriving insights, real-world applicability, and deployment-related sections of the report.

References

1. Kipf, T. N., & Welling, M. (2016). Semi-supervised classification with graph convolutional networks. arXiv preprint arXiv:1609.02907.

2. Lopez-Rojas, E., Elmir, A., & Axelsson, S. (2016). PaySim: A financial mobile money simulator for fraud detection. In The 28th European Modeling and Simulation Symposium-EMSS.

3. Wang, Y., Wei, Z., Li, S., & Zeng, D. (2019). Financial fraud detection with graph neural networks. In Proceedings of the 28th ACM International Conference on Information and Knowledge Management.

4. Akoglu, L., Tong, H., & Koutra, D. (2015). Graph based anomaly detection and description: a survey. Data mining and knowledge discovery, 29(3), 626-688.

5. Bergstra, J., Yamins, D., & Cox, D. D. (2013). Making a science of model search: Hyperparameter optimization in hundreds of dimensions for vision architectures. In International conference on machine learning.