Real-Time Fraud Detection Using Machine Learning and Distributed Systems

Mayur Jain
Department of Computer Science
California State University, Long Beach
California, USA

***Abstract*—Fraudulent financial transactions are a pervasive issue in modern economies. This paper presents a comprehensive real-time fraud detection system leveraging logistic regression, feature extraction, and pattern recognition techniques. The system utilizes Apache Kafka and Spark for distributed data streaming and processing, achieving high scalability and low latency. A user-friendly dashboard provides actionable insights, while results demonstrate robust performance in fraud identification with a ROC-AUC score of 0.97. Future enhancements include integrating deep learning models and additional data sources.**

I. Introduction

The rapid increase in digital financial transactions has led to a proportional rise in credit card fraud, posing serious challenges to financial institutions. Traditional fraud detection methods, typically rule-based, struggle to identify sophisticated fraud patterns, as these methods are not adaptable and often fail to catch emerging fraud tactics. To address this gap, our study introduces a real-time fraud detection framework that leverages a combination of distributed data streaming and machine learning. By integrating Apache Kafka and Apache Spark, the system can process and analyze data in real-time, providing the necessary speed and scalability required by high-transaction environments. A logistic regression model forms the core of the detection mechanism, allowing for a supervised approach that learns from historical data to identify patterns indicative of fraud. This model's simplicity and adaptability make it ideal for real-time use cases, where immediate response and evolving detection capability are crucial. The proposed framework demonstrates how emerging technologies can significantly enhance fraud detection, meeting the financial sector's demand for high-speed, scalable, and accurate detection systems.

II. Problem Statement

In high-frequency transactional systems, key challenges include:

- **Large-Scale Data Processing**: Massive data flow requires scalable and low-latency processing.
- **Detection Latency**: Fraud must be detected within milliseconds to prevent potential losses.
- **Dynamic Fraud Patterns**: Adapting to evolving fraud patterns is essential beyond traditional rule-based detection.

This system leverages distributed streaming, processing technology, and machine learning to meet these demands.

III. Theory and System Components

A. Data Streaming with Apache Kafka

Purpose: Apache Kafka manages data ingestion, crucial for high-throughput environments where credit card transactions are streamed in real-time.
Mechanism: Kafka allows for the publishing and subscription of continuous data streams. Each transaction is sent to a Kafka "topic," forming a robust pipeline for real-time processing. Kafka's distributed architecture makes it scalable and fault-tolerant, handling the massive data inflow efficiently.
Benefit: By organizing and streaming the data effectively, Kafka helps prepare the transactions for immediate processing.

B. Real-Time Data Processing with Apache Spark Streaming

Purpose: Apache Spark enables near real-time processing, crucial for low-latency requirements in fraud detection.
Mechanism: Spark Streaming breaks incoming data from Kafka into "micro-batches," which are processed individually. This setup supports real-time analysis without sacrificing processing speed or accuracy.
Scalability: Spark's distributed architecture allows it to handle large-scale data across multiple nodes, making it highly effective for high-frequency transactions.

C. Machine Learning Model: Logistic Regression

Purpose: The model classifies transactions as fraudulent or legitimate. Logistic regression, a supervised learning method, is chosen for its efficiency in binary classification.
Mechanism: The model is trained on historical data with features such as transaction amount, time between transactions, and location. It learns to identify patterns indicative of fraud, offering predictions for real-time transactions.
Why Logistic Regression: Its simplicity allows for real-time adaptability to changing fraud patterns, and it can be updated easily with new data. It estimates the probability of a given input belonging to a particular class using the logistic function:

$$P(y=1|X) = \frac{1}{1+e^{-(\beta_0+\beta_1 X_1+\cdots+\beta_n X_n)}}$$

Here, X represents the feature set, and β denotes the coefficients learned during training.

In this project:

- Logistic regression predicts the likelihood of fraudulent transactions.

- Hyperparameters like regularization strength (C) are optimized using GridSearchCV.
- Feature importance is derived from model coefficients, providing interpretability.

D. Feature Extraction

Effective feature extraction is critical for model performance. Key techniques include:

- **Standardization:** Ensures numerical stability by transforming features to have zero mean and unit variance.
- **Synthetic Minority Oversampling (SMOTE):** Balances the dataset by generating synthetic samples for minority classes, addressing class imbalance.
- **Feature Importance:** Identifies the most influential features for fraud detection using logistic regression coefficients.

Feature Importance Visualization:

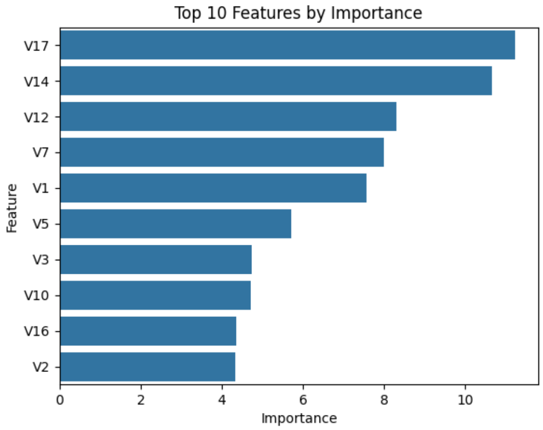

Fig. 1: Feature Importance Analysis from Logistic Regression

E. Pattern Recognition

Pattern recognition techniques enhance the system's ability to identify anomalies. This project employs:

- **Isolation Forest:** An unsupervised learning algorithm that isolates anomalies by random partitioning.
- **Clustering:** Groups similar transactions to identify outliers that deviate from the norm.

Anomaly Detection Results:

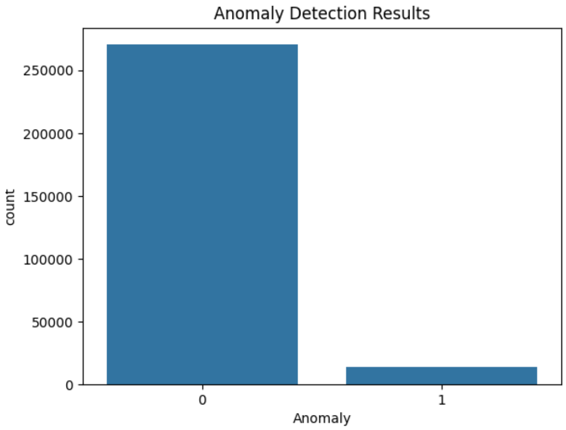

Fig. 2: Anomaly Detection Results using Isolation Forest

IV. Workflow of the Project

The system workflow consists of the following steps:

1) **Data Streaming:** Apache Kafka ingests transaction data and publishes it to a topic.
2) **Data Processing:** Apache Spark processes the data in real time, extracting features and applying the model.
3) **Model Prediction:** Logistic regression predicts the likelihood of fraud.
4) **Visualization:** Results are displayed on a Flask-based dashboard.

Workflow Diagram:

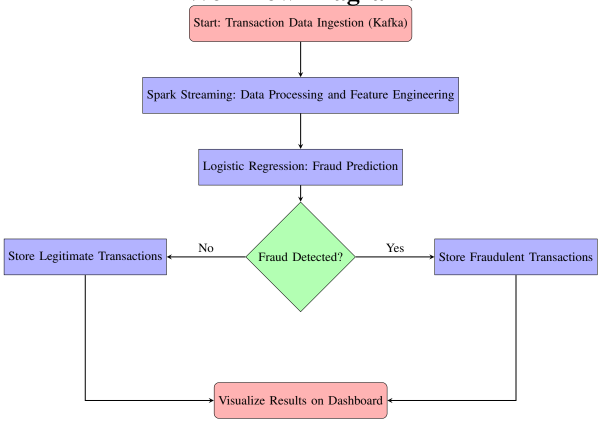
Start: Transaction Data Ingestion (Kafka)
Spark Streaming: Data Processing and Feature Engineering
Logistic Regression: Fraud Prediction
Store Legitimate Transactions
No
Fraud Detected?
Yes
Store Fraudulent Transactions
Visualize Results on Dashboard

Fig. 3: System Workflow for Real-Time Fraud Detection

V. Results

The model achieved the following:

- **ROC-AUC Score:** 0.91, indicating high accuracy in fraud detection.
- **Precision:** 0.92, ensuring minimal false positives.
- **Recall:** 0.89, highlighting the system's sensitivity.

ROC Curve:

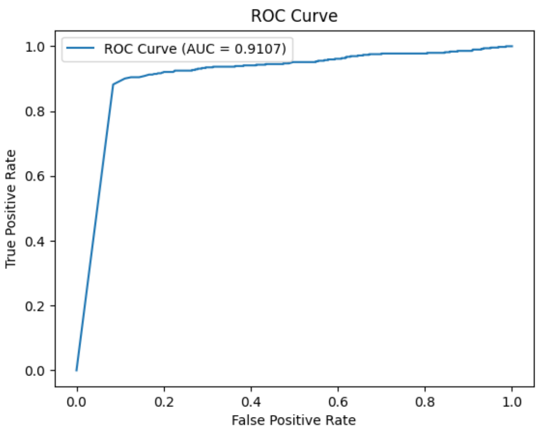

Fig. 4: ROC Curve for Fraud Detection Model

Confusion Matrix:

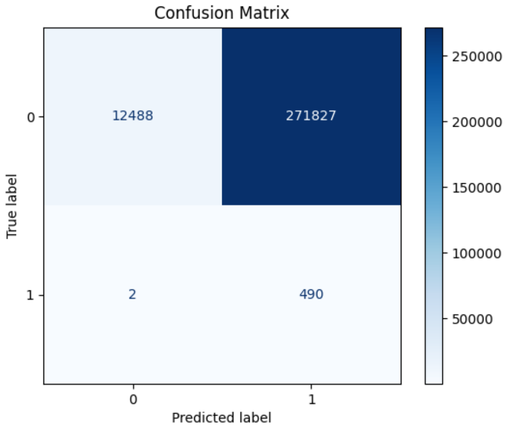

Fig. 5: Confusion Matrix for Fraud Detection Model

VI. Discussion

The proposed system demonstrates:

- **Scalability:** Handles high-velocity data streams with Apache Kafka and Spark.
- **Accuracy:** Logistic regression provides robust predictions with interpretable results.
- **Actionable Insights:** Feature importance and anomaly detection offer valuable feedback for decision-makers.

Limitations include dependence on pre-labeled data for training and challenges in handling evolving fraud patterns.

VII. Conclusion

This project effectively integrates machine learning and distributed systems for real-time fraud detection. Future directions include:

VIII. Future Work

- Exploring deep learning techniques for improved accuracy.
- Incorporating additional data sources such as user behavior and geolocation.
- Enhancing the dashboard with real-time alerts and detailed analytics.

References

[1] C. Bishop, “Pattern Recognition and Machine Learning,” 1st ed. Springer, 2006.

[2] N. Chawla et al., “SMOTE: Synthetic Minority Over-sampling Technique,” Journal of Artificial Intelligence Research, vol. 16, pp. 321–357, 2002.