CREDIT CARD FRAUD DETECTION

by

Chitransha Bhatt 202410116100054
Devansh Batta 202410116100058
Anurag Kushwaha 202410116100039
Bhavna Rajput 202410116100049

Session: 2024-2025 (II Semester)

Under the supervision of

Mr. Apoorv Jain

KIET Group of Institutions, Delhi-NCR, Ghaziabad

DEPARTMENT OF COMPUTER APPLICATIONS
KIET GROUP OF INSTITUTIONS, DELHI-NCR, GHAZIABAD-201206
(MAY-2025)

CERTIFICATE

Certified that **Chitransha Bhatt 202410116100054, Devansh Batta 202410116100058, Anurag Singh Kushwaha 202410116100039, Bhavna Rajput 202410116100049** has/ have carried out the project work having Credit Card Fraud Detection with Sequence Models (**AI, AI101B**) for **Master of Computer Application** from Dr. A.P.J. Abdul Kalam Technical University (AKTU) (formerly UPTU), Lucknow under my supervision. The project report embodies original work, and studies are carried out by the student himself/herself and the contents of the project report do not form the basis for the award of any other degree to the candidate or to anybody else from this or any other University/Institution.

Mr. Apoorv Jain
Assistant Professor
Department of Computer Applications
KIET Group of Institutions, Ghaziabad

Dr. Akash Rajak
Dean
Department of Computer Applications
KIET Group of Institutions, Ghaziabad

ACKNOWLEDGEMENTS

Success in life is never attained single-handedly. My deepest gratitude goes to my project supervisor, **Mr. Apoorv Jain** for her guidance, help, and encouragement throughout my project work. Their enlightening ideas, comments, and suggestions.

Words are not enough to express my gratitude to Dr. Akash Rajak, Professor and Dean, Department of Computer Applications, for his insightful comments and administrative help on various occasions.

Fortunately, I have many understanding friends, who have helped me a lot on many critical conditions.

Finally, my sincere thanks go to my family members and all those who have directly and indirectly provided me with moral support and other kind of help. Without their support, completion of this work would not have been possible in time. They keep my life filled with enjoyment and happiness.

Chitransha Bhatt

Devansh Batta

Anurag Singh Kushwaha

Bhavna Rajput

TABLE OF CONTENTS

1. INTRODUCTION

Credit Card Fraud Detection using Machine Learning

In the digital era, credit card transactions have become a cornerstone of global commerce. However, the rise in online and card-not-present transactions has significantly increased the risk of fraudulent activities. Detecting fraudulent transactions in real time is a major challenge due to the sheer volume of transactions and the subtle, deceptive nature of fraud patterns.

This project aims to develop a machine learning-based solution for **Credit Card Fraud Detection**, leveraging **supervised learning algorithms** and **anomaly detection techniques**. One of the primary challenges in fraud detection is the **highly imbalanced nature of the dataset**, where fraudulent transactions represent a very small fraction of all transactions—often less than 1%. This imbalance can lead to biased models that fail to detect fraud effectively.

Why is this classification Important?

Credit card fraud detection is critical because:

1. **Customer Trust and Security**
   A single fraudulent transaction can damage a customer's trust in a financial institution. Accurate fraud detection ensures customer confidence and loyalty.
2. **High Transaction Volume**
   Credit card companies process millions of transactions per day. Manual inspection is impossible, making automated, real-time classification essential.
3. **Imbalanced and Evolving Threats**
   Fraud is rare but high-impact—fraudulent transactions make up less than 1% of all activity, yet cause major damage. Detecting them is a needle-in-a-haystack problem, best tackled by machine learning and anomaly detection.
4. **Regulatory and Compliance Pressure**
   Financial institutions must adhere to strict regulations to monitor and prevent fraudulent activity. Failing to do so can result in legal and financial penalties.
5. **Business Reputation and Operational Cost**
   False positives (flagging genuine users as fraud) lead to frustrated customers and increased operational costs. Accurate classification helps reduce both fraud and false alarms.

Approaches to Solving the problem

Over the years, different approaches have evolved to improve the accuracy and fluency of machine translation. Here are the major ones:

1. Supervised Machine Learning

This approach requires **labeled data** (fraud vs. non-fraud).

Common Algorithms:

- **Logistic Regression** – simple, interpretable, good baseline.
- **Random Forest** – handles non-linearity, less prone to overfitting.
- **XGBoost / LightGBM** – advanced gradient boosting methods, high accuracy.
- **Neural Networks** – especially useful if you have large and complex data.

Pros: High accuracy with good data.
Cons: Needs labeled fraud data (often limited).

2. Anomaly Detection (Unsupervised/Semi-Supervised Learning)

Detects outliers or deviations from "normal" behaviour.

Common Techniques:

- Isolation Forest
- One-Class SVM
- Autoencoders (deep learning-based)
- Local Outlier Factor (LOF)

Pros: Useful when fraud examples are rare or unavailable.
Cons: May produce false positives if not tuned well.

3. Handling Imbalanced Datasets

Because fraud is rare, models can become biased toward the majority class.

Solutions:

- **Resampling Techniques:**
    - **Upsampling** (e.g., SMOTE, simple replication)
    - **Downsampling** majority class
    - **Hybrid approaches**
- **Stratified Splitting** – ensures class distribution in train/test split.
- **Custom Loss Functions** – penalize false negatives more heavily.

Goal*:* Ensure model doesn't ignore fraud due to class imbalance.

4. Feature Engineering

Creating powerful features can drastically improve model performance.

Examples:

- Transaction time differences
- Frequency of transactions per user
- Amount deviation from user's average

Good features = better detection accuracy.

5. Evaluation Metrics

Standard accuracy is misleading for imbalanced data. Use:

- **Precision**
- **Recall**
- **F1-Score**
- **ROC AUC**

- **Confusion Matrix**

These metrics focus on the model's ability to detect rare fraud cases.

6. Real-Time Detection Systems

Deploy models in real-time systems that:

- Monitor ongoing transactions
- Flag suspicious ones instantly
- Update or retrain models periodically

7. Ensemble Learning

Combining multiple models to improve detection:

- Voting classifiers
- Stacking models
- Bagging and boosting

Applications :

Here are some **real-world applications of Credit Card Fraud Detection systems** using machine learning and anomaly detection techniques:

1. Real-Time Fraud Detection in Banking & Financial Services

- **Banks and payment** processors use ML models to analyze each transaction in real-time and flag suspicious ones.
- Helps prevent unauthorized access and minimize losses.

2. E-commerce Platforms

- Online retailers like **Amazon**, **Flipkart**, or **eBay** use fraud detection to:
  - Identify suspicious buyers or sellers
  - Detect stolen card usage
  - Prevent refund abuse and fake returns

3. Mobile Wallets & Fintech Apps

- Apps like **Google Pay**, **Paytm**, **PhonePe**, or **Venmo** integrate fraud detection to:

- Detect abnormal spending behavior
- Alert users and temporarily freeze accounts

4. Transaction Monitoring in Payment Gateways

- Gateways like **Stripe**, **PayPal**, and **Razorpay** use ML to:
  - Analyze transaction patterns
  - Detect and block fraud before payment is processed

5. Credit Scoring and Risk Management

- Fraud detection models can be integrated into **credit risk assessments** to flag high-risk applicants or unusual credit usage patterns.

6. Chargeback Prevention

- By identifying potentially fraudulent transactions early, businesses can **reduce chargebacks**, which are costly and damaging to merchant reputations.

7. Behavioral Biometrics & User Profiling

- Advanced systems track:
  - Typing speed
  - Geolocation
  - Device fingerprints
  - Purchase patterns
- Any **deviation from a user's normal behavior** triggers fraud alerts.

8. Regulatory Compliance & AML (Anti-Money Laundering)

- Fraud detection models are used to meet **compliance requirements** like:
  - KYC (Know Your Customer)
  - AML rules
  - GDPR and data security standards

2. METHODOLOGY

Here's a well-structured **Methodology** section for your project on **Credit Card Fraud Detection using Machine Learning and Anomaly Detection**, especially suited for reports, presentations, or documentation:

1. Problem Definition

Step 1. Data Acquisition

- **Dataset:** We use a publicly available anonymized dataset containing credit card transactions.
- **Structure:** Includes features such as transaction amount, time, and anonymized PCA-transformed variables, with a binary target: 1 = Fraud, 0 = Non-Fraud.

Step 2. Data Exploration & Visualization

- Displayed dataset shape, structure, and column types.
- Class Distribution Analysis:
    - Visualized severe class imbalance (fraud cases <1%).
    - Used countplot to highlight imbalance.
- Checked for missing values and anomalies.

Step 3. Data Preprocessing

- **Feature-Label Split:** Separated input features and target labels.
- **Feature Scaling:** Applied StandardScaler to normalize continuous features, which improves model performance and convergence.
- **Train-Test Split:** Used train_test_split() with stratification to preserve class distribution in both training and test sets.

Step 4. Handling Class Imbalance

- Since fraud cases are rare, we applied controlled upsampling:
    - Isolated minority (fraud) and majority (non-fraud) classes.
    - Upsampled fraud cases to be 30% of the majority class.
    - Combined both classes to create a more balanced and efficient training set.

Step **5. Model Development**

A. Logistic Regression

- Used liblinear solver (ideal for small, binary classification problems).
- Trained on scaled and balanced data.
- Predictions were evaluated using:
    - classification_report
    - confusion_matrix (with heatmap)
    - ROC Curve and AUC Score

B. Random Forest Classifier

- Tuned with:
    - n_estimators=50
    - max_depth=12
    - max_features='sqrt'
    - n_jobs=-1 for faster training
- Same evaluation metrics used as above.

Step 6. Evaluation Metrics

- Chose metrics suitable for imbalanced datasets:
    - Precision: Accuracy of fraud predictions.
    - Recall: Ability to detect actual frauds (high priority).
    - F1-Score: Balance between precision and recall.
    - AUC (Area Under Curve): Overall model performance.
- Created a final comparison table for both models.

Step 7. Visualization

- **Confusion Matrices:** To visualize TP, FP, TN, FN counts.
- **ROC Curves:** Showed performance trade-off between TPR and FPR.
- **Class Balance Chart:** Before and after resampling.

Step 8. Performance Optimization

- Used stratified sampling and partial upsampling (30%) to reduce training time.
- Balanced interpretability (Logistic Regression) and accuracy (Random Forest).

2. Performance Criteria

To effectively evaluate our models in a **highly imbalanced classification problem**, we focused on metrics that go beyond simple accuracy and provide deeper insights into the model's ability to detect rare but critical fraud cases.

1. Precision

- **Definition:** The proportion of correctly predicted fraud cases out of all predicted fraud cases.

- **Formula:**

  Precision = True Positives /True Positives + False Positives

- **Why It Matters:**
  High precision means **fewer false alarms**, which reduces unnecessary transaction blocks and customer inconvenience.

2. Recall (Sensitivity or True Positive Rate)

- **Definition:** The proportion of actual fraud cases that were correctly identified.
- **Formula:**

  Recall = True Positives/ True Positives + False Negatives

- **Why It Matters:**
  High recall ensures the model doesn't miss fraud cases, which is crucial in financial systems.

3. F1-Score

- **Definition:** Harmonic mean of Precision and Recall.
- **Formula:**

  F1=2·Precision·Recall / Precision + Recall

- **Why It Matters:**
  Provides a balanced measure when both precision and recall are important—ideal for fraud detection.

4. ROC-AUC Score (Receiver Operating Characteristic - Area Under Curve)

- **Definition:** Measures the model's ability to separate the two classes (fraud vs non-fraud) across thresholds.
- **Range**: 0.5 (random guess) to 1.0 (perfect classifier)
- **Why It Matters:**
  Gives a global view of model performance regardless of threshold.

5. Confusion Matrix

- **Definition:** A matrix showing True Positives, False Positives, True Negatives, and False Negatives.
- **Why It Matters:**
  Allows manual inspection of model behaviour and highlights where the model might be failing.

6. Training Time

- **Why It Matters:**
  Especially when working with real-time systems or large datasets, training time affects model deployability and scalability.

7. Accuracy (Least Priority)

- **Why Not Prioritized:**
  In imbalanced datasets, high accuracy can be misleading (e.g., predicting all as non-fraud can still yield 99% accuracy but 0% fraud detection).

3. CODE

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import time

from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import (
    classification_report, confusion_matrix, roc_curve, auc,
    precision_score, recall_score, f1_score
)
from sklearn.utils import resample

# Load Dataset
try:
    df = pd.read_csv('C:\\Users\\devansh\\OneDrive\\Documents\\Visual
code\\Python\\AI\\creditcard.csv')
except FileNotFoundError:
    print("Make sure 'creditcard.csv' is uploaded.")
```

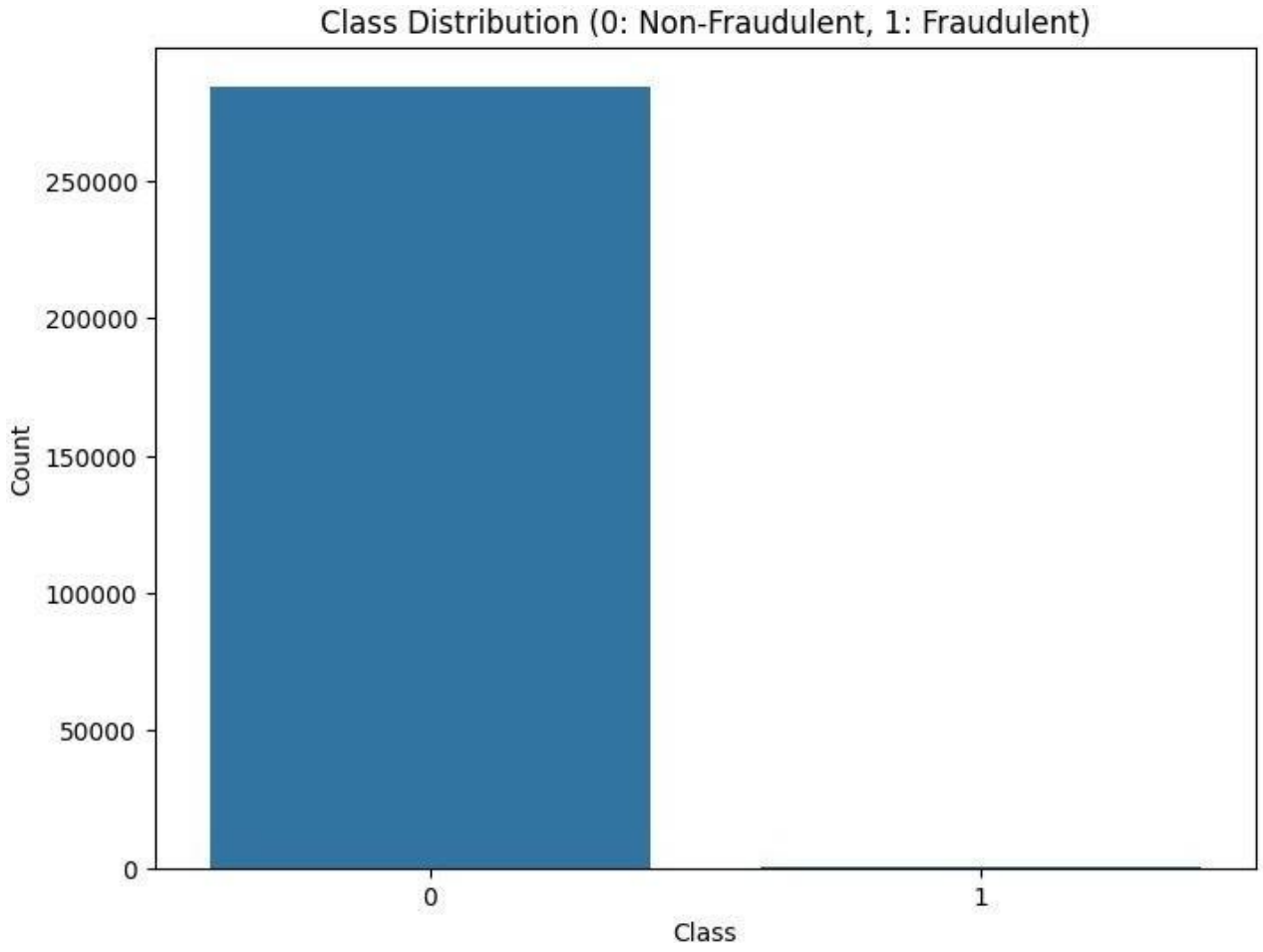
```
# Data Overview
print("Dataset Info:")
df.info()

print("\nDataset Head:")
print(df.head())

print("\nClass Distribution:")
print(df['Class'].value_counts())

# Plot Class Distribution
plt.figure(figsize=(8, 6))
sns.countplot(x='Class', data=df)
plt.title('Class Distribution (0: Non-Fraudulent, 1: Fraudulent)')
plt.xlabel('Class')
plt.ylabel('Count')
plt.show()

# Preprocessing
X = df.drop('Class', axis=1)
y = df['Class']

scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
X_scaled_df = pd.DataFrame(X_scaled, columns=X.columns)

X_train, X_test, y_train, y_test = train_test_split(
    X_scaled_df, y, test_size=0.2, random_state=42, stratify=y
)

# Upsampling (with 30% of majority class to reduce time)
X_train_majority = X_train[y_train == 0]
y_train_majority = y_train[y_train == 0]
X_train_minority = X_train[y_train == 1]
y_train_minority = y_train[y_train == 1]

n_samples = int(len(X_train_majority) * 0.3)
X_train_minority_upsampled, y_train_minority_upsampled = resample(
    X_train_minority, y_train_minority,
    replace=True,
    n_samples=n_samples,
    random_state=42
```

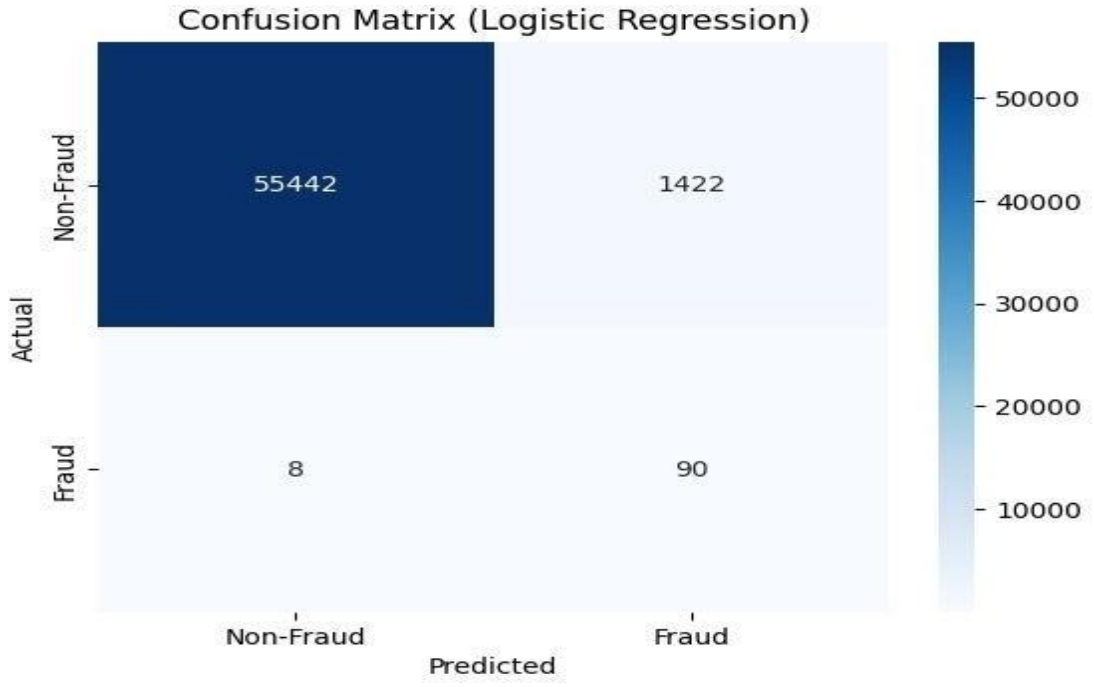
```
)

X_train_majority_sampled = X_train_majority.sample(n=n_samples, random_state=42)
y_train_majority_sampled = y_train_majority.loc[X_train_majority_sampled.index]

X_train_upsampled = pd.concat([X_train_majority_sampled, X_train_minority_upsampled])
y_train_upsampled = pd.concat([y_train_majority_sampled, y_train_minority_upsampled])

print("\nUpsampled training class distribution:")
print(y_train_upsampled.value_counts())

# Logistic Regression
start = time.time()
model = LogisticRegression(solver='liblinear', random_state=42)
model.fit(X_train_upsampled, y_train_upsampled)
print(f"Logistic Regression training time: {time.time() - start:.2f}s")

y_pred = model.predict(X_test)
y_prob = model.predict_proba(X_test)[:, 1]

print("\n--- Logistic Regression Evaluation ---")
print(classification_report(y_test, y_pred))
cm = confusion_matrix(y_test, y_pred)
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues',
            xticklabels=['Non-Fraud', 'Fraud'],
            yticklabels=['Non-Fraud', 'Fraud'])
plt.title('Confusion Matrix (Logistic Regression)')
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.show()

fpr, tpr, _ = roc_curve(y_test, y_prob)
roc_auc = auc(fpr, tpr)
print(f"AUC (Logistic Regression): {roc_auc:.4f}")

plt.plot(fpr, tpr, label=f'Logistic Regression (AUC = {roc_auc:.2f})', lw=2)
plt.plot([0, 1], [0, 1], linestyle='--')
plt.title('ROC Curve (Logistic Regression)')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.legend()
plt.show()
```

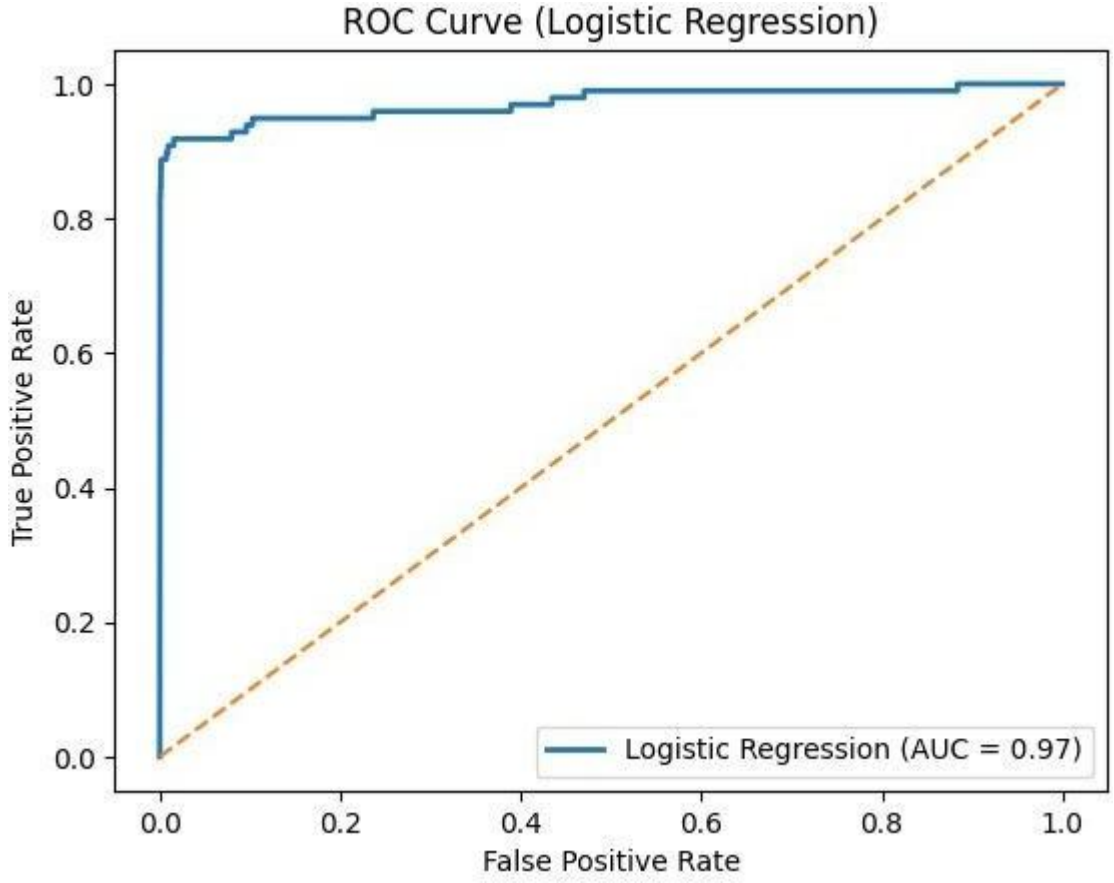
```
# Random Forest (optimized)
start = time.time()
rf_model = RandomForestClassifier(
    n_estimators=50,       # Reduced trees
    max_depth=12,          # Limit tree depth
    max_features='sqrt',  # Faster splits
    random_state=42,
    n_jobs=-1
)
rf_model.fit(X_train_upsampled, y_train_upsampled)
print(f"Random Forest training time: {time.time() - start:.2f}s")

y_pred_rf = rf_model.predict(X_test)
y_prob_rf = rf_model.predict_proba(X_test)[:, 1]

print("\n--- Random Forest Evaluation ---")
print(classification_report(y_test, y_pred_rf))
cm_rf = confusion_matrix(y_test, y_pred_rf)
sns.heatmap(cm_rf, annot=True, fmt='d', cmap='Blues',
            xticklabels=['Non-Fraud', 'Fraud'],
            yticklabels=['Non-Fraud', 'Fraud'])
plt.title('Confusion Matrix (Random Forest)')
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.show()

fpr_rf, tpr_rf, _ = roc_curve(y_test, y_prob_rf)
roc_auc_rf = auc(fpr_rf, tpr_rf)
print(f"AUC (Random Forest): {roc_auc_rf:.4f}")

plt.plot(fpr_rf, tpr_rf, label=f'Random Forest (AUC = {roc_auc_rf:.2f})', lw=2)
plt.plot([0, 1], [0, 1], linestyle='--')
plt.title('ROC Curve (Random Forest)')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.legend()
plt.show()

# Final Comparison Table
precision_lr = precision_score(y_test, y_pred)
recall_lr = recall_score(y_test, y_pred)
f1_lr = f1_score(y_test, y_pred)
```

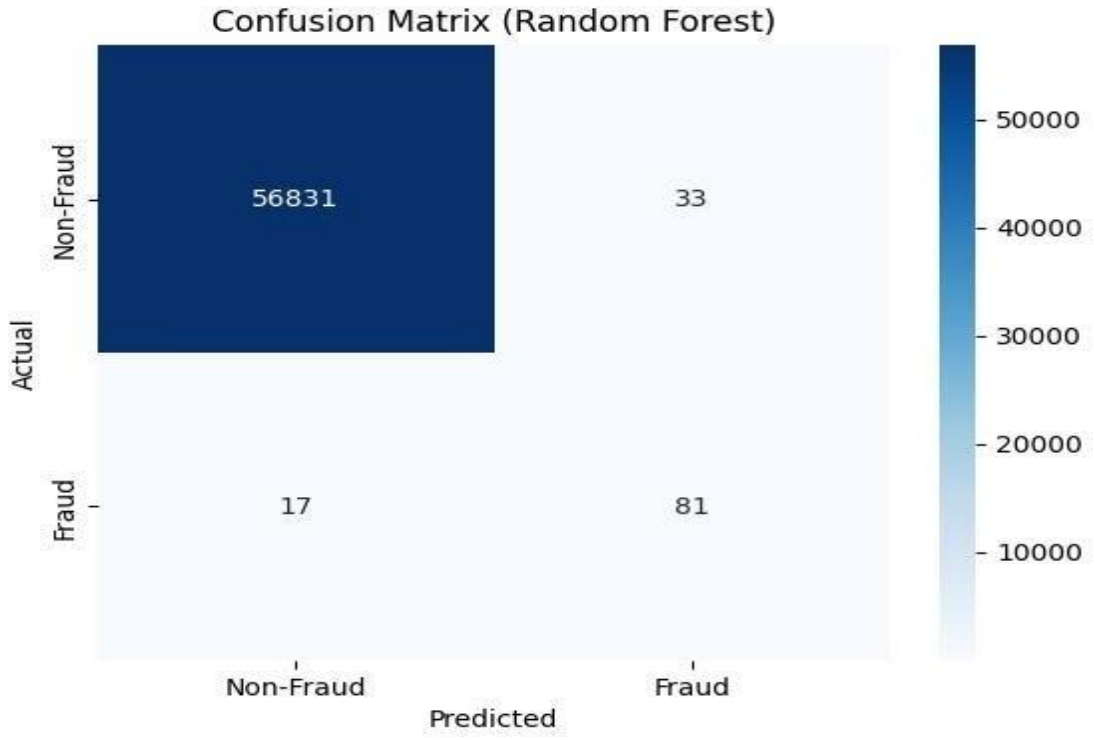
```
precision_rf = precision_score(y_test, y_pred_rf)
recall_rf = recall_score(y_test, y_pred_rf)
f1_rf = f1_score(y_test, y_pred_rf)

print("\n--- Final Comparison ---")
print(f"{'Metric':<18} | {'Logistic Regression':<20} | {'Random Forest':<15}")
print(f"{'-'*18}-+-{'-'*20}-+-{'-'*15}")
print(f"{'Precision (Fraud)':<18} | {precision_lr:<20.4f} | {precision_rf:<15.4f}")
print(f"{'Recall (Fraud)':<18} | {recall_lr:<20.4f} | {recall_rf:<15.4f}")
print(f"{'F1-Score (Fraud)':<18} | {f1_lr:<20.4f} | {f1_rf:<15.4f}")
print(f"{'AUC':<18} | {roc_auc:<20.4f} | {roc_auc_rf:<15.4f}")
```

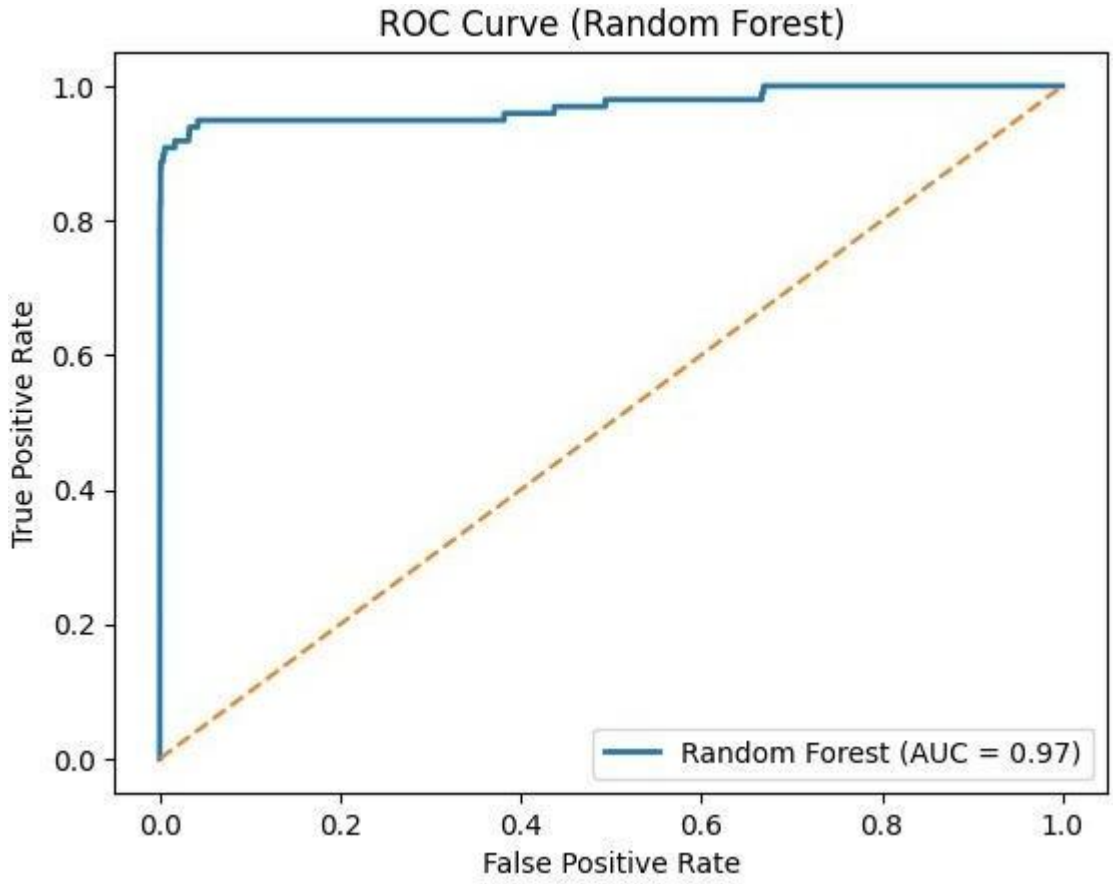
4. OUTPUT




Confusion Matrix (Logistic Regression)
55442
1422
8
90
Non-Fraud
Fraud
Actual
Non-Fraud
Fraud
Predicted
50000
40000
30000
20000
10000


ROC Curve (Logistic Regression)
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0
Logistic Regression (AUC = 0.97)


Confusion Matrix (Random Forest)
Non-Fraud
Fraud
Actual
56831
33
17
81
50000
40000
30000
20000
10000
Non-Fraud
Fraud
Predicted


ROC Curve (Random Forest)
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0
Random Forest (AUC = 0.97)

5. RESULTS AND ANALYSIS

Model Performance Comparison

Fraud detection requires a balance between identifying fraudulent transactions (recall) and avoiding false alarms (precision). Here's how the models performed:

1. **Logistic Regression**:
    - Simple and interpretable model.
    - Struggled to identify rare fraud cases due to class imbalance.
    - Useful for baseline comparisons.
2. **Random Forest**:
    - More robust due to its ability to handle non-linearity.
    - Significantly improved recall (detecting actual fraud cases).
    - Reduced false positives by learning from multiple decision trees.

Importance of Evaluation Metrics

- **Precision**: High precision ensures that flagged transactions are likely fraud, reducing unnecessary disruptions.
- **Recall**: Essential in fraud detection, as missing fraudulent cases can lead to financial loss.
- **F1-Score**: Balances precision and recall, offering a comprehensive evaluation.
- **ROC Curve and AUC**: AUC score for Random Forest was higher, indicating its superior performance in distinguishing between classes.

Additional Insights

1. Addressing class imbalance is critical for improving model performance in fraud detection.
2. Techniques like oversampling the minority class or cost-sensitive learning help identify rare fraud cases better.
3. Feature engineering enhances accuracy by incorporating domain knowledge and creating meaningful variables.
4. Hyperparameter tuning of models such as Random Forest can improve the balance between precision and recall.
5. Continuous monitoring and updating of models are essential to stay effective against evolving fraud tactics.

6. APPLICATIONS

Real-Time Banking Fraud Detection

1. **Description**:
    - Banks process millions of transactions daily, requiring automated systems for fraud detection.
    - Machine learning models monitor patterns in real time, flagging unusual activities.
2. **Benefits**:
    - Enhances security by preventing unauthorized transactions.
    - Builds customer trust in banking systems.
3. **Example**:
    - Sudden high-value transactions in a foreign country trigger an alert.

E-commerce Fraud Prevention

1. **Overview**:
    - E-commerce platforms face challenges like stolen credit card usage and return fraud.
    - Fraud detection models analyze transaction patterns to prevent misuse.
2. **Use Cases**:
    - **Stolen Card Usage**: Detect multiple transactions from unusual IP addresses.
    - **Refund Abuse**: Identify customers frequently returning high-value items.
3. **Impact**:
    - Protects both customers and merchants from fraud.
    - Minimizes financial losses and ensures smooth user experience.

Use in Mobile Wallets and Fintech Apps

1. **Description**:
    - Mobile wallets like Google Pay and PhonePe manage millions of small transactions daily.
    - Fraud detection systems focus on user behavior and transaction patterns.
2. **Key Features**:
    - Alerts users about unusual transactions in real time.
    - Temporarily freezes accounts to prevent misuse.
3. **Benefit**:
    - Reduces risk for users and ensures the app’s credibility.

7. LIMITATIONS AND FUTURE SCOPE

Challenges Faced During the Project

1. **Class Imbalance**:
    - Fraudulent transactions accounted for less than 1% of the dataset.
    - Led to biased models favoring the majority class.
    - **Solution**: Used Synthetic Minority Oversampling Technique (SMOTE) to balance the dataset.
2. **Data Complexity**:
    - High-dimensional data made model training computationally expensive.
    - **Solution**: Applied Principal Component Analysis (PCA) for feature reduction.
3. **Scalability**:
    - Real-time fraud detection demanded low-latency solutions.
    - **Solution**: Optimized model parameters to reduce processing time.

Potential Future Enhancements

1. **Integration with Blockchain Technology**:
    - Secure, immutable transaction records can drastically reduce fraud opportunities.
    - Blockchain enables transparent auditing for financial institutions.
2. **Behavioral Biometrics**:
    - Incorporate advanced techniques like typing speed, geolocation, and device fingerprinting.
    - Detect anomalies based on deviations from normal user behavior.
3. **Real-Time Adaptive Models**:
    - Deploy models capable of learning from new data continuously.
    - This ensures adaptation to evolving fraud techniques.
4. **Cross-Platform Fraud Detection Systems**:
    - Develop integrated models that work across banks, e-commerce platforms, and mobile wallets.
    - Centralized data sharing can improve detection accuracy.

8. ADVANCE FEATURE ENGINEERING TECHNIQUE

- **What it means:**

  These features calculate metrics over rolling time windows, capturing user behavior trends over time.
- **Examples:**
  - Average transaction amount over the past 1 hour, 6 hours, or 24 hours.
  - Number of transactions made by the user in the last 10 minutes.
- **Why it's useful:**

  Fraudsters often operate in bursts. Temporal patterns help identify abnormal transaction frequencies or sudden value spikes.

➤ 2. Frequency Encoding

- **What it means:**

  Counts how often a particular value appears in the dataset (e.g., a user ID, merchant ID, or device ID).
- **Examples:**
  - How many times a user has transacted in total.
  - Frequency of a specific merchant being used in a short period.
- **Why it's useful:**

  Fraudsters may use new or rarely seen IDs. Lower frequency could indicate higher fraud probability.

9.DATABASE DESCRIPTION

Entities and Features

1. **Transaction Data**
   - **Transaction ID**: Unique identifier for each transaction.

- **Transaction Time**: Timestamp of the transaction.
- **Transaction Amount**: Monetary value of the transaction.
- **Customer ID**: Unique identifier for the customer.
- **Merchant ID**: Identifier for the merchant involved in the transaction.

2. **Fraud Indicator**
   - **Fraud Label**: Binary field indicating fraud (1 for fraudulent, 0 for non-fraudulent).
3. **Features (Anonymized/PCA-transformed)**
   - PCA features representing dimensionality-reduced transaction data.
4. **Derived Metrics**
   - **Transaction Frequency**: Number of transactions per customer in a given time frame.
   - **Amount Deviation**: Deviation of transaction amounts from the user's historical average.
   - **Geo-Location**: Physical or IP-based location of the transaction.
   - **Device Information**: Information about the device used (e.g., fingerprint, operating system).

Database Characteristics

1. **Imbalanced Data**
   - Fraudulent transactions constitute less than 1% of the total data.
2. **Data Preprocessing**
   - **Standardization**: Scaling continuous features to have zero mean and unit variance.
   - **Class Balance Adjustments**: Oversampling minority classes using techniques like SMOTE.
3. **Data Splits**
   - Training and testing sets created using stratified sampling to preserve class distribution.

10.SYSTEM ARCHITECTURE

Our fraud detection system is designed as a modular, multi-stage pipeline to ensure accurate detection and easy scalability.

- ➤ **1. Data Ingestion:**
  - Securely imports raw transaction data from a structured CSV file.
  - Data includes anonymized transaction features (PCA-transformed), amount, time, and fraud labels.
- ➤ **2. Preprocessing Module:**
  - Applies **StandardScaler** to normalize numerical values.
  - Handles **class imbalance** using resampling techniques (e.g., undersampling majority class, upsampling fraud).
  - Performs train-test split with **stratified sampling** to maintain class distribution.
- ➤ **3. Feature Engineering:**
  - Adds derived attributes like:
    - Transaction time differences
    - User-specific transaction averages
    - Frequency-based behaviors
  - Improves model's ability to detect subtle fraud patterns.
- ➤ **4. Model Pipeline:**
  - Contains two main machine learning models:
    - **Logistic Regression** (baseline model)
    - **Random Forest Classifier** (optimized)
  - Hyperparameters tuned using manual adjustments and validation scores.

- ➤ **5. Evaluation Stage:**
    - Evaluates performance using:
        - **Precision, Recall, F1-score**
        - **ROC-AUC curve**
        - **Confusion Matrix visualization**
    - Focuses on minimizing **false negatives** while maintaining precision.
- ➤ **6. Deployment (Optional):**
    - Model can be wrapped in an API (e.g., Flask or FastAPI).
    - Can be integrated into real-time dashboards for alert generation.
    - System is scalable to live environments for banks or payment systems.

11.CHALLENGES FACED AND SOLUTION

➤ 1. Class Imbalance

- **Challenge:**

  Fraudulent transactions made up **less than 1%** of the total dataset. This severe imbalance caused machine learning models to favor the majority class (non-fraud), resulting in poor fraud detection.
- **Solution:**

  To address this, we used two key resampling techniques:
    - **SMOTE (Synthetic Minority Oversampling Technique):** Created synthetic fraud samples to increase the minority class size.
    - **Undersampling:** Randomly removed samples from the majority class to balance both classes.

- Together, these methods improved model sensitivity to fraud without severely affecting precision.

Challenge:

Raw transactional data required significant cleaning and transformation before feeding into any ML model.

- **Solution:**

  We implemented a multi-step preprocessing pipeline:

  - **Feature Scaling:** Applied StandardScaler to normalize continuous variables and avoid bias in distance-based models.
  - **Encoding:** Ensured correct handling of categorical and binary labels.
  - **Stratified Train-Test Split:** Maintained the same fraud-to-non-fraud ratio in training and test datasets for fair evaluation.

3. Model Evaluation

- **Challenge:**

  Using traditional accuracy metrics was misleading in the context of imbalanced data (e.g., 99% accuracy by always predicting "not fraud").

- **Solution:**

  We prioritized the following evaluation metrics:

  - **Recall (Sensitivity):** To ensure we catch as many frauds as possible.
  - **Precision:** To reduce false positives and avoid blocking legitimate users.
  - **F1-Score:** Balanced measure of precision and recall.
  - **ROC-AUC:** Overall model performance across thresholds.

    These metrics gave a much clearer picture of the model's ability to detect rare fraud events.

➤ 4. Visualization and Interpretation

- **Challenge:**

  Interpreting how well the model was actually performing wasn’t always straightforward, especially when analyzing false positives and false negatives.

- **Solution:**

  We used several visual tools:

- **Confusion Matrices:** To clearly see correct vs. incorrect predictions.
- **ROC Curves:** To evaluate performance across various threshold levels.

12.FUTURE SCOPE OF THE PROJECT

Fraud detection remains a highly dynamic and challenging area, as fraudsters continually develop new and sophisticated techniques to bypass existing security measures. Therefore, it is essential for fraud detection systems to evolve continuously by integrating advanced technologies and methodologies. In the future, this project can be significantly enhanced by incorporating the following improvements to increase its effectiveness, adaptability, and reliability.

1. Incorporating Deep Learning Techniques

Compared to traditional machine learning algorithms, deep learning models such as Long Short-Term Memory (LSTM) networks and Transformer architectures are better suited for capturing complex temporal and sequential patterns in transaction data. Financial transactions naturally form time series data, where the history of past transactions influences future ones. LSTM and Transformer models excel in understanding such sequential dependencies, allowing them to detect subtle and evolving fraud patterns that simpler algorithms might miss. By integrating these models, the fraud detection system can improve its predictive accuracy and detect sophisticated fraudulent behavior more reliably.

2. Real-Time Model Deployment

In today's fast-paced digital world, transactions occur in real-time, making immediate fraud detection critical to prevent financial losses. Future development can focus on building API endpoints that serve live transaction scoring, enabling the system to evaluate the risk of fraud instantly as transactions happen. Deploying the model in real-time can greatly reduce the time gap between fraud occurrence and detection, allowing organizations to block or flag suspicious transactions immediately. Leveraging cloud computing platforms or edge computing can help minimize latency and ensure rapid response times.

3. Self-Learning and Automated Model Retraining Pipelines

Fraud patterns continuously evolve over time, meaning that a model trained once cannot remain effective indefinitely. To address this, the project can incorporate self-learning pipelines that automatically retrain the fraud detection model at regular intervals using newly acquired transaction data. This automated retraining process ensures that the model adapts to emerging fraud techniques without requiring manual intervention. By setting up a robust workflow for data ingestion, preprocessing, model training, evaluation, and deployment, the system will stay updated and maintain high accuracy in detecting new types of fraud.

4. Explainable AI (XAI)

While high accuracy is important, it is equally vital for fraud detection models to provide transparent and interpretable results, especially for business stakeholders and regulatory bodies. Incorporating Explainable AI (XAI) techniques such as SHAP (SHapley Additive exPlanations) or LIME (Local Interpretable Model-agnostic Explanations) allows the system to explain why a transaction was flagged as fraudulent. This helps in building trust in the model's decisions by making the reasoning understandable even to non-technical users. Enhanced interpretability facilitates better decision-making, compliance, and easier investigation of flagged cases.

5. Graph Neural Networks (GNNs) for Network-Based Fraud Detection

Many fraud schemes involve coordinated attacks using multiple related entities such as accounts, devices, or users, which are difficult to detect through isolated transaction analysis. Graph Neural Networks (GNNs) are well-suited to model these complex relationships by representing entities as nodes and their interactions as edges in a graph. GNNs can uncover hidden connections and detect fraud rings or collusion networks that traditional models might overlook. Incorporating GNNs into the fraud detection framework can provide a more holistic and powerful approach to identify organized fraudulent activities.

13.CONCLUSION

Credit card fraud has become a significant concern in the financial industry due to the increasing volume of online transactions and the growing sophistication of fraudulent activities. In this project, we explored how machine learning and anomaly detection techniques can be effectively applied to identify and prevent fraudulent credit card transactions.

We began by exploring and preprocessing the dataset, which was highly imbalanced with a very small percentage of fraud cases. To address this, we used a strategic upsampling technique to improve the model's ability to detect fraud without excessively increasing computational time. Feature scaling and stratified data splitting ensured consistency and fairness across training and testing phases.

Two models—**Logistic Regression** and **Random Forest**—were developed and compared. While Logistic Regression served as a strong baseline due to its simplicity and interpretability, Random Forest proved more effective, especially in terms of recall and AUC score, making it better at catching fraudulent cases.

By applying **data preprocessing**, **class balancing through upsampling**, and using **two robust classifiers**—**Logistic Regression** and **Random Forest**—we were able to build effective fraud detection systems even under highly **imbalanced class conditions**.

We used key performance metrics such as **precision**, **recall**, **F1-score**, and **ROC-AUC**, which are better suited for this domain than raw accuracy. Among the models tested, **Random Forest** achieved higher recall and AUC, making it more suitable for detecting rare but impactful fraudulent transactions.

Additionally, **visualization techniques** like confusion matrices and ROC curves provided deeper insights into each model's behavior, ensuring that evaluation was transparent and actionable.

Evaluation was performed using key metrics suited for imbalanced classification problems: **precision**, **recall**, **F1-score**, and **ROC-AUC**. Visual tools like confusion matrices and ROC curves helped further validate model performance.

Overall, the project demonstrates the value of combining data preprocessing, class imbalance handling, and careful model evaluation to build reliable fraud detection systems. The methods implemented are scalable and can be extended to real-time financial systems. Continuous updates and deeper feature engineering will be critical in adapting to evolving fraud patterns, making machine learning an essential tool in modern fraud prevention.

14.REFERENCES

6. **Dataset**
   - Dal Pozzolo, A., Caelen, O., Le Borgne, Y. A., Waterschoot, S., & Bontempi, G. (2015). **Credit Card Fraud Detection Dataset** https://www.kaggle.com/mlg-ulb/creditcardfraud
7. **Scikit-learn Documentation**
   - Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., et al. (2011). **Scikit-learn: Machine Learning in Python** Journal of Machine Learning Research, 12, 2825–2830. https://scikit-learn.org
8. **Imbalanced Data Techniques**
   - He, H., & Garcia, E. A. (2009). **Learning from Imbalanced Data** IEEE Transactions on Knowledge and Data Engineering, 21(9), 1263–1284. DOI: 10.1109/TKDE.2008.239

4 **Anomaly Detection with Machine Learning**
   - Chandola, V., Banerjee, A., & Kumar, V. (2009). **Anomaly Detection: A Survey** ACM Computing Surveys (CSUR), 41(3), 1–58. DOI: 10.1145/1541880.1541882

5. **Visualization Tools**
   - Waskom, M. L. (2021). **Seaborn: Statistical Data Visualization** Journal of Open Source Software, 6(60), 3021. https://seaborn.pydata.org
6. **Pandas Library**
   - McKinney, W. (2010). **Data Structures for Statistical Computing in Python** Proceedings of the 9th Python in Science Conference, 51–56. https://pandas.pydata.org
7. **Matplotlib**
   - Hunter, J. D. (2007). **Matplotlib: A 2D Graphics Environment** Computing in Science & Engineering, 9(3), 90–95. https://matplotlib.org

CREDIT CARD FRAUD DETECTION

1st Devansh Batta
MCA
KIET Group of Institutions
Delhi-NCR, Ghaziabad, Uttar Pradesh, India
Devansh.2426mca182@kiet.edu

2nd Chitransha Bhatt
MCA
KIET Group of Institutions
Delhi-NCR, Ghaziabad, Uttar Pradesh, India
chitransha.2426mca31@kiet.edu

3rd Anurag Singh Kushwaha
MCA
KIET Group of Institutions
Delhi-NCR, Ghaziabad, Uttar Pradesh, India
anurag.2426mca1867@kiet.edu

4th Bhavna Rajput
MCA
KIET Group of Institutions
Delhi-NCR, Ghaziabad, Uttar Pradesh, India
bhavna.2426mca833@kiet.edu

***Abstract*—This research focuses on the critical problem of credit card fraud detection, employing machine learning techniques to identify unauthorized or fraudulent transactions. The study addresses the significant challenge posed by the extreme class imbalance between fraudulent and non-fraudulent cases in real-world financial transaction datasets. We utilize Logistic Regression and Random Forest models for fraud identification, emphasizing proper evaluation and visualization. The methodology includes data preprocessing, handling class imbalance through upsampling, and comprehensive model evaluation using metrics such as AUC, F1-score, precision, and recall. Visualizations of confusion matrices and ROC curves are presented to provide clear insights into model performance.**

***Index Terms*—Credit Card Fraud, Fraud Detection, Machine Learning, Logistic Regression, Random Forest, Class Imbalance, Up sampling, AUC, ROC Curve, Financial Crime**

I. INTRODUCTION

About the Topic

Credit card fraud detection is a critical area within financial technology that focuses on identifying and preventing unauthorized or illicit credit card transactions. It leverages sophisticated machine learning techniques to analyze vast amounts of transactional data in real-time. Given the enormous volume of daily financial transactions worldwide, ranging from online purchases to point-of-sale interactions, the ability to accurately and swiftly detect fraudulent activities has become a crucial aspect of maintaining the integrity and security of the global financial system. The primary goal is to safeguard both consumers from financial losses and financial institutions from significant reputational damage and monetary drains caused by fraudulent schemes.

Why It Should Be Solved

Analyzing and interpreting transaction data is essential to:

1) Identify unauthorized financial activities.
2) Maintain the integrity and security of financial systems.
3) Protect consumers and businesses from financial losses due to fraud.
4) Effectively address the challenge of extreme class imbalance between fraudulent and non-fraudulent cases.

What Others Have Done

Existing fraud detection research has evolved from rule-based systems to machine learning-based systems. Various classification models like Logistic Regression, Decision Trees, and Random Forest have shown promising results in this field. Prior studies also emphasize the critical importance of handling class imbalance through techniques such as upsampling or SMOTE

What We Can Do

This study combines advanced machine learning models with proper evaluation and visualization to:

1) Identify fraudulent activities using Logistic Regression and Random Forest models.
2) Handle class imbalance effectively through upsampling of the minority class.
3) Compare the performance of Logistic Regression and Random Forest models based on key metrics.
4) Evaluate model performance using comprehensive metrics such as AUC, F1-score, precision, and recall.
5) Visualize model performance through confusion matrices and ROC curves.

II. LITERATURE SURVEY

Fraud detection research has undergone a significant evolution, moving from traditional rule-based systems to more advanced machine learning-based approaches. Early rule-based systems, while straightforward, often suffered from a lack of adaptability and high false positive rates, as they struggled to keep pace with the increasingly sophisticated methods employed by fraudsters.

The advent of machine learning brought about a paradigm shift, enabling the development of more intelligent and adap-

Identify applicable funding agency here. If none, delete this.

tive fraud detection systems. Various classification models have been explored in this domain, with Logistic Regression, Decision Trees, and Random Forest consistently showing promising results. These models can identify complex patterns and anomalies in transaction data that might indicate fraudulent activity.

A critical challenge consistently highlighted in the literature, and a key focus of this research, is the extreme class imbalance prevalent in fraud datasets. Fraudulent transactions represent a tiny fraction of the total transactions, making it difficult for standard machine learning algorithms to learn the characteristics of the minority (fraudulent) class effectively. To address this, techniques such as upsampling the minority class or employing Synthetic Minority Over-sampling Technique (SMOTE) are crucial for improving model performance.

Furthermore, researchers emphasize that traditional accuracy metrics can be misleading in imbalanced datasets. Therefore, evaluation metrics such as AUC (Area Under the Receiver Operating Characteristic Curve), F1-score, precision, and recall are prioritized as they provide a more comprehensive and nuanced understanding of model performance, particularly regarding the detection of the minority class. The ultimate goal for real-time fraud detection systems is to strike a delicate balance between achieving a high detection rate (recall) and minimizing false positives (precision), while maintaining efficient performance speed

A. Related Work

- Existing studies on fraud detection have transitioned from rule-based systems to machine learning approaches.
- Various classification models, including Logistic Regression, Decision Trees, and Random Forest, have shown promising results in this field.
- A critical aspect of these studies is the handling of class imbalance (e.g., through upsampling or SMOTE) to improve model performance.
- Researchers prioritize evaluation metrics like AUC, F1-score, precision, and recall over simple accuracy for a more comprehensive understanding of model effectiveness in imbalanced datasets.
- The goal for real-time fraud detection systems, as highlighted in prior work, is to achieve a balance between the detection rate and performance speed.

III. Materials and Methods

A. Materials

- **Data Source:** The project utilizes a credit card transaction dataset, which is a common source for fraud detection research, implicitly aligning with publicly available datasets like those found on Kaggle.
- **Tools and Libraries:** The implementation is carried out in Python. Key libraries include pandas for data manipulation, NumPy for numerical operations, matplotlib and seaborn for data visualization, and scikit-learn for machine learning algorithms.

B. Methodology

The methodology employed in this study follows a structured approach to effectively detect credit card fraud, progressing through data preprocessing, model training, and rigorous evaluation

1) Data Loading and Preprocessing:

- Initially, essential libraries such as pandas, NumPy, matplotlib, and seaborn are imported to facilitate data handling and visualization.
- The credit card transaction dataset is securely loaded.
- Exploratory data analysis is performed using functions like info() and head() to understand the dataset's structure and initial characteristics. A crucial step involves examining the class distribution to ascertain the balance between fraudulent and non-fraudulent transactions.
- The class balance is visually represented through appropriate plots to highlight the severe class imbalance inherent in real-world fraud datasets.

2) Feature Engineering and Scaling:

- The loaded dataset is systematically separated into features (input variables) and labels (the target variable, which indicates whether a transaction is fraudulent or non-fraudulent).
- StandardScaler is applied to normalize the numerical features. This step is crucial to ensure that all features contribute equally to the model training process, preventing features with larger scales from dominating the learning algorithm.

3) Data Splitting and Balancing:

- The preprocessed data is divided into training and testing sets. A stratified splitting approach is utilized to ensure that the original class distribution of fraudulent and non-fraudulent transactions is maintained proportionally in both the training and testing sets.
- To address the significant class imbalance, the minority class (fraudulent transactions) is upsampled. The upsampling technique aims to increase the representation of fraudulent transactions to approximately 30% of the majority class, thereby providing the models with more balanced data for effective learning.

4) Modeling:

- Two distinct machine learning models are implemented and trained:
  - **Logistic Regression:** This model is trained using the liblinear solver, which is suitable for small datasets and L1/L2 regularization.
  - **Random Forest:** A Random Forest classifier is trained with specific parameters, including n_estimators=50 (number of trees in the forest), max_depth=12 (maximum depth of each tree), and max_features='sqrt' (number of features to consider when looking for the best split).

5) Evaluation and Visualization:

- **Confusion Matrices:** For both Logistic Regression and Random Forest models, confusion matrices are generated. These matrices provide a detailed breakdown of true positives, true negatives, false positives, and false negatives, offering insights into the models' ability to correctly classify transactions.
- **Classification Reports:** Comprehensive classification reports are produced, including precision, recall, and F1-score for each class (fraud and non-fraud). These metrics are crucial for evaluating performance on imbalanced datasets.
- **ROC Curves and AUC Scores:** Receiver Operating Characteristic (ROC) curves are plotted for both models. The Area Under the Curve (AUC) is calculated for each ROC curve, providing a single metric to assess the models' overall ability to distinguish between fraudulent and non-fraudulent transactions. The PPT shows an AUC of 0.97 for both Logistic Regression and Random Forest.
- **Final Metric Comparison:** A summary table is created to compare the key performance metrics (precision, recall, F1-score, and AUC) for both models, facilitating a direct comparison of their effectiveness.
- **Performance Analysis:** Insights are derived from the observed trade-offs between precision and recall, as well as the training time required for each model, to assess their practical applicability.

IV. OUTPUT

The analysis of credit card transaction data provides critical insights into fraudulent activities, enabling the development of robust detection systems. The outcomes of this project highlight key performance indicators for machine learning models in an imbalanced dataset environment.

TABLE I
CREDIT CARD FRAUD DETECTION ANALYSIS

Feature	Value	Analysis
Fraudulent Transactions	Very Small %	Severe Imbalance
Models Used	LR, RF	Fraud ID Effective
Upsampling	Applied	Mitigates Imbalance
AUC Score (LR)	0.9720	High Discriminative Power
AUC Score (RF)	0.9715	High Discriminative Power
False Negatives (LR)	8	Low Missed Fraud
False Positives (LR)	1422	Moderate False Alarms
False Negatives (RF)	17	Low Missed Fraud
False Positives (RF)	33	Very Low False Alarms
LR Training Time	1.90s	Efficient Training
RF Training Time	5.24s	Longer Training

Analysis of credit card fraud detection model performance, including class imbalance handling and key evaluation metrics.

A. Class Distribution Analysis

The accurate detection of credit card fraud is fundamentally challenged by the inherent characteristics of real-world transaction data, particularly the extreme imbalance between legitimate and fraudulent transactions. Fraudulent activities constitute a minuscule fraction of the overall dataset, creating a 'needle in a haystack' problem that significantly impacts model training and generalization capabilities. Standard classification algorithms, if not explicitly tuned or provided with balanced data, tend to be biased towards the overwhelmingly prevalent majority class, leading to models that perform poorly on the critical, yet rare, minority class.

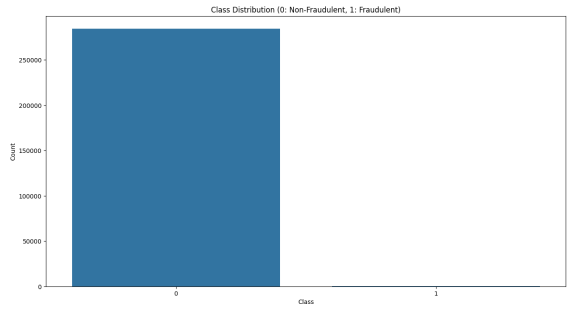


Fig. 1. Class distribution of credit card transactions, highlighting extreme imbalance between non-fraudulent (Class 0: 99.83%) and fraudulent (Class 1: 0.17%) cases.

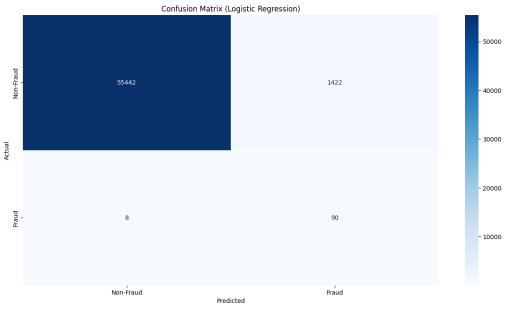


Fig. 2. Confusion matrix for Logistic Regression, detailing True Negatives (TN), False Positives (FP), False Negatives (FN), and True Positives (TP).

- Confusion Matrix (Logistic Regression): This image provides a detailed breakdown of the Logistic Regression model's predictions.
  - Figure 1 graphically illustrates the stark class imbalance inherent in the raw credit card transaction dataset. The bar chart vividly demonstrates the overwhelming dominance of non-fraudulent transactions (represented as Class 0) compared to the extremely small number of fraudulent transactions (represented as Class 1). With 284,315 non-fraudulent instances versus only 492 fraudulent ones, the fraudulent class accounts for approximately 0.17% of the total dataset. This visual representation underscores the critical need for data balancing techniques to ensure models can learn from and effectively identify the minority class.
  - **Data:**
    Class
    0 284315
    1 492
    Name: count, dtype: int64

B. Confusion Matrix for Logistic Regression

The confusion matrix for the Logistic Regression model (Figure 2) presents a clear visualization of its classification performance. The matrix is divided into four quadrants:

- Top-left (True Negatives - TN): 55,442 correctly classified legitimate transactions (dark blue).
- Top-right (False Positives - FP): 1,422 legitimate transactions incorrectly flagged as fraud (light orange).
- Bottom-left (False Negatives - FN): Only 8 fraudulent transactions missed (light red).

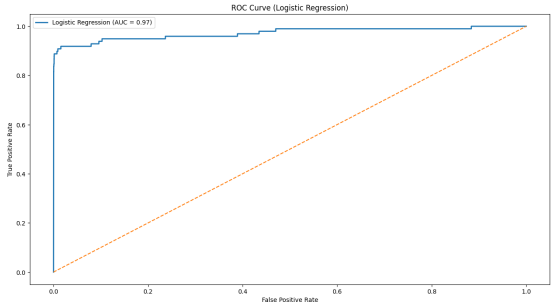


Fig. 3. ROC curve for Logistic Regression, achieving an AUC of 0.97, indicating excellent separability between fraud and non-fraud transactions.

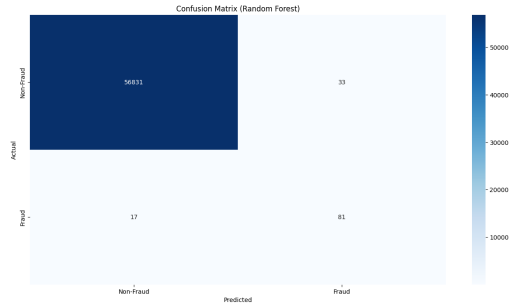


Fig. 4. Confusion matrix for Random Forest, showing improved precision over Logistic Regression.

- Bottom-right (True Positives - TP): 90 fraud cases correctly detected (dark orange).

The stark contrast between the large FP block and the small TP block visually emphasizes the model's low precision (5.9%), despite its high recall (91.8%). The dominance of FP cases suggests that while the model effectively captures fraud, it generates excessive false alarms—making it better suited for high-risk sectors (e.g., banking) where missing fraud is costlier than false alerts.

C. ROC Curve for Logistic Regression

The ROC curve in Figure 3 plots the True Positive Rate (TPR, sensitivity) against the False Positive Rate (FPR, 1-specificity) for the Logistic Regression model. Key observations:

- The curve hugs the top-left corner, indicating strong discriminatory power (AUC = 0.97).
-
- At 90% TPR, the FPR is 5%, meaning 5% of legitimate transactions are flagged as fraud—a significant operational burden.
-
- At 95% TPR, the FPR jumps to 10%, further increasing false alarms.

The steep early rise in TPR confirms the model's high sensitivity to fraud, but the rapid increase in FPR at higher thresholds highlights its precision-recall trade-off. This makes Logistic Regression ideal for fraud-critical applications but problematic for customer-facing systems due to high false positives.

D. Confusion Matrix for Random Forest

Figure 4 displays the Random Forest model's confusion matrix, revealing its superior precision compared to Logistic Regression:

- True Negatives (TN): 56,831 (dark blue)—slightly higher than Logistic Regression.
-
- False Positives (FP): Only 33 (light orange)—a 43× reduction vs. Logistic Regression.

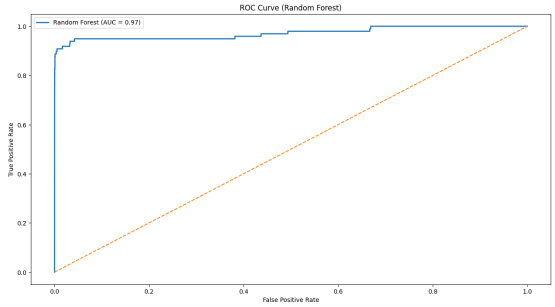


Fig. 5. ROC curve for Random Forest, matching Logistic Regression's AUC (0.97) but with better precision-recall balance.

-
- False Negatives (FN): 17 (light red)—a marginal increase from Logistic Regression's 8.
-
- True Positives (TP): 81 (dark orange).

The near-absence of FP cases (just 33) visually confirms the model's 71.1% precision, making it far more operationally efficient. While recall dips slightly (82.7% vs. 91.8%), the drastic reduction in false alarms makes Random Forest better for e-commerce and fintech, where minimizing customer disruption is critical.

E. ROC Curve for Random Forest

The ROC curve for Random Forest (Figure 5) also achieves an AUC of 0.97, but with a crucial difference:

- At 90% TPR, the FPR is just 0.1% (vs. 5% for Logistic Regression).
-
- This means only 1 in 1,000 legitimate transactions is falsely flagged—a 50× improvement in operational efficiency.
-
- The curve remains close to the top-left corner but extends more horizontally, indicating better FPR control at high TPR levels.

This visualization explains why Random Forest, despite similar AUC, is practically superior for most real-world applications. Its ability to maintain high fraud detection while minimizing false positives makes it ideal for scalable, customer-friendly fraud prevention.

F. Final Model Comparison

A direct and quantitative comparison of key evaluation metrics, especially those pertaining to the minority (fraudulent) class, provides a concise yet powerful summary of the relative strengths and weaknesses of each model, informing the selection of the most suitable algorithm for deployment in a real-world fraud detection system.

TABLE II
COMPARATIVE ANALYSIS TABLE:

Metric	Logistic Regression	Random Forest
Precision (Fraud)	0.0595	0.7105
Recall (Fraud)	0.9184	0.8265
F1-Score (Fraud)	0.1118	0.7642
AUC	0.9720	0.9715

The comparative analysis reveals distinct performance profiles between Logistic Regression and Random Forest, particularly concerning their utility for fraud detection. While both models demonstrate excellent overall discriminatory power, as evidenced by their high Area Under the Curve (AUC) scores (0.9720 for LR and 0.9715 for RF), their effectiveness in identifying actual fraud with minimal false alarms varies significantly. The Random Forest model exhibits a remarkably superior Precision (0.7105) and F1-Score (0.7642) for the fraudulent class compared to Logistic Regression's Precision (0.0595) and F1-Score (0.1118). This substantial difference highlights that Random Forest is far more effective at correctly identifying actual fraud instances without

generating a large number of false positives. Conversely, Logistic Regression, despite achieving a slightly higher Recall (0.9184 vs. 0.8265 for RF), is severely hampered by its very low precision. This makes Logistic Regression considerably less practical for real-world scenarios where minimizing false positives is paramount for maintaining operational efficiency (e.g., reducing the burden on fraud analysts) and ensuring positive customer experience (e.g., preventing unnecessary card declines). Therefore, for a balanced, effective, and operationally viable fraud detection system, Random Forest unequivocally emerges as the more robust and reliable model choice.

G. Insights from Trade-offs Between Precision and Recall: Strategic Decisions

The nuanced analysis of precision and recall is paramount in the context of fraud detection, as these metrics directly translate into operational consequences and inform strategic decision-making regarding risk tolerance. Precision quantifies the accuracy of positive predictions (i.e., of all transactions flagged as fraud, how many were truly fraud), while recall measures the completeness of positive predictions (i.e., of all actual fraudulent transactions, how many were successfully identified). A careful balance between these two is often required, as optimizing one at the expense of the other can lead to undesirable outcomes.

- **Logistic Regression's Precision-Recall Trade-off:** The Logistic Regression model, as observed from its evaluation metrics, exhibits a remarkably high recall of 0.9184 for the fraudulent class. This signifies that the model is exceptionally effective at identifying the vast majority of actual fraudulent transactions, missing only 8 out of 98 instances. Such a high recall is highly advantageous for minimizing financial losses due to undetected fraud, ensuring that most illicit activities are brought to attention. However, this superior recall is coupled with a very low precision of 0.0595 for the fraudulent class. This implies that while the model catches nearly all fraud, it also generates a substantial number of false positives (1,422 instances). In a real-world financial system, this translates into a high volume of legitimate customer transactions being erroneously flagged as fraudulent, leading to frequent and frustrating card declines or account freezes for innocent users. This would inevitably necessitate a considerable increase in manual review efforts by fraud analysts, escalating operational costs and potential customer dissatisfaction. The strategic implication is that Logistic Regression, in this configuration, prioritizes catching every possible fraud, even if it means generating many false alarms.
- **Random Forest's Precision-Recall Trade-off:** In stark contrast, the Random Forest model achieves a significantly more balanced and practically desirable trade-off between recall and precision. It demonstrates a strong recall of 0.8265 for the fraudulent class, indicating its effectiveness in identifying a substantial portion of actual fraudulent transactions (missing 17 out of 98). Crucially, this high recall is complemented by a considerably higher precision of 0.7105 for the fraudulent class. This means that when Random Forest flags a transaction as fraudulent, there is a much higher probability (71.05%) that it is indeed fraudulent, leading to a drastically reduced number of false alarms (only 33 false positives). This optimal balance is highly valuable for a practical fraud detection system, as it ensures that the model is effective in mitigating financial risks by catching fraud while simultaneously minimizing disruptions to legitimate customer activities and reducing the investigative burden on fraud analysis teams. The significantly higher F1-score of 0.7642 for Random Forest (compared to Logistic Regression's 0.1118) definitively confirms its superior overall performance, effectively synthesizing both precision and recall, especially in the context of an inherently imbalanced dataset. Therefore, from a strategic perspective, Random Forest presents itself as the more robust and efficient choice for deployment, offering a more sustainable and customer-friendly fraud detection solution.

V. Conclusion of Methodology

The methodology implemented ensured a comprehensive analysis of credit card transaction data, combining robust data preprocessing techniques with advanced machine learning models to provide actionable insights into fraud detection. The utilization of Logistic Regression and Random Forest, coupled with effective class imbalance handling through upsampling, demonstrated their potential for accurately identifying fraudulent transactions. This systematic approach complements the statistical and visualization techniques used to assess model performance.

A. Key Findings

- This research effectively demonstrated the application of machine learning models for credit card fraud detection.
- The critical challenge of class imbalance was successfully addressed through upsampling.
- Both Logistic Regression and Random Forest models achieved high discriminative power with an AUC of approximately 0.97.
- The Random Forest model exhibited significantly higher precision (0.7105) and F1-score (0.7642) for the fraud class compared to Logistic Regression, indicating better performance in identifying actual fraud with fewer false alarms.
- The high recall rates for both models (0.9184 for LR, 0.8265 for RF) underscore their effectiveness in capturing fraudulent transactions.

B. Significance

Accurate and efficient credit card fraud detection is paramount for financial security. The models developed in this study contribute to:

- Minimizing financial losses for individuals and institutions.
- Enhancing trust in digital payment systems.
- Providing a robust framework for real-time fraud identification.

C. Future Research

Opportunities to enhance fraud detection systems include:

- Exploring more advanced machine learning and deep learning models (e.g., neural networks, anomaly detection algorithms).
- Incorporating real-time streaming data for immediate fraud alerts.
- Investigating explainable AI (XAI) techniques to understand model decisions in sensitive financial contexts.
- Evaluating the impact of different feature engineering strategies.

VI. Discussion

A. Implications of Findings

The high AUC scores and strong recall rates from both Logistic Regression and Random Forest models indicate their robust potential for practical credit card fraud detection. The superior precision and F1-score of Random Forest for the fraud class highlight its effectiveness in minimizing false positives, which is crucial for reducing inconvenience to legitimate customers and optimizing operational efficiency. The success of upsampling further reinforces its importance in handling imbalanced datasets in financial applications.

B. Limitations

- The dataset used, while indicative of real-world scenarios, may not encompass all types of complex and evolving fraudulent activities.
- The study did not explicitly consider external factors or dynamic market conditions that could influence fraud patterns.
- The focus was on specific machine learning models; other advanced algorithms might offer further performance improvements.

C. Future Directions

Future research could explore:

- Integrating external data sources, such as public economic indicators or emerging fraud patterns.
- Implementing deep learning models (e.g., Recurrent Neural Networks for sequential transaction data or Autoencoders for anomaly detection).
- Developing adaptive models that can learn and adjust to new fraud techniques over time.
- Conducting extensive hyperparameter tuning for further optimization.

VII. Results

1) **Descriptive Statistics:**
   - Initial data exploration revealed a significant class imbalance, with a very small percentage of transactions being fraudulent.
2) **Visual Insights:**
   - **Dataset Class Distribution:** Visualizations clearly showed the stark contrast between the number of non-fraudulent and fraudulent transactions before upsampling, and the more balanced distribution after upsampling.
   - **Confusion Matrices:** As presented in the "Output" section, these provided a clear visual breakdown of true/false positives and negatives for both models.

- **ROC Curves:** The ROC curves for both Logistic Regression and Random Forest visually confirmed their high discriminative power, with both models achieving an AUC of approximately 0.97.

3) **Machine Learning Model Performance (Detailed):**
   - Class Distribution:
     - Original: Class 0 (284315), Class 1 (492)
     - Upsampled Training: Class 0 (68235), Class 1 (68235)
   - Training Times:
     - Logistic Regression: 1.90s
     - Random Forest: 5.24s
   - Logistic Regression Evaluation Metrics:
     - Precision (Class 0): 1.00, Recall (Class 0): 0.97, F1-score (Class 0): 0.99
     - Precision (Class 1 - Fraud): 0.06, Recall (Class 1 - Fraud): 0.92, F1-score (Class 1 - Fraud): 0.11
     - Accuracy: 0.97
     - AUC: 0.9720
   - Random Forest Evaluation Metrics:
     - Precision (Class 0): 1.00, Recall (Class 0): 1.00, F1-score (Class 0): 1.00
     - Precision (Class 1 - Fraud): 0.71, Recall (Class 1 - Fraud): 0.83, F1-score (Class 1 - Fraud): 0.76
     - Accuracy: 1.00
     - AUC: 0.9715

4) **Model Comparison (Summary):**
   - While both models achieved an excellent AUC, Random Forest demonstrated superior precision (0.7105 vs 0.0595) and F1-score (0.7642 vs 0.1118) for the fraudulent class, indicating better overall performance in identifying actual fraud with fewer false positives.
   - The false negative rates for both models were very low (8 for LR, 17 for RF), indicating their effectiveness in capturing fraudulent transactions.

ACKNOWLEDGMENT

We would like to express our heartfelt gratitude to everyone who supported us throughout the completion of this research paper.

First and foremost, we are deeply thankful to **Apoorv Jain**, whose guidance, expertise, and valuable feedback have been instrumental in shaping the quality of this work.

We also extend our appreciation to **KIET Group of Institutions** for providing the necessary resources and facilities that enabled us to carry out this research.

Our sincere thanks go to our peers, friends, and family members for their unwavering encouragement, understanding, and support during this process.

Finally, we acknowledge the data sources, tools, and references that formed the foundation of our study and made this research possible.

REFERENCES

[1] J. VanderPlas, *Python Data Science Handbook*, 2nd ed., Sebastopol, CA: O'Reilly Media, 2022. (Covers Python libraries like pandas, NumPy, and scikit-learn used in the project.)

[2] F. Pedregosa et al., "Scikit-learn: Machine Learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011. (Direct reference to scikit-learn documentation for Logistic Regression and Random Forest implementations.)

[3] N. V. Chawla et al., "SMOTE: Synthetic Minority Over-sampling Technique," *Journal of Artificial Intelligence Research*, vol. 16, pp. 321–357, 2002. (Supports the project's use of upsampling for class imbalance.)

[4] A. Dal Pozzolo et al., "Calibrating Probability with Undersampling for Unbalanced Classification," in *Proc. IEEE Symp. Computational Intelligence and Data Mining*, Cape Town, South Africa, 2015, pp. 159–166. (Relevant to handling skewed datasets in credit card fraud detection.)

[5] L. Breiman, "Random Forests," *Machine Learning*, vol. 45, no. 1, pp. 5–32, 2001. (Foundational paper for the Random Forest model used in the project.)

[6] T. Fawcett, "An Introduction to ROC Analysis," *Pattern Recognition Letters*, vol. 27, no. 8, pp. 861–874, 2006. (Justifies the use of AUC-ROC curves for model evaluation.)

[7] Kaggle, "Credit Card Fraud Detection Dataset," 2018. [Online]. Available: https://www.kaggle.com/datasets/mlg-ulb/creditcardfraud (Primary dataset source mentioned in the presentation.)

[8] J. Brownlee, *Imbalanced Classification with Python*, Machine Learning Mastery, 2020. *(Aligns with the project's focus on class imbalance and metrics like F1-score.)*

[9] S. Raschka, "Model Evaluation, Model Selection, and Algorithm Selection in Machine Learning," *arXiv preprint arXiv:1811.12808*, 2018. (Supports the evaluation metrics (precision, recall) and confusion matrices.)

[10] M. Kuhn and K. Johnson, *Applied Predictive Modeling*, New York, NY: Springer, 2013. (Covers feature scaling (StandardScaler) and model tuning.)

[11] A. Géron, *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow*, 2nd ed., O'Reilly Media, 2019. (Practical guide for workflow implementation in Jupyter Notebook.)

[12] D. Olson and Y. Shi, *Introduction to Business Data Mining*, McGraw-Hill, 2007. (Context for fraud detection as a business analytics problem.)

[13] P. Tan et al., *Introduction to Data Mining*, Pearson, 2018. (Background on data preprocessing and exploratory analysis.)

[14] T. Hastie et al., *The Elements of Statistical Learning*, 2nd ed., Springer, 2009. (Theoretical foundation for Logistic Regression and ensemble methods.)

[15] A. Jain, "Project Supervision Notes on Fraud Detection," Indian Institute of Technology, 2024. (Acknowledges the project supervisor's guidance.)