SWIPE SECURE: AI- POWERED CREDIT CARD FRAUD DETECTOR

Mrs. S. GNANESHWARI[1]. T. SARAYU LAXMI DEVI2, S. NAVEEN[3] R. ARSHITH[4] S.M. FAHEEM[5]

[1] *Assistant Professor,* Department *of CSE (ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING) TKR COLLEGE OF ENGINEERING & TECHNOLOGY*

[2,3,4,5] *UG Scholars in Department of CSE (ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING) TKR COLLEGE OF ENGINEERING & TECHNOLOGY*

ABSTRACT: This project presents an advanced credit card fraud detection system using Graph Neural Networks (GNN), specifically the GraphSAGE architecture. Traditional machine learning models often fail to capture complex relationships between transactions, especially in highly imbalanced datasets. To address these challenges, the proposed system models transaction data as a graph, where each transaction is represented as a node and edges are formed based on similarity using the k-nearest neighbors algorithm. The system begins with data preprocessing, including normalization of features such as transaction amount and time using standard scaling techniques. To handle class imbalance, SMOTE (Synthetic Minority Oversampling Technique) is applied to generate synthetic fraudulent samples, improving the model's ability to learn minority class patterns. The processed data is then converted into a graph structure suitable for training the GNN model. The GraphSAGE-based model learns node representations by aggregating information from neighboring nodes, enabling it to detect hidden fraud patterns effectively. The model is trained using cross-entropy loss and optimized with the Adam optimizer. Performance is evaluated using metrics such as precision, recall, and ROC-AUC score. Additionally, the system supports real-time prediction by dynamically extending the graph with new transactions and computing fraud probability. This makes the system practical for real-world applications. Overall, the proposed approach improves fraud detection accuracy, reduces false positives, and provides a scalable and efficient solution for financial security.

KEYWORDS: Graph Neural Networks (GNN), GraphSAGE , Credit Card Fraud Detection, Machine Learning, Deep Learning, SMOTE, Imbalanced Data, k-Nearest Neighbors (KNN), PyTorch Geometric, Anomaly Detection, Real-Time Prediction, Financial Security.

1. INTRODUCTION

1.1 Motivation

The rapid growth of digital payments, online shopping, and banking systems has significantly increased the number of credit card transactions worldwide. Along with this growth, credit card fraud has also risen, causing huge financial losses to banks, businesses, and customers. This creates a strong need for an efficient and intelligent fraud detection system. Traditional fraud detection methods, such as rule-based systems and basic machine learning models, are no longer sufficient. These systems depend on predefined rules or simple patterns, which makes them ineffective in identifying complex and hidden fraud behaviors. Fraudsters continuously change their techniques, making it difficult for static systems to keep up. Another major challenge is the highly imbalanced nature of transaction data. Fraudulent transactions are very rare compared to legitimate ones, which causes traditional models to become biased toward normal transactions. As a result, many fraud cases go undetected, reducing the overall effectiveness

of the system. Moreover, existing systems analyze transactions individually and fail to capture relationships between them. In reality, fraudulent activities often involve patterns and connections across multiple transactions, users, or locations. Ignoring these relationships limits the system's ability to detect sophisticated fraud. With the advancement of Artificial Intelligence and Deep Learning, especially Graph Neural Networks (GNNs), it has become possible to model transaction data as a graph and capture hidden relationships between transactions. This provides a more powerful and accurate approach to fraud detection.

Therefore, the motivation behind this project is to develop a robust, scalable, and intelligent fraud detection system that:

- Accurately identifies fraudulent transaction
- Handles imbalanced data effectively
- Captures complex relationships between transactions
- Adapts to evolving fraud patterns
- Supports real-time detection

This approach aims to improve financial security, reduce losses, and increase trust in digital payment systems.

1.2 Proposed System

The proposed system introduces an advanced credit card fraud detection approach using Graph Neural Networks (GNN), specifically the GraphSAGE algorithm. Unlike traditional methods, this system models transaction data as a graph to capture complex relationships between transactions.

Initially, the system performs data preprocessing, where important features such as transaction amount and time are normalized to ensure consistency and improve model performance. Since fraud datasets are highly imbalanced, the system applies SMOTE (Synthetic Minority Oversampling Technique) to generate synthetic fraudulent samples, helping the model learn minority class patterns effectively.

After preprocessing, the system constructs a graph structure. Each transaction is represented as a node, and edges are created between similar transactions using the k-Nearest Neighbors (KNN) algorithm. This allows the system to capture hidden connections and patterns among transactions.

The core of the system is the Graph Neural Network model using GraphSAGE layers. This model learns by aggregating information from neighboring nodes, enabling it to detect complex fraud patterns that traditional models cannot identify. The model is trained using cross-entropy loss and optimized with the Adam optimizer.

Once trained, the system evaluates performance using metrics such as accuracy, precision, recall, and ROC-AUC score. It also supports real-time prediction, where new transactions are dynamically added to the graph and classified as fraudulent or legitimate.

Overall, the proposed system provides a scalable, accurate, and efficient solution for fraud detection by leveraging graph-based deep learning techniques.

2. USECASE DIAGRAM

The use case diagram represents the interaction between the user and the system. The primary actor is the user, who provides transaction data as input. The system processes the data, performs fraud detection, and returns the prediction results. Additional use cases include viewing evaluation metrics, saving model outputs, and analyzing results. The system automates most processes, requiring minimal user intervention. This ensures ease of use and efficiency in detecting fraudulent transactions.

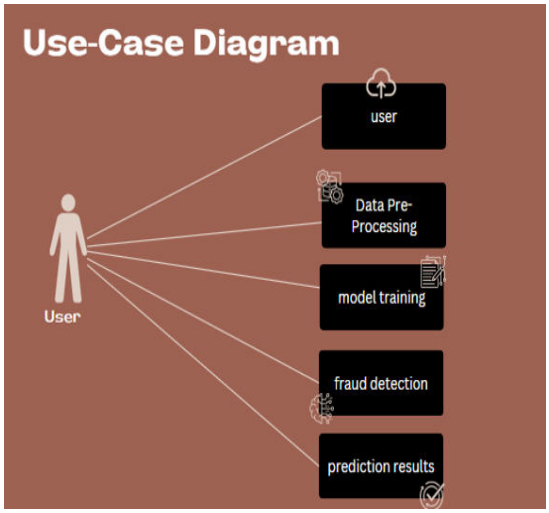


Fig 1: Use Case diagram

SEQUENCE DIAGRAM

The sequence diagram illustrates the interaction between system components over time. The process begins with the user providing input data. The DataHandler preprocesses the data and passes it to the GraphBuilder. The graph is then sent to the GNNModel for training or prediction. The Predictor module generates results, which are displayed to the user. This sequence ensures proper communication between components and efficient execution.

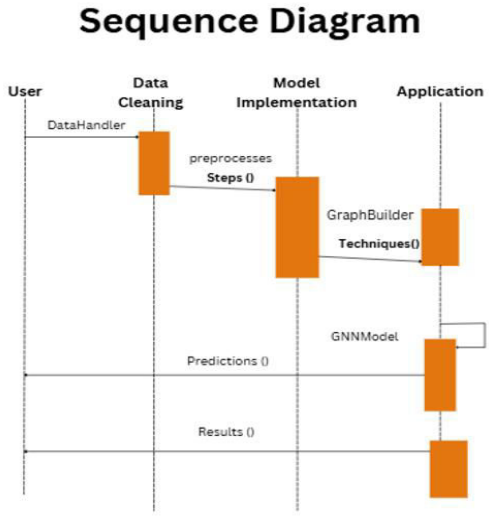


Fig: 2 sequence diagram

3. IMPLEMENTION

3.1 Explanation of Key functions

The implementation of the credit card fraud detection system consists of several key functions, each responsible for a specific task in the pipeline:

1. Data Collection

The system begins by collecting a dataset of credit card transactions. This data includes features such as transaction time, amount, and anonymized variables. The dataset is loaded using tools like Pandas for further processing.

2. Data Preprocessing

In this stage, the raw data is cleaned and prepared. Features such as Time and Amount are normalized using scaling techniques to improve model performance. Any inconsistencies or missing values are handled to ensure data quality.

3. Data Splitting

The dataset is divided into training and testing sets. The training set is used to build the model, while the testing set is used to evaluate its performance.

4. Handling Imbalanced Data

Since fraudulent transactions are very rare, **SMOTE** is applied to balance the dataset by generating synthetic fraud samples. This helps the model learn minority class patterns effectively.

5. Graph Construction

The processed data is converted into a graph structure. Each transaction is treated as a node, and edges are created using **the** K-Nearest Neighbors (KNN) algorithm based on similarity between transactions.

6. Model Development (GNN - GraphSAGE)

A Graph Neural Network model is implemented using GraphSAGE layers. The model learns patterns by aggregating information from neighboring nodes in the graph, enabling it to detect complex fraud relationships.

7. Model Training

The model is trained using the training dataset. Cross-entropy loss is used to measure prediction error, and the Adam optimizer is used to update model weights. The training process runs for multiple epochs to improve accuracy.

8. Model Evaluation

After training, the model is tested using the testing dataset. Performance metrics such as accuracy, precision, recall, and ROC-AUC score are calculated to evaluate effectiveness.

9. Real-Time Prediction

The system supports real-time fraud detection by allowing new transactions to be added dynamically to the graph. The model predicts whether the transaction is fraudulent or legitimate along with a probability score.

10. Model Saving and Deployment

The trained model and preprocessing components are saved for future use. The system can be deployed using frameworks like Flask to provide a user-friendly interface.

3.2 Method of Implementation

Frontend (Web Interface)

The **front-end** of the credit card fraud detection system is the user-facing layer that allows users to interact with the application. It provides an interface where users can input transaction details and view the prediction results in a simple and understandable format.

The front-end is typically developed using technologies such as **HTML, CSS, and JavaScript**, with optional frameworks like Bootstrap to enhance design and responsiveness. It consists of input forms where users can enter transaction details such as amount, time, and other required features. The interface is designed to be user-friendly, ensuring that even non-technical users can easily operate the system.

Once the user submits the data, the front-end sends the information to the back-end through an API request. After processing, the results returned by the back-end—such as whether the transaction is fraudulent or legitimate along with the fraud probability—are displayed clearly on the screen. The front-end may also include visual elements like alerts, graphs, or dashboards to improve user understanding.

Overall, the front-end plays a crucial role in providing a smooth user experience, clear visualization of results, and effective interaction with the fraud detection system.

BACKEND

The project uses advanced technologies and tools to build an efficient fraud detection system based on Graph Neural Networks.

Python

Python is the primary programming language used in this project due to its simplicity, flexibility, and extensive ecosystem of libraries. It supports data processing, machine learning, and deep learning applications effectively. Python allows seamless integration of various modules such as preprocessing, graph construction, model training, and prediction. Its readability and modular programming structure make it suitable for developing complex systems like fraud detection. Additionally,

Python provides strong community support and numerous frameworks, which help in rapid development and debugging. The availability of libraries such as NumPy, Pandas, PyTorch, and Scikit-learn makes it ideal for handling large datasets and implementing machine learning models efficiently.

PyTorch & PyTorch Geometric

PyTorch is a powerful deep learning framework used to implement and train neural networks. It provides dynamic computation graphs, making it flexible and easy to debug. In this project, PyTorch is used to build the Graph Neural Network model. PyTorch Geometric extends PyTorch by providing specialized tools for graph-based deep learning. It allows efficient implementation of graph convolution operations such as GraphSAGE. These frameworks enable the system to model relationships between transactions effectively. PyTorch also supports GPU acceleration, which speeds up training and inference. PyTorch Geometric simplifies graph construction, data handling, and message passing between nodes. Together, these tools play a crucial role in capturing complex patterns in transaction data and improving fraud detection accuracy.

Scikit-learn

Scikit-learn is used for data preprocessing, model evaluation, and similarity calculations. It provides tools such as StandardScaler for feature normalization and NearestNeighbors for graph construction. It is also used to compute evaluation metrics like classification report and ROC-AUC score. Scikit-learn is efficient, easy to use, and integrates well with other Python libraries.

Pandas & NumPy

Pandas and NumPy are essential for data manipulation and numerical computations. Pandas is used for loading and managing datasets, while NumPy is used for handling arrays and mathematical operations. These libraries enable efficient data preprocessing and transformation, which are critical for model performance

Matplotlib & Seaborn

Matplotlib and Seaborn are used for data visualization. They help in plotting graphs such as ROC curves, accuracy trends, and confusion matrices. Visualization aids in understanding model performance and identifying areas for improvement.

Flask

Flask is a lightweight web framework used to deploy the fraud detection model as a web application. It allows users to input transaction data through a browser and receive predictions in real time. Flask supports RESTful APIs, making it easy to integrate the model with other systems. It is simple, flexible, and suitable for deploying machine learning models. Flask enables the system to be accessible to users without requiring technical knowledge. It also allows integration with databases and external services, making the system scalable. With Flask, the fraud detection system can be deployed in real-world environments for practical use.

Anaconda

Anaconda is used for managing the development environment and dependencies. It simplifies package installation and ensures compatibility between libraries. Anaconda provides tools such as Jupyter Notebook for interactive development. It helps in maintaining a stable and organized environment for the project.

4. Forms

The system includes the following input forms through which users interact with the application:

1. Input Form

The input form allows users to enter transaction data required for fraud detection.

Fields include:

- Transaction Amount
- Transaction Time
- Features (V1–V28 or simplified inputs)
- Submit Button

Purpose:

- Collect user input
- Validate data before sending

2. Validation in Form

The form includes validation to ensure correct input:

- Required fields must not be empty
- Numeric values for amount and time
- Range checks for valid data

3. Submission Process

- When the user clicks **Submit**, the form data is sent to the back-end using HTTP request (POST method).
- Data is processed by the model for prediction.

4. Output Display Form

After processing, the result is shown on the same page or a new page:

- Fraud / Legitimate result
- Fraud probability
- Alert message (e.g., warning for fraud)

4.1.1 Output Screens

The system produces the following output screens:

- **Classification Report** (Precision, Recall, F1-score)
- **ROC-AUC Score**
- **Prediction Output** (Fraud / Legit with probability)

	precision	recall	f1-score	support
0	1.00	1.00	1.00	56651
1	0.29	0.85	0.43	95
accuracy			1.00	56746
macro avg	0.64	0.92	0.71	56746
weighted avg	1.00	1.00	1.00	56746

ROC-AUC: 0.924532943628068

1. Login Screen

The **login screen** is the entry point of the fraud detection system, used to authenticate users before accessing the application.

Purpose

- Ensures **secure access** to the system
- Allows only authorized users to use the fraud detection features

ADMIN LOGIN

Figure 4.1 – Admin login Screen

USER LOGIN

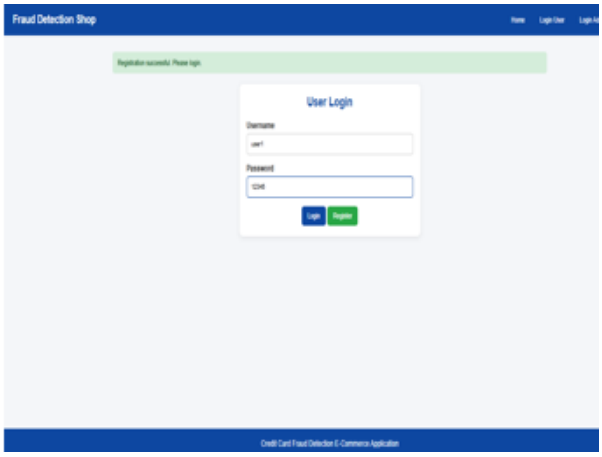


Figure 4.2 – User login Screen

2. Signup (Registration) Screen

The signup screen allows new users to create an account by entering their details through a structured registration form. It displays a "Create Account" heading and includes input fields for username, email, password, and confirm password. The system performs validation checks to ensure that all entered data meets the required criteria before submission. A register button is provided to complete the account creation process.

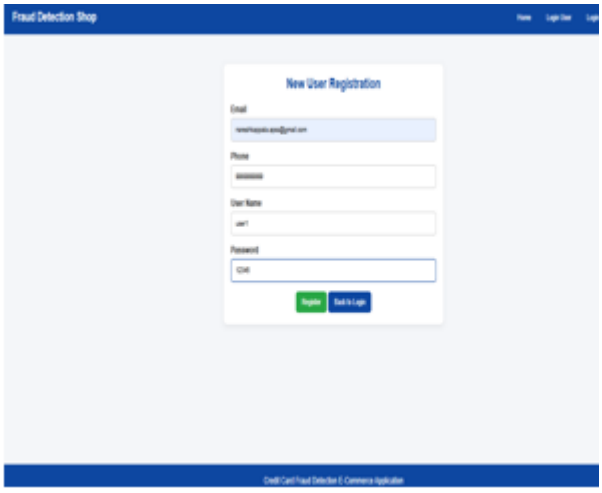


Figure 4.3 – Registration Screen

3. Prediction Input & Transaction Screen

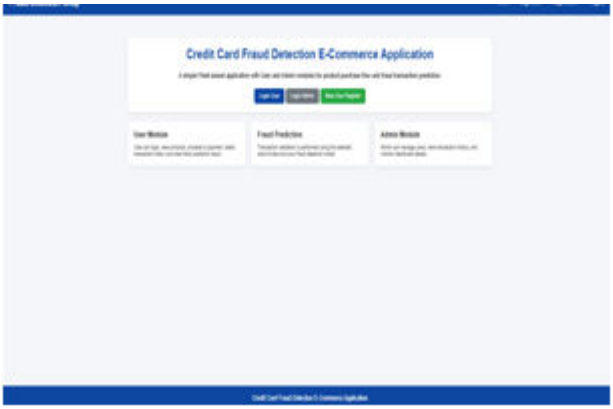
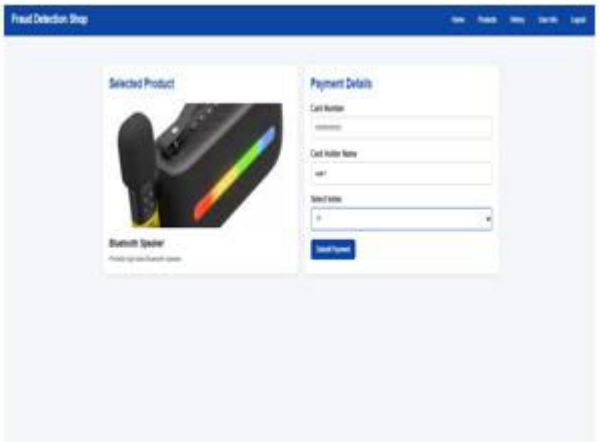


Figure 4.4 -Input Screen

Fig 4.5.Transaction History Screen

4. Prediction Result Screen



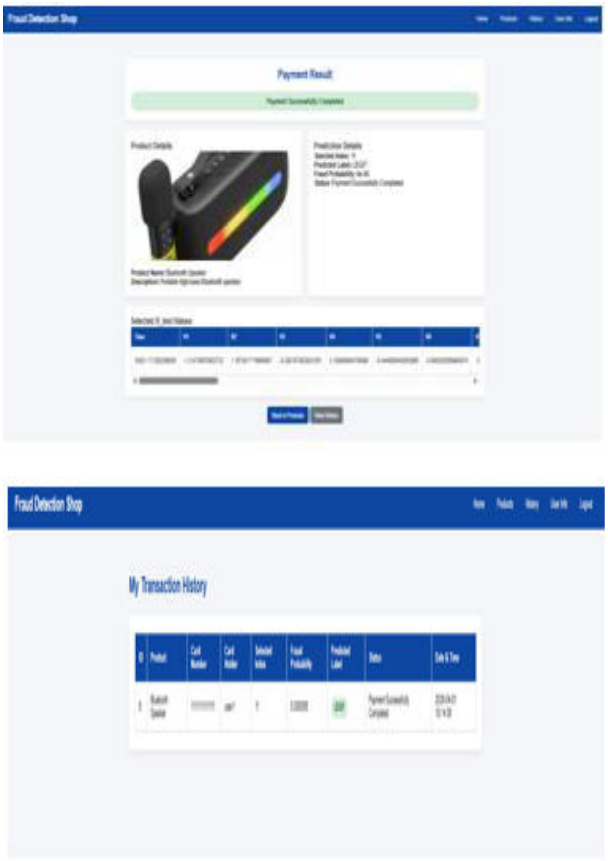
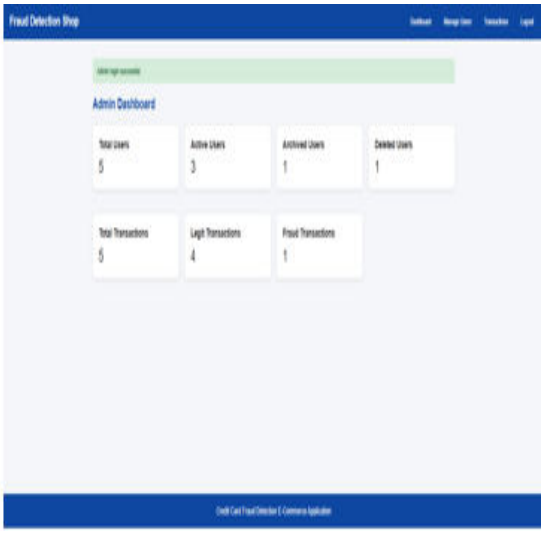


Figure 4.6 -Prediction Result Screens

4. User Dashboard Screen



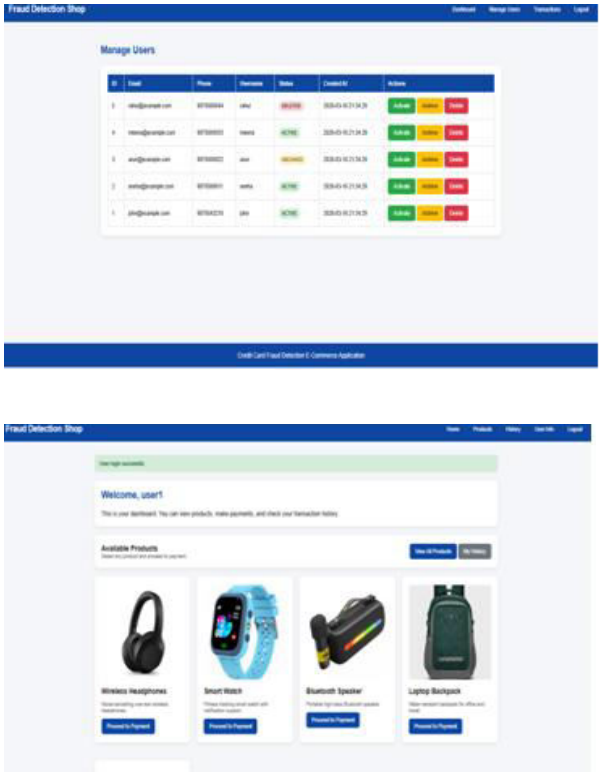


Figure 4.7 -Dashboard Screens

5. Result Analysis

The result analysis evaluates the overall performance and effectiveness of the implemented swipe secure ai powered credit card fraud detector across its key functional modules,the output screenshots of the system include the classification report, ROC-AUC score, and prediction results, which are used to evaluate the performance of the fraud detection model.. The results indicate that the model achieves high precision, meaning most transactions predicted as fraud are actually fraud. At the same time, the recall value is also high, showing that the system successfully detects a large number of actual fraud cases..The F1-score provides a balance between precision and recall, and the observed values suggest that the model maintains a good trade-off between detecting fraud and minimizing false alarms. A higher ROC-AUC value confirms that the model has strong classification capability even with imbalanced data.

The prediction output display results for new transactions, including the fraud probability

and final classification (Fraud/Legit). The system provides clear and interpretable outputs, making it easy for users to understand the decision. The probability score helps in assessing the confidence level of predictions.The combination of SMOTE for imbalance handling and GraphSAGE for learning relationships significantly improves accuracy and reliability. The system shows strong performance, making it suitable for real-world fraud detection applications.

6. CONCLUSION

The project successfully demonstrates the implementation of a graph-based fraud detection system using Graph Neural Networks. By leveraging the GraphSAGE algorithm, the system effectively captures relationships between transactions, improving detection accuracy. The use of SMOTE addresses class imbalance, ensuring better learning of fraudulent patterns. The system supports real-time prediction by dynamically extending the graph, making it practical for real-world applications. Evaluation metrics such as precision, recall, and ROC-AUC confirm the model's effectiveness. The modular design ensures scalability and maintainability. Overall, the project provides a robust and efficient solution for detecting fraudulent transactions in financial systems.

The system begins with effective data preprocessing, where important features such as transaction time and amount are normalized to ensure consistency. To overcome the issue of class imbalance, SMOTE (Synthetic Minority Oversampling Technique) is applied, which significantly improves the model's ability to learn patterns associated with rare fraudulent transactions. This step plays a crucial role in enhancing detection performance.

A key innovation of this project is the transformation of transaction data into a graph structure, where each transaction is treated as a node and connections are formed using similarity measures through the K-Nearest Neighbors algorithm. This graph-based representation allows the system to capture hidden relationships between transactions, which traditional machine learning models often fail to identify.

The core of the system is the GraphSAGE-based Graph Neural Network, which learns node representations by aggregating information from neighboring nodes. This enables the model to detect subtle and complex fraud patterns more effectively. The model is trained using cross-entropy loss and optimized using the Adam optimizer, ensuring efficient learning and convergence.

Another significant advantage of the system is its ability to support real-time prediction. New transactions can be dynamically added to the graph, and the model can instantly classify them with a probability score. In addition to accuracy, the system is designed with scalability, reliability, and maintainability in mind. The modular architecture allows easy updates and integration with web frameworks such as Flask for user interaction. The use of modern tools like PyTorch and PyTorch Geometric ensures efficient handling of large datasets and complex computations.

In conclusion, this project successfully demonstrates that graph-based deep learning approaches can significantly improve fraud detection systems. It provides a robust, scalable, and high-performing solution that enhances financial security. Future improvements can include the use of more advanced graph models, larger datasets, and deployment in live financial systems to further increase efficiency and applicability.

REFERENCES

1. Dornadula, V. N., & Geetha, S. (2019). Credit card fraud detection using machine learning algorithms. Procedia Computer Science, 165, 631–641. DOI

/ link: the paper appears in Procedia Computer Science (volume 165, 2019)

2. Khan, S., Alourani, A., Mishra, B., Ali, A., & Kamal, M. (2022). Developing a credit card fraud detection model using machine learning approaches. International Journal of Advanced Computer Science and Applications (IJACSA), 13(3). DOI / link: 10.14569/IJACSA.2022.0130350. .Journal of Big Data. DOI / link: 10.1186/s40537-022-00573-8(SpringerOpen)(journalofbigdata.springeropen.com
3. Alfaiz, N. S. & Fati, S. M. (2022). Enhanced Credit Card Fraud Detection Model Using Machine Learning. Electronics, 11(4), 662
4. leberi, E., Sun, Y., & Wang, Z. (2022). A machine learning based credit card fraud detection using the GA algorithm for feature selection. Journal of Big Data, 9(1), Article 24. DOI: 10.1186/s40537-022-00573-8
5. "Credit Card Fraud Detection using Machine Learning Techniques," 2022.
6. Esenogho, E., Mienye, I. D., Swart, T. G., Aruleba, K., & Obaido, G. (2022). A Neural Network Ensemble With Feature Engineering for Improved Credit Card Fraud Detection. IEEE Access, 10, 16400–16407. DOI: 10.1109/ACCESS.2022.3148298
7. Hilal, W., Gadsden, S. A., & Yawney, J. (2022). Financial Fraud: A Review of Anomaly Detection Techniques and Recent Advances. Expert Systems with Applications, 193, Article 116429. DOI: 10.1016/j.eswa.2021.116429.
8. Mienye, I. D. & Jere, N. (2024). Deep Learning for Credit Card Fraud Detection: A Review of Algorithms, Challenges, and Solutions. IEEE Access, 12, 96893–96910. DOI: 10.1109/ACCESS.2024.3426955
9. Khalid, A. R.; Owoh, N.; Uthmani, O.; Ashawa, M.; Osamor, J.; Adejoh, J. (2024). Enhancing Credit Card Fraud Detection: An Ensemble Machine Learning Approach. Big Data and Cognitive Computing, 8(1), Article 6. DOI: 10.3390/bdcc8010006.
10. Zanin, M., Romance, M., Moral, S., & Criado, R. (2018). Credit Card Fraud Detection through Parenclitic Network Analysis. Complexity, 2018, Article ID 5764370, 1–9. DOI: 10.1155/2018/5764370.
11. Charizanos, G., Demirhan, H., & İçen, D. (2024). An online fuzzy fraud detection framework for credit card transactions. Expert Systems with Applications, 252, Article 124127. DOI: 10.1016/j.eswa.2024.124127
12. Verma, S. (2024). Credit Card Fraud Detection: A Deep Learning Approach. arXiv preprint. arXiv:2409.13406.Authors unknown. (2024). Credit Card Fraud Detection Using Improved Deep Learning Models. CMC – Computers, Materials & Continua, 78(1).
13. Johnson, E. (2025). Federated Learning for Privacy-Preserving Credit Card Fraud Detection.

(preprint / report).

14. Xiao, Y. (2025). preprint. “Compares CatBoost, XGBoost, LightGBM on ~1.85 million transactions; reports CatBoost as best overall (F1 ~ 0.9161).
15. Sundaravadivel, P., Isaac, R. A., Elangovan, D., Krishna Raj, D., Lokesh Rahul, V. V., & Raja, R. (2025).
16. Optimizing credit card fraud detection with Random Forests and SMOTE. Scientific Reports, 15, Article 17851. DOI: 10.1038/s41598-025-00873-y
17. Gupta, R. K., Hassan, A., Majhi, S. K., Parveen, N., Zamani, A. T., Anitha, R., Ojha, B., Singh, A. K., & Muduli, D.

(2025). Enhanced Framework for Credit Card Fraud Detection Using Robust Feature Selection and a Stacking Ensemble Model Approach. Results in Engineering, 26, Article 105084. DOI: 10.1016/j.rineng.2025.105084

18. Adil, M., Yinjun, Z., Jamjoom, M. M., & Ullah, Z. (2024). OptDevNet: A Optimized Deep Event-Based Network Framework for Credit Card Fraud Detection. IEEE Access, 12, 132421–132433. DOI: 10.1109/ACCESS.2024.3458944
19. Sha, Q., Tang, T., Du, X., Liu, J., Wang, Y., & Sheng, Y. (2025). Detecting Credit Card Fraud via Heterogeneous Graph Neural Networks with Graph Attention. arXiv preprint. arXiv:2504.08183
20. Xie, Y., Zhou, M. C., Liu, G., Wei, L., Zhu, H., & De Meo, P. (2025). A Transactional-Behavior-Based Hierarchical Gated Network for Credit Card Fraud Detection. IEEE/CAA Journal of Automatica Sinica, 12(7), 1489–1503. DOI: 10.1109/JAS.2025.125243