ZIDIO DEVELOPMENT

Financial Fraud Detection System: Project Report

-by Group-14

Sabahath Taj Z

Riya Yadav

This report details the development of a comprehensive Financial Fraud Detection System. The project successfully leveraged a combination of data preprocessing, supervised and unsupervised machine learning, and advanced analytical techniques to build a robust framework for identifying fraudulent transactions.

The project began with the acquisition and rigorous preprocessing of the "Credit Card Fraud Detection Dataset 2023" from Kaggle. This involved data cleaning, inspection, and feature engineering to prepare the data for modeling. Subsequently, a suite of machine learning models was developed and evaluated. **Supervised models, particularly ensemble methods like Random Forest and XGBoost, demonstrated exceptional performance, achieving accuracy levels exceeding 99%** on the balanced dataset.

In addition to supervised learning, unsupervised anomaly detection techniques, including Isolation Forest and a PCA-based reconstruction error model, were successfully implemented. These models provide a valuable secondary line of defense for identifying novel or unusual fraud patterns not seen in labeled training data. A conceptual demonstration of graph-based analysis was also conducted, showcasing how network relationships can be used to detect sophisticated fraud rings by creating and analyzing a synthetic transactional graph.

The project culminates in a design for a real-time monitoring system and an interactive dashboard, providing a clear path from data analysis to a deployable, operational tool.

1. Phase 1: Data Acquisition & Preprocessing

1.1. Objective

The primary goal of this phase was to perform ETL (Extract, Transform, Load) operations on the raw transactional data to create a clean, optimized, and feature-rich dataset ready for machine learning.

1.2. Dataset

- **Source:** Kaggle
- **Name:** Credit Card Fraud Detection Dataset 2023
- **Characteristics:** A large-scale dataset containing 568,630 transactions with 30 features. It is a balanced dataset with an equal number of legitimate (Class 0) and fraudulent (Class 1) transactions.

1.3. Methodology

1. **Data Loading & Inspection:** The dataset was loaded into a pandas DataFrame. Initial inspection revealed that it contained 31 columns, including an id column, 28 anonymized features (V1 to V28), an Amount feature, and the target Class variable.
2. **Data Cleaning:** The id column was identified as a simple row identifier with no predictive value and was subsequently dropped from the dataset.
3. **Missing Value Analysis:** The dataset was scanned for missing values. It was confirmed that the dataset was complete, and no imputation or handling of missing data was necessary.
4. **Data Normalization:** A review of the feature statistics (mean near 0, standard deviation near 1) and the dataset's documentation confirmed that all features, including Amount, were already pre-scaled. Therefore, no further normalization was required.
5. **Feature Engineering:** To enhance the model's ability to capture complex patterns, two new features were engineered:
    - V1_V2_interaction: A multiplicative interaction term (V1 * V2) to capture potential combined effects.
    - V_feature_sum: An additive feature (V3 + V4 + V5) to create a composite signal from several individual features.

1.4. Outcome

This phase concluded with the successful creation of a clean and engineered dataset, which was saved as **processed_transactions_2023.csv**. This file served as the single source of truth for all subsequent modeling phases.

---

2. Phase 2: Model Development & Evaluation

2.1. Supervised Learning Models

- **Objective:** To train and evaluate models that learn the direct relationship between transaction features and the "fraud" label.
- **Methodology:**
    - The preprocessed data was split into an 80% training set and a 20% testing set. Stratification was used to ensure the 50/50 class balance was maintained in both sets.

    - Four models were trained and evaluated: Logistic Regression, Random Forest, XGBoost, and LightGBM.
- **Results:** The ensemble models demonstrated outstanding performance, indicative of their ability to capture the complex, non-linear patterns in the data. The Random Forest, XGBoost, and LightGBM models all achieved near-perfect accuracy and high precision/recall scores.

Model	Accuracy	Precision (Fraud)	Recall (Fraud)	F1-Score (Fraud)
XGBoost	~99.8%	~1.00	~0.99	~1.00
LightGBM	~99.8%	~1.00	~0.99	~1.00
Random Forest	~99.7%	~1.00	~0.99	~0.99
Logistic Regression	~95.6%	~0.97	~0.94	~0.95

2.2. Unsupervised Anomaly Detection

- **Objective:** To identify fraudulent transactions by treating them as "anomalies" or "outliers" without using the Class label during training. This simulates the detection of new fraud types.
- **Methodology:**
    - **Isolation Forest:** Trained on the full dataset with contamination set to 0.5 to reflect the known 50% fraud rate.
    - **One-Class SVM:** Due to its high computational cost, this model was trained on a 5% random sample of the data.
- **Results:** Both models successfully identified fraudulent transactions with reasonable accuracy, though not as high as the supervised models. The Isolation Forest performed particularly well, demonstrating its effectiveness for this type of anomaly detection task.

2.3. PCA-Based Anomaly Detection (Autoencoder Alternative)

- **Objective:** To implement a deep learning-style anomaly detection system using a classical machine learning technique, avoiding dependency on complex libraries.
- **Methodology:**
    1. A PCA (Principal Component Analysis) model was trained **only on legitimate** transactions to learn their fundamental structure.
    2. This model was then used to reconstruct *all* transactions (both legitimate and fraudulent).
    3. The "reconstruction error" was calculated for each transaction. The core hypothesis is that the model will be poor at reconstructing fraudulent transactions, resulting in a high error.
    4. A threshold was set (95th percentile of the normal error) to classify transactions with high error as fraudulent.
- **Results:** This method proved highly effective. The visualization of reconstruction errors showed a clear separation between the error distributions of legitimate and fraudulent transactions, validating this approach as a powerful alternative to a neural network autoencoder.

---

3. Phase 3: Graph-Based Analysis (Conceptual)

- **Objective:** To demonstrate the methodology for detecting sophisticated fraud patterns, such as colluding networks or fraud rings, using graph analysis.
- **Methodology:**
    - As the primary dataset was anonymized and lacked linkable entities, a **synthetic dataset** was generated with customer_id and merchant_id columns.
    - A specific fraud pattern—a "fraud ring" involving 4 customers and 1 merchant—was programmatically injected into this synthetic data.
    - The networkx library was used to build a graph where customers and merchants were nodes and transactions were edges.
- **Results:**
    - **Visual Detection:** The visualization of the graph clearly exposed the fraud ring as a small, dense, and isolated cluster of nodes, visually distinct from the sprawling network of legitimate transactions.
    - **Programmatic Detection:** By analyzing the graph's "connected components," the script successfully and automatically identified the suspicious community,

proving the viability of this technique for automated fraud detection in non-anonymized datasets.

To demonstrate a practical application, the project pivoted to building a real-time, end-to-end prototype. A simplified two-feature model was trained for this purpose, serving as the predictive engine for a live pipeline built with **Apache Kafka**. This pipeline consists of a **Producer** simulating a stream of transactions, a **Consumer** that uses the ML model to make real-time predictions, and an automated **email alert system** for high-risk cases.

The final component is an interactive **Streamlit dashboard** that visualizes the real-time activity processed by the consumer, presenting live metrics, transaction logs, and summary charts. The project successfully meets all its initial objectives, culminating in a functional, proof-of-concept system that showcases the power of combining data science with real-time data engineering

---

Phase 3: Real-Time Fraud Monitoring with Apache Kafka

- **Objective:** To build a functional, end-to-end pipeline that simulates the detection of fraud as transactions happen.
- **Architecture:** The system was built with three core components communicating via a Kafka topic named transactions.

1. **Kafka Producer:**
    - A Python script was developed to act as a transaction simulator.
    - It reads sample transactions from a local transactions.json file.
    - It sends one transaction every 1-3 seconds to the transactions Kafka topic,mimicking a live data stream.
2. **Kafka Consumer & Alerter:**
    - A second, continuously running Python script subscribes to the transactions topic.
    - For each message received, the consumer performs the following actions:
        - Deserializes the JSON transaction data.
        - Transforms the data into the two-feature format required by the simplified model (amount, location).
        - Loads the pre-trained fraud_model.pkl.

        - Predicts if the transaction is fraudulent (1) or legitimate (0).
        - **If fraud is detected**, it calls an alert function to send a notification email via SMTP.
        - **Crucially, it appends the transaction and its prediction result to a file named** , which acts as a data source for the dashboard.
3. **Alerting System:**
    - The email alert function was integrated directly into the consumer. It uses the smtplib and email libraries to send a formatted alert, detailing the suspicious transaction.

Phase 4: Interactive Dashboard & Reporting

- **Objective:** To create a user-friendly, real-time interface for monitoring the system's activity.
- **Technology: Streamlit** was used to build the web-based dashboard.
- **Functionality:**
    1. **Live Data Feed:** The dashboard runs in a loop, automatically reading the dashboard_data.json file every 5 seconds. This allows it to display the latest results from the Kafka consumer without a direct connection.
    2. **Key Metrics:** It displays prominent metrics for "Total Transactions Processed", "Fraudulent Transactions Detected", and "Legitimate Transactions".
    3. **Live Transaction Log:** It shows a data table of the most recent transactions processed, with the latest appearing at the top.
    4. **Visualizations:** A bar chart visualizes the real-time distribution of fraudulent vs. legitimate transactions detected by the system

---

Conclusion

This project successfully achieved its goal of building a complete financial fraud detection system, from data analysis to a real-time operational prototype. The initial deep analysis validated the high predictability of the dataset, while the subsequent development of the Kafka pipeline demonstrated a practical application of the trained models. The final Streamlit dashboard effectively provides a "mission control" view of the system, making the results accessible and understandable.

The project highlights the critical importance of a phased approach: beginning with thorough analysis to build confidence and understanding, followed by targeted, practical implementation for a real-world use case. The final system stands as a robust proof-of-concept for a scalable, real-time fraud detection and alerting platform.

Future enhancements could include:

- **AI-driven Risk Scoring:** Developing more nuanced risk scores for customers.
- **Blockchain Integration:** Exploring blockchain technology for a transparent and tamper-proof transaction ledger.
- **Mobile App Integration:** Providing instant fraud alerts and verification prompts directly to customers' mobile devices.