Final Project Report

Scalable Fraud Detection for E-Commerce.

1. Introduction

It is essential to use effective data processing and machine learning approaches to spot suspicious activity since online fraud is becoming more and more common. The goal of this project is to create a reliable and scalable fraud detection system by utilizing Apache Spark for data processing, Amazon Athena for data storage, and Spark's MLlib for machine learning.

2. Objectives

This project's goal is to provide a thorough pipeline for data processing and analysis that makes use of distributed computing technologies in order to identify fraudulent e-commerce transactions.

3. Data Source

An e-commerce transaction dataset with 1.47 million rows that covers a variety of topics related to online transactions, including as payment methods, product categories, customer information, and signs of fraudulent activity, was selected for this research. The dataset, which is stored in an Amazon S3 bucket, was chosen for meaningful analysis using machine learning models because of its size and importance to fraud detection.

4. Data Processing and Transformation

PySpark was used for data processing and transformation, taking use of Spark's distributed computing capabilities to effectively manage the sizable dataset. Important actions included:

- **Data Cleaning**: In order to minimize dimensionality, fields including transaction ID, IP address, shipping, and billing addresses were eliminated, and incomplete rows were dropped to manage missing information.
- **Data Type Conversion**: Transaction amounts, client age, and transaction hour were all converted to the proper data types in the columns.
- **Feature Engineering**: To gain a better understanding of consumer behavior, aggregated customer-level features were developed, including total transactions, average transaction amount, and total transaction amount.
- **Transformation for Machine Learning**: To further process the data, StringIndexer was used to turn categorical variables into numerical features. A VectorAssembler was then used to put all of the features together for model training.
- **Managing Skewness:** To improve model performance, log transformations and capping were used to address feature skewness.

Amazon Athena was used to store the cleaned and aggregated data, making it easier to search for and retrieve it for use in further machine learning processes. After that, the data was prepared for machine learning, which included managing skewness to improve model performance.

5. Machine Learning Model Development

For fraud detection, the machine learning component made use of Spark MLlib. Two models were created and evaluated: Logistic Regression and Random Fores t Classifier.

- **Logistic Regression**: Using cross-validation, a logistic regression model was trained and refined, yielding an accuracy of 94.99% and an AUC of 0.75. With coefficients revealing feature significance, the model successfully differentiated between fraudulent and non-fraudulent transactions.
- **Random Forest Classifier**: A Random Forest model was also trained, and it demonstrated efficacy in terms of precision, recall, and F1 score, with an accuracy of 95.49%. The model's AUC of 0.56 indicated that more fine-tuning would be necessary to improve its discriminatory power.

A thorough grasp of the models' performance was obtained by evaluating them using measures including accuracy, precision, recall, and confusion matrices.

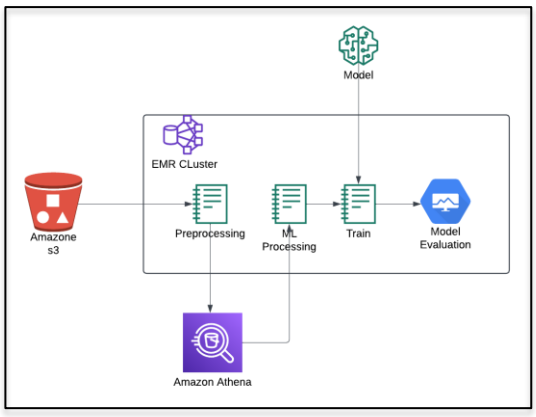
6. Evaluation and Results

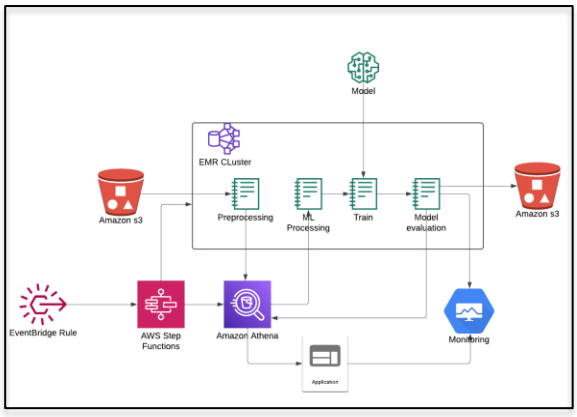
- **Model Evaluation**: The logistic regression model's limited AUC and good accuracy suggested that there may be space for improvement in terms of fraud case detection. Although the Random Forest classifier's accuracy was somewhat higher, its AUC limits were comparable.
- **Confusion Matrix**: In order to illustrate the difficulties in reducing false negatives in fraud detection, confusion matrices were created for both models, showing the quantity of true positives, true negatives, false positives, and false negatives.

7. Architecture Diagrams

- The core architecture for the fraud detection project includes Amazon S3, Amazon EMR, and Amazon Athena. Data is staged in Amazon S3, where it is initially ingested. Amazon EMR is used for preprocessing and transformation tasks using Apache Spark, which prepares the data for machine learning. Once processed, the data is utilized for model training using MLlib, followed by model evaluation. Amazon Athena is then used to store and query the transformed data for further analysis and insights
- The full solution expands on the core architecture by incorporating AWS Step Functions, Amazon EventBridge, and automation tools to ensure a scalable and repeatable pipeline. The main source of data storage is Amazon S3. To guarantee that every step is carried out in the right sequence, AWS Step Functions manage the data flow across the pipeline. To ensure that the pipeline is executed on time, Amazon EventBridge is used to start the workflow on a predetermined timetable. Amazon Athena makes querying easier, while

the EMR cluster handles data processing and training. The pipeline runs smoothly from data intake to model assessment thanks to performance monitoring provided by Amazon CloudWatch.





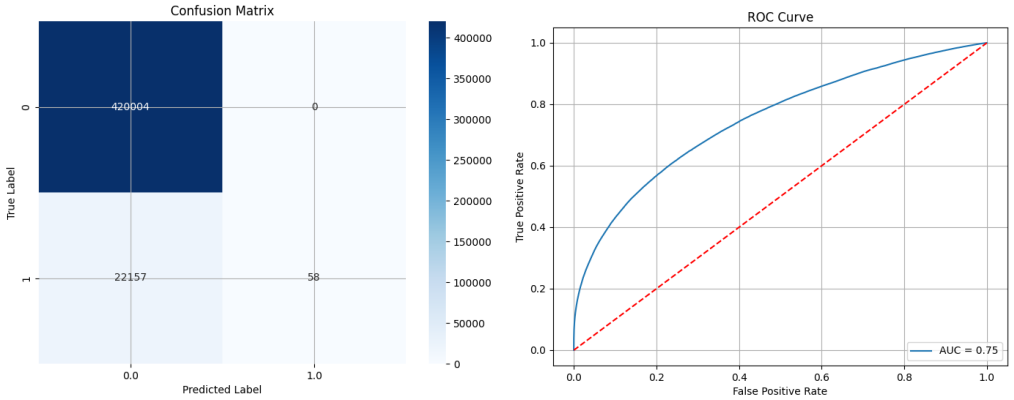
Core Architecture

Full Architecture

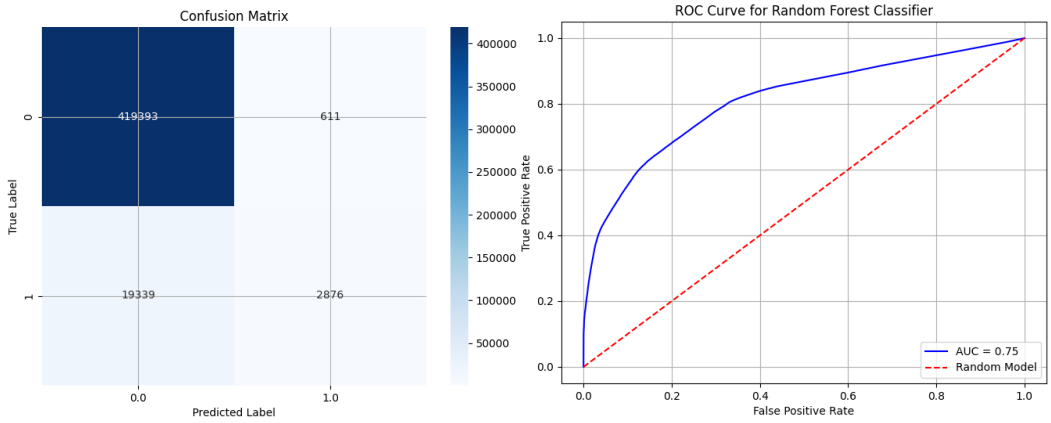
8. Visualizations and Insights

For both models, visualizations such as ROC curves and confusion matrices were made in order to comprehend performance. While ROC curves displayed the trade-off between true positive and false positive rates, identifying areas for improvement in detecting fraudulent transactions, heatmaps were utilized to demonstrate the distribution of projected vs. real labels.

- Logistic Regression: The model struggled to detect fraud situations, as seen by its high number of true negatives and low number of false negatives. An AUC of 0.75 indicates a modest level of performance.



- Random Forest: With greater accuracy and recall metrics, the model outperformed Logistic Regression in terms of true positive rate. The predictions' AUC, however, was 0.56, suggesting that class distinction was not very successful. With an AUC of 0.75, the ROC curve of the best model showed modest in train set performance with considerable potential for improvement.



9. Challenges Faced and Solutions

- Data Size and Distributed Processing: Spark was needed to handle a dataset of 1.47 million rows in order to do distributed data processing. Bottleneck-free operations were made possible by Spark's effectiveness in managing transformations and aggregations.
- During the project, I faced issues with the installation of the NumPy module while importing machine learning libraries. This was resolved by bootstrapping the cluster with NumPy installation script.
- It was challenging to find the ideal ratio of sensitivity to accuracy for fraud detection. Hyperparameter adjustment and cross-validation were employed to enhance model performance.

10. Conclusion and Future Work

Using Apache Spark for scalable processing, Amazon Athena for effective storage, and Spark MLlib for building models, this project successfully created an end-to-end data processing and machine learning pipeline for fraud detection. The random forest model outperformed the logistic regression model in terms of recall, but it struggled with class differentiation. ROC curves and confusion matrices were among the visualizations that highlighted the necessity of improving sensitivity and lowering false negatives.

Future enhancements will involve adding more features, such as past customer transactions, and investigating cutting-edge methods, such neural networks and gradient boosting machines, to increase accuracy. Improved hyperparameter tuning and the incorporation of real-time fraud detection capabilities might significantly improve the system's usefulness and efficacy.