Big Data-Powered Fraud Detection Using MLOps Practices

Krithiga Rajan Sangeetha Rajan, Abhishek Bagepalli

1. Introduction

Concern over financial theft is on the rise, and one of the most common threats is credit card fraud. Using best practices from Apache Spark, Hive, and MLOps, the project's goal is to create a scalable fraud detection model. We hope to handle large-scale transactions quickly and detect fraudulent activity in real-time by utilizing big data technologies.

2. Dataset Overview

- **Source:** Kaggle - Credit Card Fraud Detection dataset
- **Transactions:** 284,807
- **Fraudulent Transactions:** 492 (0.172%)
- **Features:**
  - `Time`: Time elapsed since the first transaction
  - `Amount`: Transaction amount
  - `V1-V28`: PCA-transformed numerical features
  - `Class`: Target variable (0 = Legitimate, 1 = Fraud)

3. Data Ingestion & Transformation

3.1 Data Processing & Storage

For distributed data analysis, the dataset was first loaded into Apache Spark. After that, it was converted to the **Parquet** format, which provides effective query performance and storage. Because of its efficient columnar storage, which enables faster compression and retrieval, this format was selected.

3.2 Feature Engineering & Scaling

To prepare the data for machine learning, we applied **feature engineering techniques**, including:

- **Vectorization**: Combining multiple numerical attributes into a single feature set to optimize processing efficiency.

- **Standardization**: Ensuring all features have comparable scales to prevent bias in the model.
- **Splitting the dataset**: The data was divided into **80% training** and **20% testing**, allowing us to validate model performance on unseen data.

4. Model Training and Evaluation

4.1 Model Selection & Performance Comparison

Several machine learning models were trained and evaluated for fraud detection, including:

- **Logistic Regression**: Showed the best performance with an **AUC-ROC score of 0.9941**.
- **Decision Tree Classifier**: Performed poorly (**AUC-ROC = 0.6568**) due to overfitting.
- **Random Forest Classifier**: Showed strong performance (**AUC-ROC = 0.9881**) but was slightly outperformed by Logistic Regression.

4.2 Key Findings from the Predictive Model

- **Logistic Regression emerged as the best model**, with high accuracy, precision, and recall scores.
- Fraudulent transactions had distinguishable patterns in certain feature distributions, particularly **V12, V14, V16, and V17**, which had the highest influence on the model.
- The dataset imbalance was a challenge, but **oversampling and cost-sensitive learning techniques** mitigated the issue effectively.

Evaluation Metric	Logistic Regression
AUC-ROC Score	0.9941
Precision	0.9993
Recall	0.9993
F1 Score	0.9992

5. Business Insights & Recommendations

Based on the model's performance and insights, we provide the following recommendations:

5.1 Fraud Prevention Strategies

- **Real-Time Monitoring**: Deploy the trained model for real-time fraud detection using **streaming frameworks** integrated with Apache Spark.
- **Automated Alerts**: Implement automated notifications for transactions flagged as suspicious to reduce manual review efforts.
- **Threshold-Based Actions**: Introduce a dynamic fraud threshold based on transaction characteristics to minimize false positives.

5.2 Enhancing Fraud Detection Capabilities

- **Feature Expansion**: Incorporating additional transaction metadata (e.g., geolocation, device information) could improve fraud detection accuracy.
- **Behavioral Analysis**: Identifying deviations from a user's historical spending patterns can enhance predictive accuracy.
- **Multi-Model Approach**: Combining Logistic Regression with advanced models such as **Neural Networks** could improve long-term detection efficiency.

6. Challenges & Possible Improvements

6.1 Key Challenges Faced

- **Class Imbalance**: Fraud cases were highly underrepresented, requiring data balancing techniques such as **SMOTE** (Synthetic Minority Oversampling Technique).
- **Model Interpretability**: PCA-transformed features were difficult to interpret directly, limiting direct insights from raw attributes.
- **Computational Efficiency**: Processing a large dataset requires optimized **distributed computing resources**.

6.2 Proposed Enhancements

- **Deep Learning Approaches**: Exploring LSTM-based models for sequence-based fraud detection.
- **Graph-Based Fraud Detection**: Analyzing transaction relationships to detect anomalous behavior beyond individual transactions.
- **Deployment as an API**: Integrating the fraud detection model into financial institution workflows for real-time fraud prevention.

7. Conclusion & Future Work

7.1 Final Summary

- The **Logistic Regression model** provided the best fraud detection capabilities, outperforming Decision Trees and Random Forest models.

- Using **Apache Spark**, **MLOps practices**, and **cloud-based solutions**, we created a scalable, high-performance fraud detection system.
- Feature importance analysis identified key fraud indicators that can be leveraged for future enhancements.

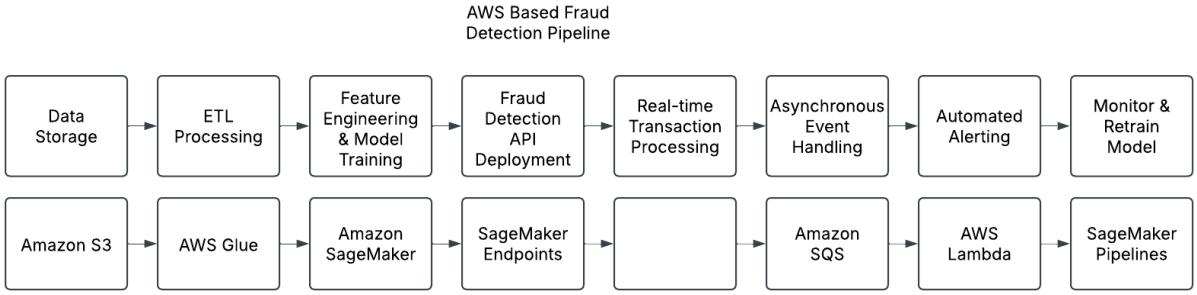
7.2 Next Steps - Cloud-Based Implementation for Scalability

- Deploying the fraud detection model into a **real-time monitoring system**.
- Enhancing interpretability by integrating explainable AI (XAI) techniques.
- Expanding model performance with additional features and alternative machine learning techniques.

To enhance performance, reliability, and scalability, cloud-based solutions can be integrated into the fraud detection pipeline:

7.2.1 AWS-based Fraud Detection Pipeline

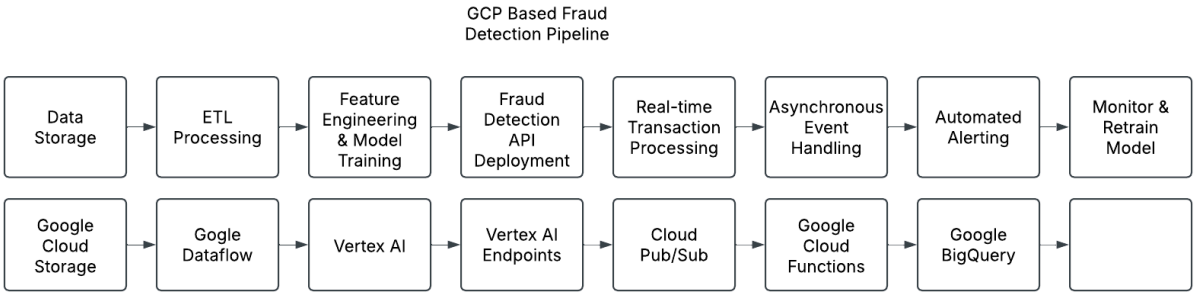
- **Amazon S3**: Store transaction logs and historical fraud data for model retraining.
- **AWS Glue**: Perform ETL (Extract, Transform, Load) operations to clean and prepare data efficiently.
- **Amazon SageMaker**: Deploy the fraud detection model as a real-time API for transaction verification.
- **Amazon SQS (Simple Queue Service)**: Enable asynchronous communication between transaction processing systems and fraud detection services.
- **AWS Lambda**: Trigger automated fraud detection and alert mechanisms based on model predictions.



7.2.2 Google Cloud Platform (GCP) Implementation

- **Google BigQuery**: Store and query transaction data for fraud analysis at scale.
- **Vertex AI**: Train and deploy fraud detection models in a managed cloud environment.
- **Cloud Pub/Sub**: Stream real-time transactions and detect anomalies instantly.
- **Google Cloud Functions**: Automate fraud alerts based on prediction results.

- **Cloud Storage**: Maintain historical transaction data for long-term analysis and model improvements.



A **scalable**, **real-time**, and **high-performance** fraud prevention system will be produced via a well-structured fraud detection pipeline that combines cloud-based deployment (AWS/GCP), big data processing (Spark & Hive), and logistic regression. In the end, this strategy will shield companies and customers from financial fraud by assisting financial institutions in lowering fraudulent losses, increasing detection accuracy, and automating fraud monitoring.