FRAUD TRANSACTION DETECTION

A project report submitted to ICT Academy of Kerala

in partial fulfillment of the requirements

for the certification of

CERTIFIED SPECIALIST

IN

DATA SCIENCE & ANALYTICS

submitted by

ANJANA C

HARITHA K V

ADITHYA KRISHNAN C

ICT ACADEMY OF KERALA
THIRUVANANTHAPURAM, KERALA, INDIA
JAN 2025

List of Figures

List of Abbreviations

EDA	**Exploratory Data Analysis**
KNN	**K-Nearest Neighbor**
RF	**Random Forest**
LGBM	**Light Gradient Boosting Machine**
DT	**Decision Tree**
GDPR	**General Data Protection Regulation**
SMOTE	**Synthetic Minority Oversampling Technique**
XGBoost	**Extreme Gradient Boosting**

Table of Contents

1.Abstract

This project report presents the development of a Fraud Transaction Detection System aimed at identifying fraudulent activities in financial transactions. By leveraging machine learning algorithms and advanced data analytics, the system ensures accurate detection and prevention of potential losses. The project encompasses exploratory data analysis (EDA), web deployment for individual users, and bulk transaction analysis for organizational use. The solution is designed to enhance security and trust in financial systems, integrating Python-based tools and a user-friendly web interface.

2. Problem Definition

2.1 Overview

Fraudulent transactions pose a significant threat to the integrity of financial systems, resulting in considerable monetary and reputational losses for individuals and institutions. Detecting fraudulent activities is complex due to the evolving tactics used by fraudsters and the massive volume of transactions processed daily. Financial fraud not only undermines consumer trust but also imposes substantial costs on businesses and regulatory bodies, necessitating robust and scalable detection systems. The rise of digital banking and e-commerce has further exacerbated the issue, creating a critical need for innovative solutions that can handle large datasets in real-time.

2.2 Problem Statement

The objective of this project is to develop a robust fraud detection system capable of identifying and mitigating fraudulent financial transactions in real-time, ensuring the safety and security of stakeholders. This system aims to overcome limitations in traditional detection methods, such as high false positive rates and inability to adapt to evolving fraudulent tactics. The goal is to provide an efficient and scalable solution that can protect both individual users and financial institutions from financial fraud.

3. Introduction

With the rapid growth of digital banking and e-commerce, financial fraud has become a major concern. Traditional methods of fraud detection are no longer sufficient to address the increasing sophistication of fraudulent activities. Machine learning and data analytics provide innovative solutions to detect anomalies and identify fraudulent patterns. These advanced methods can analyze large volumes of transactional data, identifying subtle patterns that human analysts might miss. This project aims to develop a comprehensive fraud detection system to cater to both individual users and financial institutions, with features that include real-time detection, bulk analysis, and a user-friendly web interface.

The introduction of real-time fraud detection systems represents a significant advancement in the financial sector. By integrating machine learning models with accessible interfaces, such systems enable faster responses to potential threats, minimizing the impact of fraudulent activities. This project combines theoretical insights with practical implementation to deliver a reliable and efficient fraud detection system.

4. Literature Survey

Fraud detection in financial transactions is a critical area of research, as financial institutions face increasing challenges from sophisticated fraudulent activities. Machine learning (ML) has emerged as a powerful tool in addressing these challenges by identifying complex patterns and anomalies in large volumes of transaction data. Early approaches to fraud detection utilized traditional statistical and rule-based methods, but these techniques often struggled to keep pace with evolving fraud tactics. As a result, machine learning algorithms, particularly supervised models such as Decision Trees, Random Forests, and Support Vector Machines (SVM), have been widely adopted for detecting fraudulent transactions.

In addition to supervised learning, unsupervised learning techniques have gained attention for fraud detection, particularly in scenarios where labeled data is scarce. Anomaly detection methods, including clustering and outlier detection algorithms such as k-means, Gaussian Mixture Models, and Isolation Forest, have been used to detect unusual patterns indicative of fraud. Chandola et al. (2009) explored these anomaly detection techniques, highlighting their effectiveness in identifying novel fraudulent activities that may not be represented in the training data. The Isolation Forest algorithm, introduced by Liu et al. (2012), is particularly notable for its efficiency in handling large datasets and is effective in identifying fraudulent transactions as outliers.

As the complexity of fraud detection increases, deep learning techniques have become increasingly important. Zhao et al. (2017) applied deep neural networks (DNNs) and Long Short-Term Memory (LSTM) networks to capture intricate patterns and long-term dependencies in transaction sequences, which allowed for the detection of more sophisticated fraud. Autoencoders, another deep learning approach, have been used to detect anomalies by learning to reconstruct normal transaction data and flagging those with high reconstruction errors as potential fraud. Cheng et al. (2018) demonstrated the efficacy of autoencoders for fraud detection, particularly in cases where fraudulent transactions are rare or unknown.

5. Dataset

	step	type	amount	nameOrig	oldbalanceOrg	newbalanceOrig	nameDest	oldbalanceDest	newbalanceDest	isFraud	isFlaggedFraud
0	283	CASH_IN	210329.84	C1159819632	3778062.79	3988392.64	C1218876138	1519266.60	1308936.76	0	0
1	132	CASH_OUT	215489.19	C1372369468	21518.00	0.00	C467105520	6345756.55	6794954.89	0	0
2	355	DEBIT	4431.05	C1059822709	20674.00	16242.95	C76588246	80876.56	85307.61	0	0
3	135	CASH_OUT	214026.20	C1464960643	46909.73	0.00	C1059379810	13467450.36	13681476.56	0	0
4	381	CASH_OUT	8858.45	C831134427	0.00	0.00	C579876929	1667180.58	1676039.03	0	0

Fig.5.1.Dataset

5.1. Dataset Composition

- **Number of Rows (Transactions):** 636,262

- **Number of Columns (Features):** 11

5.2. Dataset Composition

Feature Name	Data Type	Description
Step	Integer	Represents the time step in hours. Each step corresponds to an hour since the start of data collection.
Amount	Numeric	The transaction amount in monetary units. Helps in identifying anomalies such as unusually high amounts.
NameOrig	String	Anonymized identifier of the sender (origin account).
OldbalanceOrg	Numeric	Initial balance of the sender before the transaction.
NewbalanceOrig	Numeric	Updated balance of the sender after the transaction.
NameDest	String	Anonymized identifier of the recipient (destination account).
OldbalanceDest	Numeric	Initial balance of the recipient before the transaction.
NewbalanceDest	Numeric	Updated balance of the recipient after the transaction.
Type	Categorical	Type of transaction. Possible values include: CASH-IN, CASH-OUT, DEBIT, PAYMENT, TRANSFER.
IsFraud	Binary	Target variable indicating if the transaction is fraudulent. (0 = Not Fraud, 1 = Fraud)
IsFlaggedFraud	Binary	Indicates transactions flagged as suspicious by monitoring systems (0 = Not Flagged, 1 = Flagged).

6. Methodology

6.1 Exploratory Data Analysis (EDA)

6.1.1 Visualization Tools

6.1.1.1 Matplotlib

Matplotlib is a versatile, low-level library used for creating static, interactive, and animated visualizations in Python. It serves as the foundation for many other visualization libraries.

- Supports a wide variety of plots: line charts, bar charts, scatter plots, histograms, etc.
- Highly customizable: You can control every aspect of a plot, from axis labels to colors and line styles.
- Exports high-quality visualizations in multiple formats (e.g., PNG, PDF, SVG).
- Includes both functional (pyplot) and object-oriented interfaces for greater flexibility.
- Simple visualizations for quick insights (e.g., plt.plot()).
- Customizing plots for research papers or presentations.
- Creating subplots and advanced layouts.

6.1.1.2 Seaborn

Seaborn is built on top of Matplotlib and provides a high-level interface for creating aesthetically pleasing and informative statistical graphics.

- Simplifies complex visualizations with minimal code.
- Built-in themes and styles for attractive plots (e.g., darkgrid, whitegrid).
- Specializes in statistical plots: distribution plots, heatmaps, pair plots, etc.
- Easily integrates with Pandas DataFrames for data visualization.

- **Univariate Analysis:** Examined distributions of individual numerical variables like `Amount` and `Balances`.
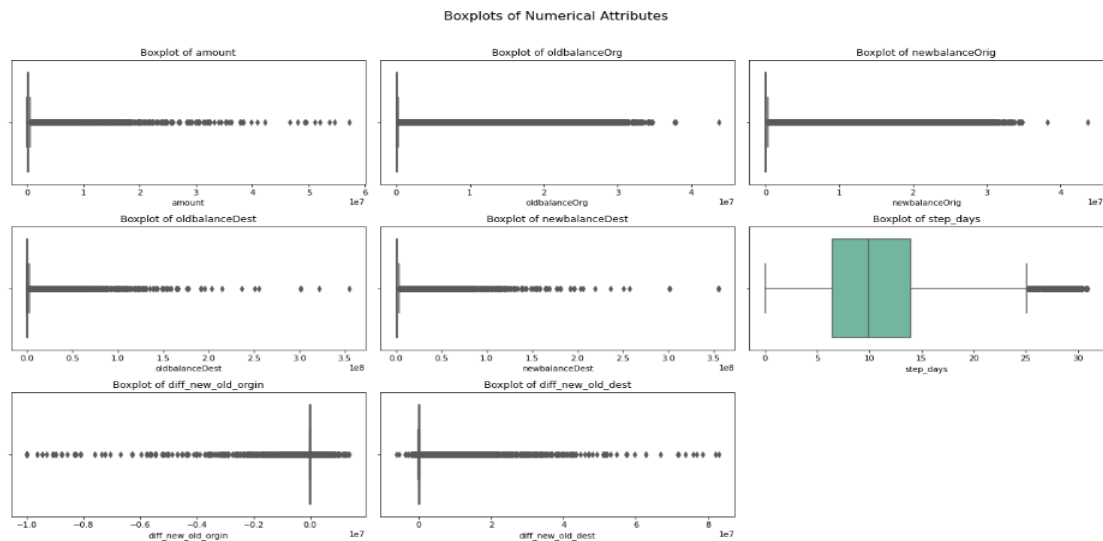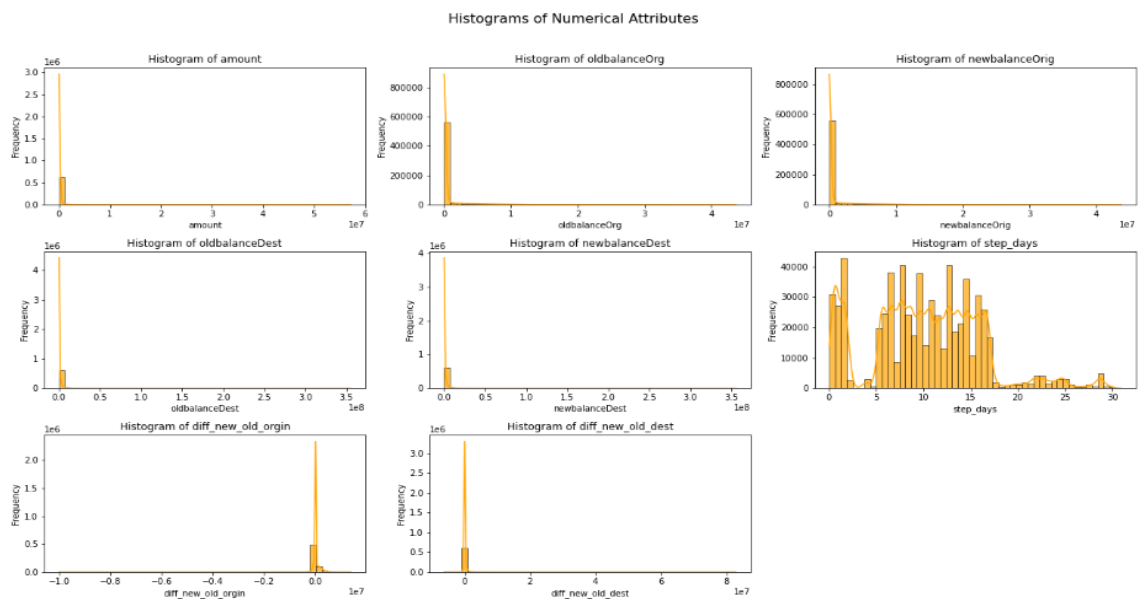

Fig.6.1.1.Boxplots(Univairate)
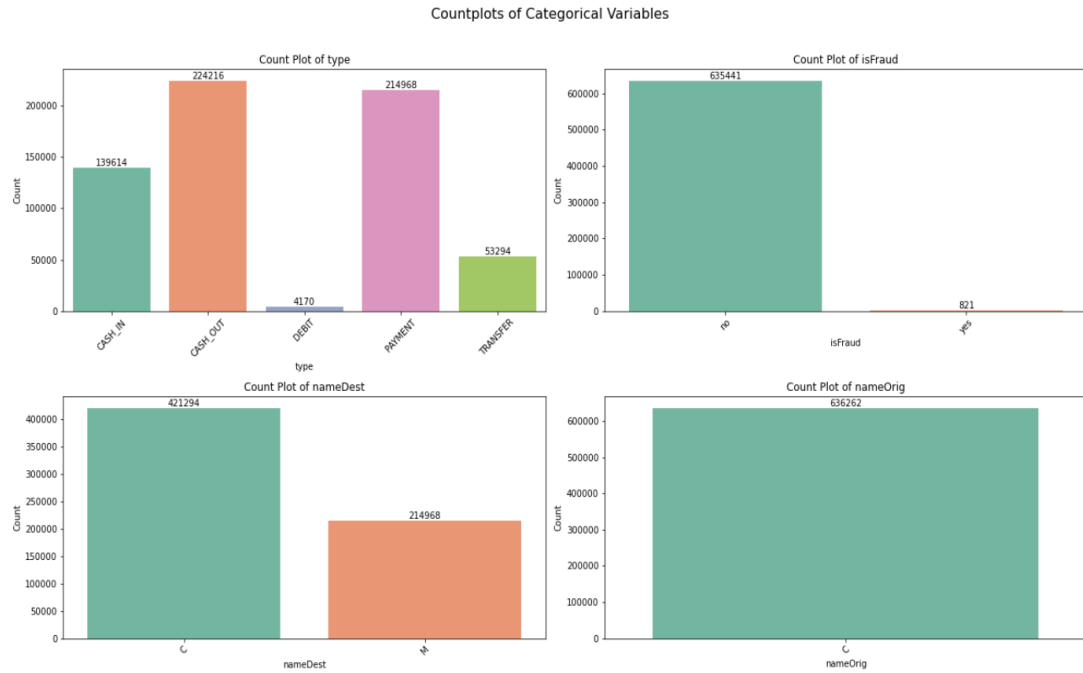

Fig.6.1.2.Histograms

Fig.6.1.3.Countplots

- **Bivariate and Multivariate Analysis:** Explored relationships between features using heatmaps and scatter plots.
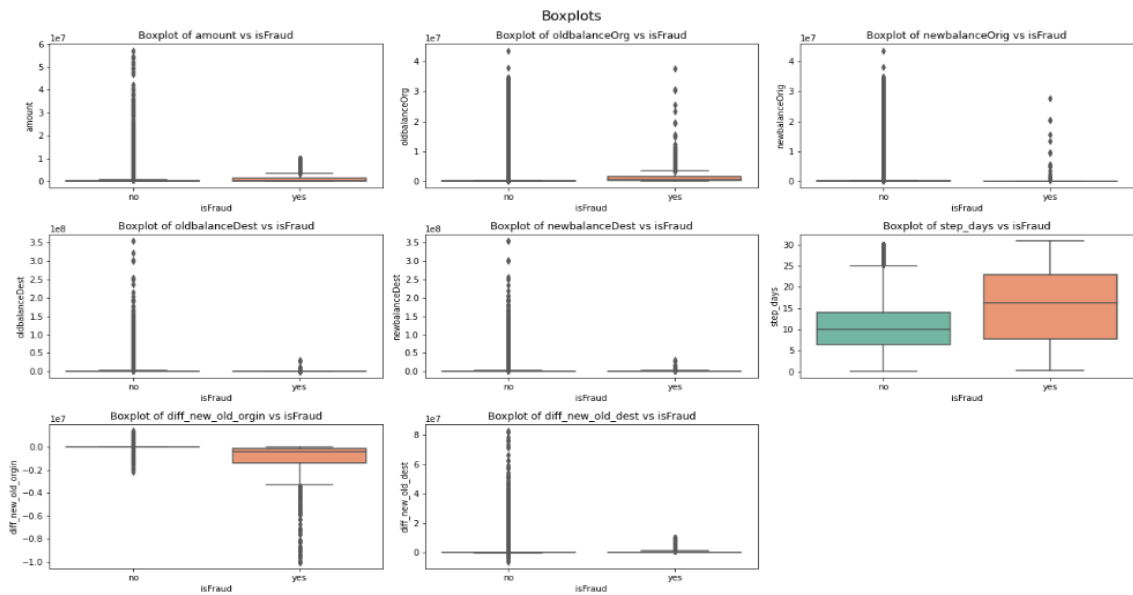

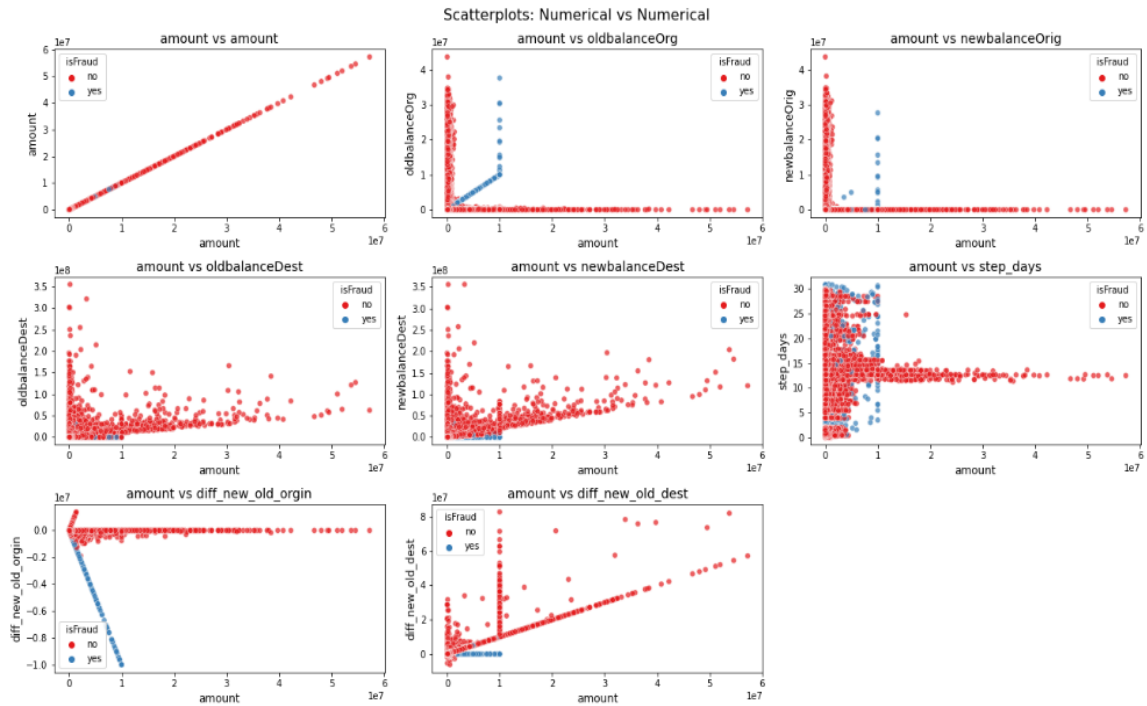
Fig.6.1.4.Boxplots(Bivariate)

Fig.6.1.5.Scatterplots

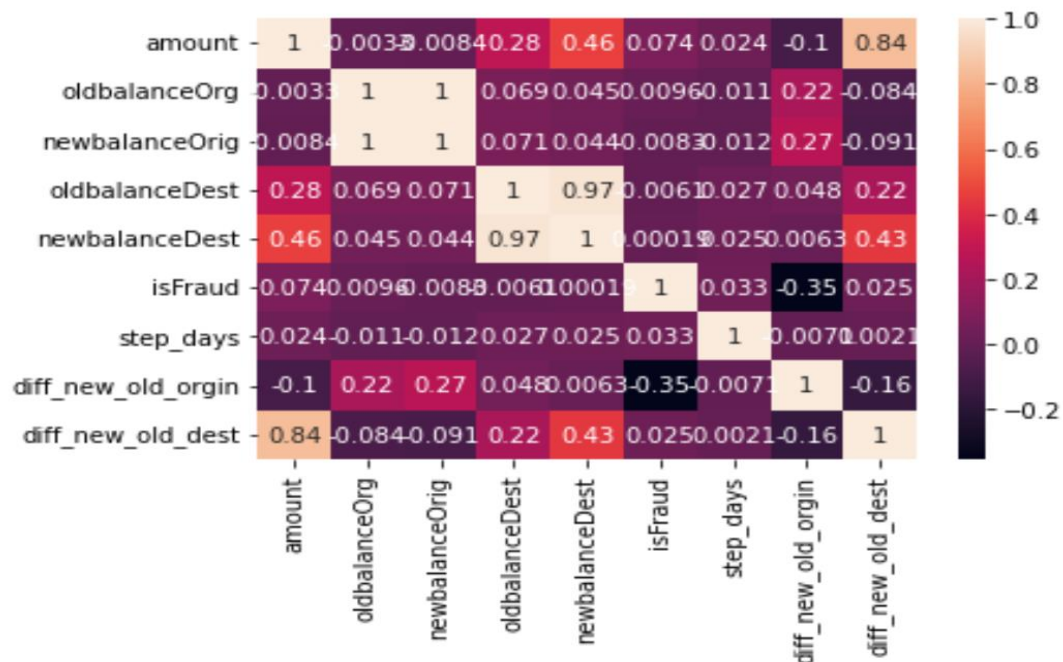- **Heatmap Insights:** Verified the absence of missing values or duplicates in the dataset.

Fig.6.1.6.Heatmap

6.2. Data Preprocessing

- **Dropping Unnecessary Columns:** `IsFlaggedFraud` was dropped due to low utility.

- **Feature Encoding:** One-hot encoding was applied to the `Type` column.

	type_CASH_IN	type_CASH_OUT	type_DEBIT	type_PAYMENT	type_TRANSFER
0	1.0	0.0	0.0	0.0	0.0
1	0.0	1.0	0.0	0.0	0.0
2	0.0	0.0	1.0	0.0	0.0
3	0.0	1.0	0.0	0.0	0.0
4	0.0	1.0	0.0	0.0	0.0

Fig.6.2.1.Encoding

- **Handling Imbalance:** SMOTE (Synthetic Minority Oversampling Technique) and undersampling were used.
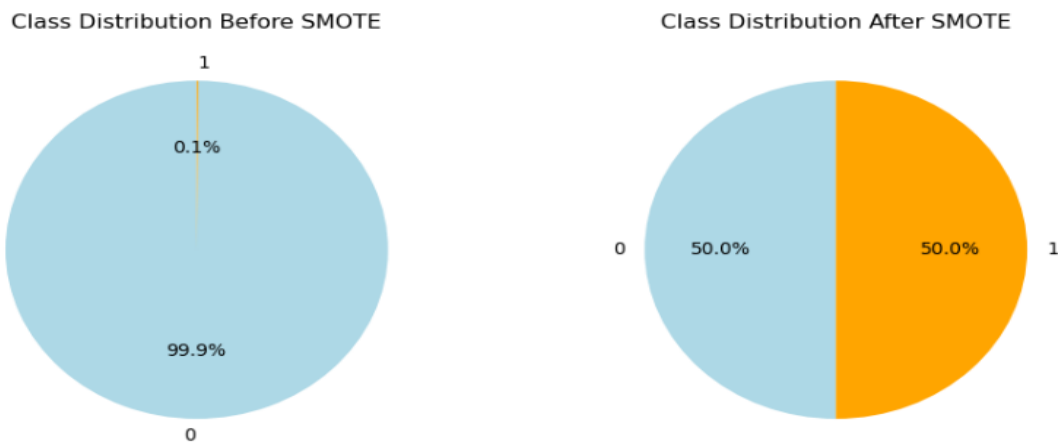
Fig.6.2.2.SMOTE

- **Feature Scaling:** Robust scaling, which uses the median and IQR, was applied to handle outliers effectively.

6.3. Model Training and Evaluation

6.3.1. Algorithms Used:

➢ **K-Nearest Neighbors (KNN)**

- A distance-based algorithm that classifies transactions by comparing them to their nearest neighbors in the feature space.
- Strength: Simple and intuitive.
- Limitation: Can be computationally expensive with large datasets.

➢ **Decision Tree**

- A tree-structured algorithm that splits data based on feature thresholds to classify transactions.
- Strength: Easy to interpret and visualize.
- Limitation: Prone to overfitting on noisy data.

➢ **Random Forest**

- An ensemble method that builds multiple decision trees and aggregates their predictions.
- Strength: Reduces overfitting and improves accuracy.
- Limitation: Less interpretable than a single decision tree.

➢ **XGBoost**

- A gradient boosting algorithm that optimizes the model by combining weak learners iteratively.
- Strength: Highly efficient and performs well on structured data.
- Limitation: Requires careful tuning to avoid overfitting.

> **LightGBM (LGBM)**

- A gradient boosting framework designed for speed and efficiency with large datasets.
- Strength: Handles categorical features natively and is computationally faster than XGBoost.
- Limitation: Sensitive to hyperparameter tuning.

6.3.2.Evaluation Metrics:

> **Confusion Matrix**

- A table showing true positives, true negatives, false positives, and false negatives.
- Provides an overview of model performance.

Fig.6.3.2.1.Confusion Matrix

> **Precision**

- Measures the proportion of correctly identified fraudulent transactions among all transactions predicted as fraud.
- Precision = TP / (TP + FP)

> **Recall**

- Measures the proportion of actual fraudulent transactions correctly identified by the model.
- Recall = TP / (TP + FN)

➢ **F1-Score**

- The harmonic mean of precision and recall, balancing both metrics.
- F1score = (2 × Precision × Recall) / (Precision + Recall)

6.4. Model Training Results

MODEL	TRAINING METRICES					
Logistic Regression	Accuracy	Precision	Recall	F1 Score	Balanced Accuracy	ROC AUC
	Metrics 0.53	1.0	0.53	0.69	0.76	0.92
KNN Classifier	Accuracy	Precision	Recall	F1 Score	Balanced Accuracy	ROC AUC
	Metrics 0.75	1.0	0.75	0.85	0.87	0.89
Decision Tree	Accuracy	Precision	Recall	F1 Score	Balanced Accuracy	ROC AUC
	Metrics 0.7	1.0	0.7	0.82	0.84	0.84
Random Forest	Accuracy	Precision	Recall	F1 Score	Balanced Accuracy	ROC AUC
	Metrics 0.91	1.0	0.91	0.95	0.93	0.99
XG Boost	Accuracy	Precision	Recall	F1 Score	Balanced Accuracy	ROC AUC
	Metrics 0.67	1.0	0.67	0.8	0.84	0.97
Light GBM	Accuracy	Precision	Recall	F1 Score	Balanced Accuracy	ROC AUC
	Metrics 0.83	1.0	0.83	0.91	0.91	0.99

Fig.6.4.1.Model Training Results

6.5. Model Deployment

Deploying machine learning models is a critical step in transitioning from development to practical application, enabling real-time predictions and user interactions. In this project, we utilized two prominent Python frameworks for deployment: Flask and Streamlit.

Flask is a lightweight web framework that facilitates the creation of web applications and APIs. By leveraging Flask, we exposed our trained model as a RESTful web service, allowing external applications to send transaction data and receive fraud predictions in response. This approach provides flexibility and control over the deployment process, making it suitable for integration into existing systems.

On the other hand, Streamlit is an open-source app framework designed specifically for machine learning and data science projects. It enables the rapid development of interactive web applications with minimal coding effort. Using Streamlit, we built a user-friendly interface where users can input transaction details and obtain immediate fraud detection results. This streamlined the deployment process, allowing for quick sharing and demonstration of our model's capabilities.

By employing both Flask and Streamlit, we ensured that our fraud detection model is accessible and user-friendly, catering to various deployment needs and enhancing its practical utility.

7. Future Scope

The future of fraud transaction detection presents several exciting opportunities to improve model accuracy, scalability, and real-world applicability. Key areas for future development include:

1. **Real-time Detection**: Transitioning to real-time fraud detection systems, using frameworks like Apache Kafka or Spark, will enable instant flagging of fraudulent transactions, minimizing damage before fraudsters can act.

2. **Anomaly Detection**: Expanding the use of unsupervised learning methods (e.g., autoencoders, Isolation Forest) to detect rare fraud patterns in highly imbalanced datasets, where labeled data is scarce.

3. **Scalability and Cloud Integration**: Leveraging cloud platforms (AWS, Azure, Google Cloud) for deploying fraud detection models can handle large-scale transaction volumes, providing flexibility and computational power for growing datasets.

4. **Privacy and Security**: Implementing privacy-preserving techniques like federated learning and homomorphic encryption will ensure sensitive data remains secure while still enabling fraud detection, meeting regulatory requirements such as GDPR.

5. **Cross-industry Applications**: Extending fraud detection capabilities to other sectors like e-commerce, healthcare, and insurance could offer broader applicability, adapting models to detect domain-specific fraudulent behavior.

6. **Model Interpretability**: Enhancing model transparency through techniques like SHAP or LIME will help explain the rationale behind fraud predictions, increasing user trust and aiding compliance with regulatory standards.

7. **Automated Response Systems**: Developing automated actions (e.g., blocking transactions, alerting users) based on detected fraud patterns can improve response times and reduce the impact of fraudulent activity.

By focusing on these areas, the fraud transaction detection system can evolve to become more robust, adaptive, and scalable, ensuring continued effectiveness in detecting fraud in an increasingly complex and fast-paced digital environment.
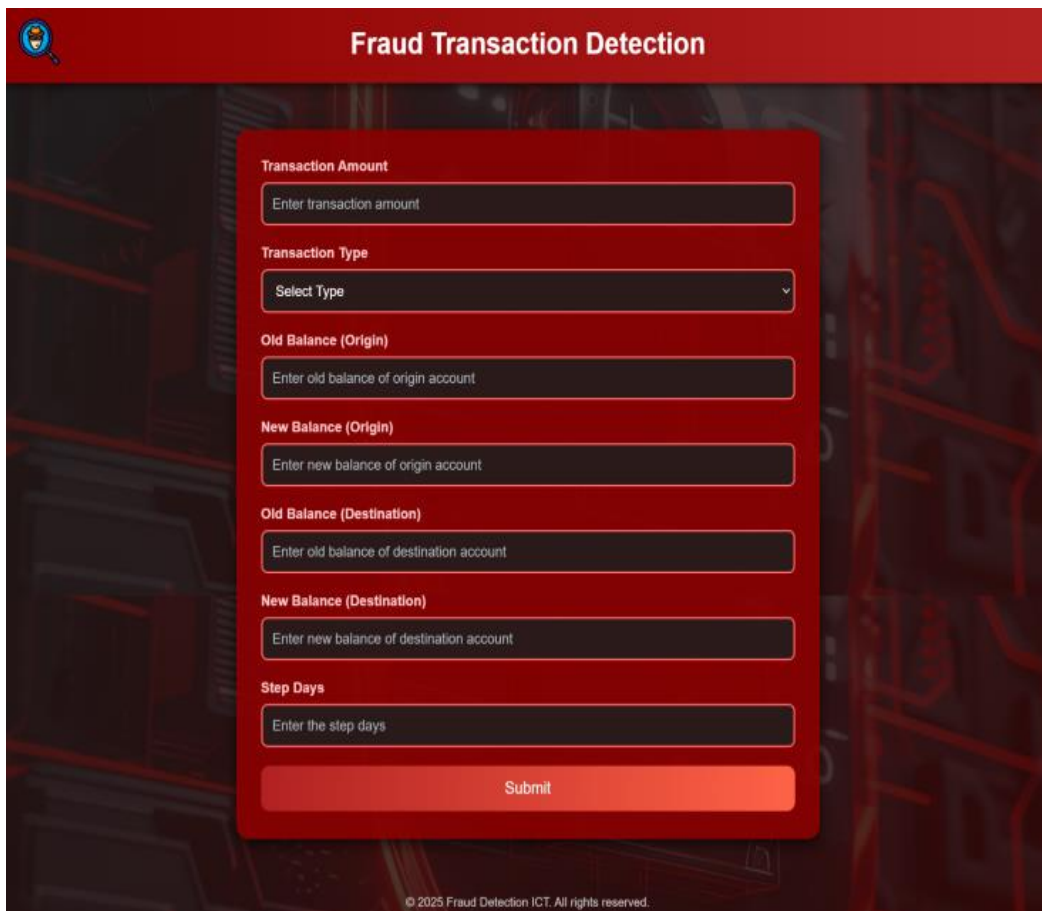
8.Limitations

1. **Imbalanced Data**: Fraudulent transactions are rare, making datasets highly imbalanced, which can lead to biased models and lower detection accuracy for fraud.

2. **Lack of Labeled Data**: A sufficient amount of labeled fraud data is often not available, limiting the ability to train accurate supervised models and detect emerging fraud patterns.

3. **Evolving Fraud Tactics**: Fraudsters constantly evolve their techniques to bypass detection systems, requiring models to be updated regularly, which can be resource-intensive.

4. **False Positives**: Fraud detection models may flag legitimate transactions as fraud, leading to customer dissatisfaction and operational inefficiencies.

5. **Model Complexity**: Advanced machine learning models can be computationally expensive and difficult to scale, especially for real-time applications.

6. **Feature Engineering Challenges**: Selecting the right features for fraud detection is challenging, and poor feature selection can lead to suboptimal model performance.

7. **Explainability Issues**: Many machine learning models are "black-box" models, making it difficult to interpret why certain transactions are flagged, which can impact trust and compliance.

8. **Privacy Concerns**: Transaction data contains sensitive information, and ensuring privacy while processing this data, especially under regulatory requirements (e.g., GDPR), is a major challenge.

9. **Scalability**: As transaction volumes increase, ensuring the system can scale efficiently without compromising performance is a key concern.

10. **Data Quality**: Inconsistent, incomplete, or inaccurate data can negatively impact model performance, requiring extensive data cleaning and preprocessing.

Despite these challenges, advancements in machine learning and data privacy techniques are helping mitigate many of these limitations

9. Results

The **Random Forest model** shows strong performance with high accuracy, recall, and F1-score, making it a reliable choice for fraud detection. The model is effective at minimizing false negatives (missed fraud cases) and false positives (wrongly flagged legitimate transactions), making it a robust tool for identifying fraudulent transactions in a financial system. With these results, this model can be considered suitable for real-world deployment in detecting fraud.

9.1. Using Flask

Fig.9.1.1.Home Page

Fig.9.1.2.Result Page

9.2. Using Streamlit

Fig.9.2.1.Streamlit Webpage

10. Conclusion

The Fraud Transaction Detection System effectively identifies and mitigates fraudulent activities in financial transactions. By integrating machine learning algorithms and user-friendly deployment, the system provides a scalable solution for both individuals and organizations. The system's ability to analyze both individual and bulk transactions makes it versatile and adaptable to various use cases. Future improvements include real-time transaction monitoring and the integration of additional data sources for enhanced accuracy.

Further development could focus on reducing false positives and improving the user experience of the web interface. Additionally, incorporating external data sources such as geolocation or transaction history could improve the model's predictive power. This project lays the foundation for more advanced fraud detection systems, ensuring financial security in an increasingly digital world.

11.References

- Transaction dataset from Kaggle

- Hyperparameter tuning - GeeksforGeeks

- XGBoost Documentation — xgboost 2.1.3 documentation

- Feature Encoding Techniques - Machine Learning - GeeksforGeeks

- Stack Overflow - Where Developers Learn, Share, & Build Careers