BCSE206L

Foundation of Data Science

Digital Assignment 1

Jeeth M Shah

22BDS0183

Detecting and Preventing Money Laundering and Other Financial Crimes Using Big Data Analytics

1. Introduction

Money laundering is a significant financial crime where illicitly obtained money is processed to appear legitimate. Criminals exploit financial systems using techniques such as structuring transactions, utilizing offshore accounts, or creating shell companies to obscure the origins of funds. These activities pose serious risks to economic stability and national security.

To combat money laundering, financial institutions and regulatory bodies must leverage advanced analytical techniques. Traditional rule-based systems are no longer sufficient, as criminals continuously adapt to evade detection. Instead, **Big Data analytics** offers a powerful approach by analyzing vast amounts of **transaction data, network activity, and customer behavior** to identify suspicious patterns.

By applying the **Big Data Life Cycle**, financial institutions can efficiently **collect, preprocess, store, analyze, and visualize** transaction data. Machine learning models, artificial intelligence, and graph-based analytics help detect anomalies, unusual fund flows, and hidden connections between fraudulent entities.

This report explores how **Big Data-driven fraud detection systems** can enhance financial security. It also proposes a **Big Data Model and Architecture** that integrates **real-time monitoring, predictive analytics, and regulatory compliance mechanisms** to detect and prevent financial crimes effectively. Implementing such a system strengthens anti-money laundering efforts, ensuring transparency and trust in the global financial ecosystem.

---

2. Big Data Life Cycle Process

2.1 Data Acquisition

To effectively detect money laundering activities, vast amounts of data must be collected from various sources.

Sources of Data:

- **Bank transaction records:** Deposits, withdrawals, fund transfers, credit/debit card transactions, and wire transfers.
- **Customer KYC (Know Your Customer) data:** Identity verification details, employment information, and financial background.
- **Network activity logs:** Online banking usage, IP addresses, device locations, and login behaviors.
- **Regulatory Suspicious Activity Reports (SARs):** Reports filed by financial institutions regarding suspected fraudulent transactions.

- **News and social media feeds:** Public discussions and reports on financial crimes.

Data Collection Methods:

- **Real-time streaming:** APIs and event-driven data collection using Apache Kafka and Flume.
- **Batch processing:** Scheduled data extraction from financial databases.
- **Web scraping:** Collecting external news and social media data to detect emerging threats.

2.2 Data Preprocessing

Raw financial data is often messy and requires cleaning before analysis.

Challenges:

- **Inconsistent formats:** Transactions from different banks use different formats.
- **Missing or incomplete data:** Some customer details may be missing, requiring imputation.
- **Duplicate or redundant transactions:** Need to filter and deduplicate records.
- **Anomalies in transactions:** Irregular transaction patterns that may indicate fraud.

Techniques Used:

- **Data cleansing:** Removing null values and standardizing data formats.
- **Data normalization:** Converting values into a consistent scale.
- **Anomaly detection:** Using statistical methods and AI models to identify outliers.

2.3 Data Storage

Efficient storage solutions are needed to handle large volumes of financial data.

Storage Options:

- **Hadoop Distributed File System (HDFS):** For storing large-scale transaction logs.
- **NoSQL databases (MongoDB, Cassandra):** To manage unstructured customer behavior data.
- **Cloud storage (AWS S3, Google Cloud Storage):** For scalable and cost-efficient storage.

Data Security Considerations:

- **Encryption:** Protect sensitive customer data with AES encryption.
- **Access control:** Use role-based access to prevent unauthorized data access.
- **Data anonymization:** Mask personally identifiable information (PII).

2.4 Data Processing and Analysis

Detecting money laundering requires analyzing vast amounts of structured and unstructured data.

Processing Framework:

- **Apache Spark:** Distributed real-time data processing for fraud detection.
- **MapReduce:** Batch processing of historical transaction data.
- **Apache Flink:** Streaming analytics for monitoring transactions in real time.

Techniques Used:

- **Machine Learning Models:**
    - **Anomaly detection:** Identifies unusual transaction behavior.
    - **Decision trees and random forests:** Classify transactions as normal or suspicious.
    - **Neural networks:** Advanced AI for fraud detection.
- **Graph Analytics:**
    - Identifies suspicious relationships between accounts.
    - Detects hidden money-laundering networks.
- **Pattern Recognition:**
    - Analyzes repeated small transactions (structuring/smurfing).
    - Detects rapid fund movements across multiple accounts.

2.5 Data Visualization

Visualization helps financial analysts interpret data trends and detect fraud.

Tools Used:

- **Power BI/Tableau:** Dashboards for real-time monitoring.
- **Neo4j Graph Visualization:** To identify relationships between fraudulent transactions.
- **Heatmaps:** To detect high-risk geographic transaction zones.

Example Visualization:

- **Suspicious Transaction Trends Over Time:** A line graph showing abnormal spikes in transactions.
- **Money Flow Networks:** A graph showing illicit fund transfers between accounts.

2.6 Data Interpretation and Decision-Making

Financial institutions and regulatory agencies use processed data to take action.

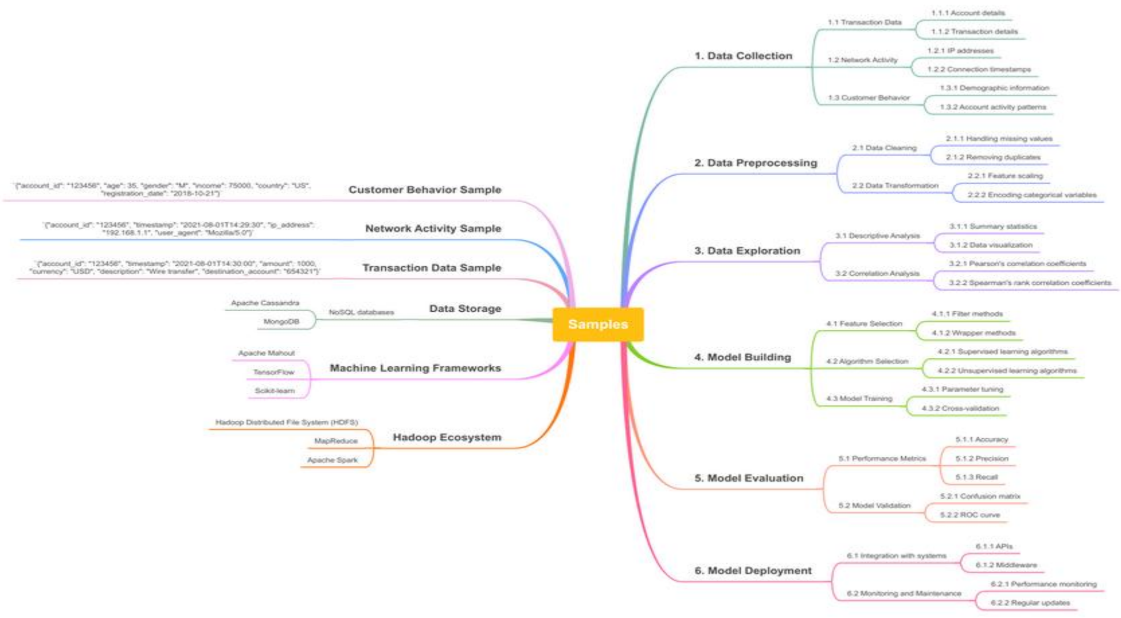
Outputs:

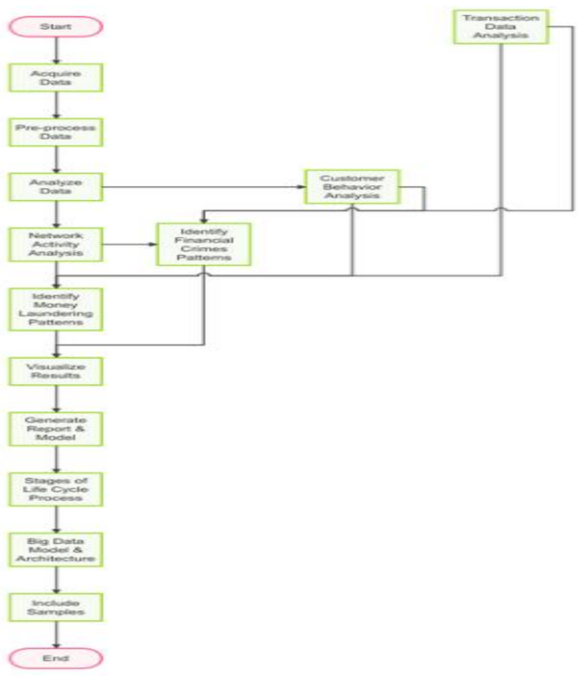
- **Risk scores:** Assign a risk score to customers based on transaction history.

- **Automated alerts:** Detect real-time suspicious transactions and notify compliance teams.
- **Regulatory reports:** Generate reports for government and law enforcement.

Actionable Decisions:

- **Flagging and freezing suspicious accounts.**
- **Reporting money laundering activities to regulatory authorities (FATF, FinCEN).**
- **Enhancing customer due diligence (CDD) for high-risk individuals.**


Samples
1. Data Collection
1.1 Transaction Data
1.1.1 Account details
1.1.2 Transaction details
1.2 Network Activity
1.2.1 IP addresses
1.2.2 Connection timestamps
1.3 Customer Behavior
1.3.1 Demographic information
1.3.2 Account activity patterns
2. Data Preprocessing
2.1 Data Cleaning
2.1.1 Handling missing values
2.1.2 Removing duplicates
2.2 Data Transformation
2.2.1 Feature scaling
2.2.2 Encoding categorical variables
3. Data Exploration
3.1 Descriptive Analysis
3.1.1 Summary statistics
3.1.2 Data visualization
3.2 Correlation Analysis
3.2.1 Pearson's correlation coefficients
3.2.2 Spearman's rank correlation coefficients
4. Model Building
4.1 Feature Selection
4.1.1 Filter methods
4.1.2 Wrapper methods
4.2 Algorithm Selection
4.2.1 Supervised learning algorithms
4.2.2 Unsupervised learning algorithms
4.3 Model Training
4.3.1 Parameter tuning
4.3.2 Cross-validation
5. Model Evaluation
5.1 Performance Metrics
5.1.1 Accuracy
5.1.2 Precision
5.1.3 Recall
5.2 Model Validation
5.2.1 Confusion matrix
5.2.2 ROC curve
6. Model Deployment
6.1 Integration with systems
6.1.1 APIs
6.1.2 Middleware
6.2 Monitoring and Maintenance
6.2.1 Performance monitoring
6.2.2 Regular updates
Customer Behavior Sample
'{"account_id": "123456", "age": 35, "gender": "M", "income": 75000, "country": "US", "registration_date": "2018-10-21"}'
Network Activity Sample
'{"account_id": "123456", "timestamp": "2021-08-01T14:29:30", "ip_address": "192.168.1.1", "user_agent": "Mozilla/5.0"}'
Transaction Data Sample
'{"account_id": "123456", "timestamp": "2021-08-01T14:30:00", "amount": 1000, "currency": "USD", "description": "Wire transfer", "destination_account": "654321"}'
Data Storage
NoSQL databases
Apache Cassandra
MongoDB
Machine Learning Frameworks
Apache Mahout
TensorFlow
Scikit-learn
Hadoop Ecosystem
Hadoop Distributed File System (HDFS)
MapReduce
Apache Spark


Start
Acquire Data
Pre-process Data
Analyze Data
Network Activity Analysis
Identify Money Laundering Patterns
Visualize Results
Generate Report & Model
Stages of Life Cycle Process
Big Data Model & Architecture
Include Samples
End
Transaction Data Analysis
Customer Behavior Analysis
Identify Financial Crimes Patterns

3. Proposed Big Data Model and Architecture

To effectively detect and prevent money laundering, a robust **Big Data Model and Architecture** must be implemented. This architecture should be capable of handling large volumes of **structured and unstructured data**, processing real-time and batch transactions, and integrating advanced **fraud detection models**. Below is a detailed breakdown of the architecture components that facilitate efficient money laundering detection.

3.1 Architecture Components

The proposed architecture consists of five essential layers: **Data Ingestion, Data Storage, Processing, Visualization, and Regulatory Compliance Integration**. Each layer plays a crucial role in handling, processing, and analyzing financial transaction data to identify suspicious activities.

---

1. Data Ingestion Layer

The **Data Ingestion Layer** is responsible for collecting data from various sources in real-time and batch formats. Given the high volume and velocity of financial transactions, this layer ensures seamless and efficient data collection.

Key Components:

- **Apache Kafka:**
  - A distributed event-streaming platform used to capture, process, and route real-time financial transactions.
  - Ensures high-throughput and fault-tolerant message queuing, critical for monitoring transaction flows.
  - Helps detect anomalies as transactions occur.
- **ETL (Extract, Transform, Load) Pipelines:**
  - Extracts transaction data from various financial databases, logs, and APIs.
  - Cleanses and transforms data to remove inconsistencies, missing values, and duplicate entries.
  - Loads structured data into **HDFS** and unstructured data into **NoSQL databases** for further analysis.
- **Data Sources:**
  - **Banking transactions** (deposits, withdrawals, transfers).
  - **KYC (Know Your Customer) data** (customer profiles, ID verification).
  - **Social media and external sources** for detecting emerging fraud patterns.

    - **Regulatory Suspicious Activity Reports (SARs)** filed by financial institutions.

This layer ensures that **all relevant data is captured, preprocessed, and stored efficiently**, making it ready for further analysis.

---

2. Data Storage Layer

The **Data Storage Layer** serves as the central repository for transactional and customer data. This layer ensures scalability, security, and efficient data retrieval for further processing and analysis.

Key Components:

- **Hadoop Distributed File System (HDFS):**
    - Stores large-scale structured financial records, transaction logs, and regulatory reports.
    - Optimized for batch processing of historical transaction data.
- **NoSQL Databases (MongoDB, Cassandra):**
    - Stores unstructured and semi-structured data, such as customer profiles, behavioral patterns, and network activity logs.
    - Provides flexibility for handling heterogeneous financial datasets.
- **Cloud Storage (AWS S3, Google Cloud Storage):**
    - Enables scalable and cost-efficient storage for long-term data retention.
    - Used for storing regulatory compliance reports and backups.

Data Security Considerations:

- **Data Encryption:** Ensures transaction data is encrypted at rest and in transit.
- **Access Control:** Role-based access control (RBAC) prevents unauthorized modifications.
- **Anonymization:** Personally Identifiable Information (PII) is masked to maintain customer privacy.

This layer ensures **high availability, security, and scalability** for efficient transaction monitoring and fraud detection.

---

3. Processing Layer

The **Processing Layer** is responsible for analyzing financial transactions, detecting anomalies, and identifying fraudulent activities. This layer employs **machine learning, graph analytics, and AI-driven fraud detection models**.

Key Components:

- **Apache Spark:**

    - A distributed computing framework for processing large volumes of financial data in real-time and batch mode.
    - Performs **parallel processing** to analyze transaction trends and suspicious patterns.
- **Machine Learning Models for Fraud Detection:**
    - **Anomaly Detection:** Identifies deviations from normal transaction behavior.
    - **Random Forest & Decision Trees:** Classifies transactions as normal or suspicious.
    - **Neural Networks & Deep Learning:** Detects complex money laundering schemes through pattern recognition.
    - **Natural Language Processing (NLP):** Analyzes unstructured data (such as emails and messages) to detect fraudulent communications.
- **Graph-Based Analytics (Neo4j, NetworkX):**
    - Helps visualize and detect hidden relationships between fraudulent entities.
    - Identifies money-laundering networks by analyzing fund transfers across multiple accounts.
- **Real-time Stream Processing (Apache Flink, Storm):**
    - Enables continuous monitoring of transactions to detect real-time suspicious activity.

This layer enhances the **detection capabilities** of financial institutions, helping them **identify fraud patterns and mitigate risks effectively**.

---

4. Visualization Layer

The **Visualization Layer** presents transaction insights in an intuitive and actionable format. It allows compliance teams and regulators to analyze fraud trends, customer behavior, and suspicious transaction patterns.

Key Components:

- **Power BI & Tableau:**
    - Provides **interactive dashboards** for real-time transaction monitoring.
    - Generates **risk scores** for customers based on historical behavior.
    - Displays **heatmaps** showing high-risk geographic zones for money laundering.
- **Neo4j Graph Analytics:**
    - Maps relationships between high-risk entities.
    - Detects **circular transactions** often used for layering funds.
- **Custom Reporting APIs:**

- Allows financial institutions to **generate automated reports** for law enforcement agencies.
- Provides regulatory compliance teams with **detailed suspicious activity logs**.

This layer **translates raw data into meaningful insights**, enabling fraud analysts to take **informed actions against suspicious activities**.

---

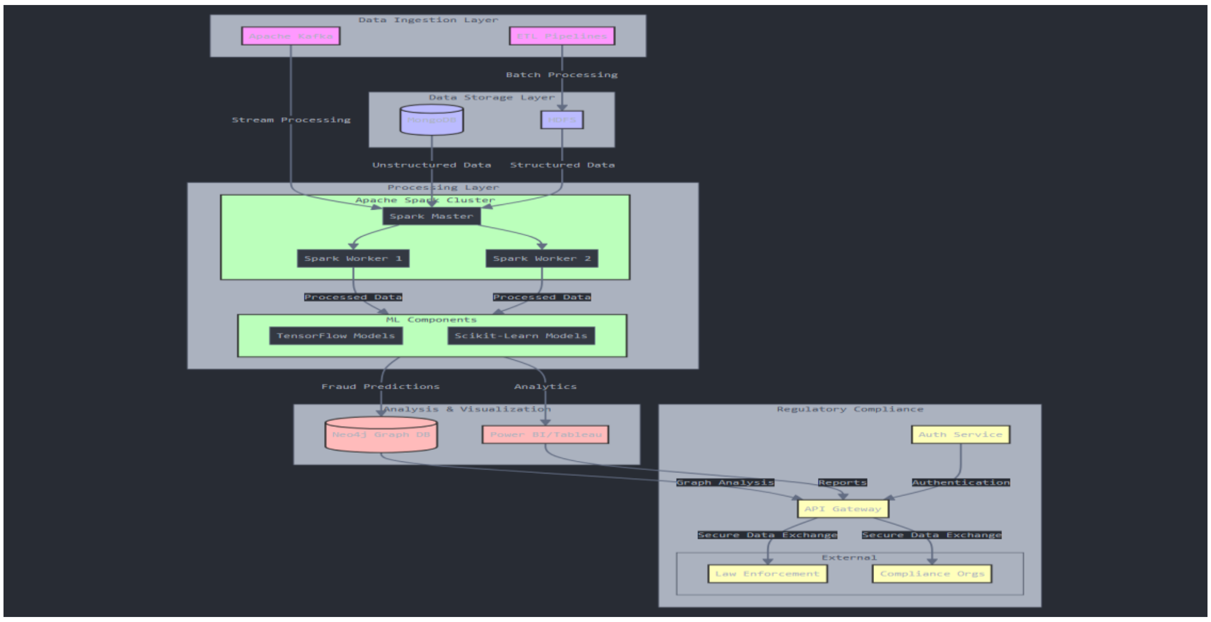
5. Regulatory Compliance Integration

The **Regulatory Compliance Integration Layer** ensures that financial institutions comply with anti-money laundering (AML) regulations and can share suspicious activity reports with regulatory bodies.

Key Components:

- **API-based Integration with Regulatory Authorities (FATF, FinCEN, RBI, SEC):**
  - Enables direct reporting of flagged transactions and high-risk entities.
  - Ensures compliance with global AML regulations and KYC requirements.
- **Automated Alerts & Case Management:**
  - Sends real-time alerts to compliance teams upon detecting suspicious activities.
  - Automates case documentation for law enforcement investigations.
- **Regulatory Reporting Dashboards:**
  - Provides government agencies with summarized AML reports.
  - Tracks compliance adherence and risk exposure.

This layer **strengthens transparency** between financial institutions and regulatory bodies, ensuring compliance with **global financial security laws**.



3.2 Sample Data

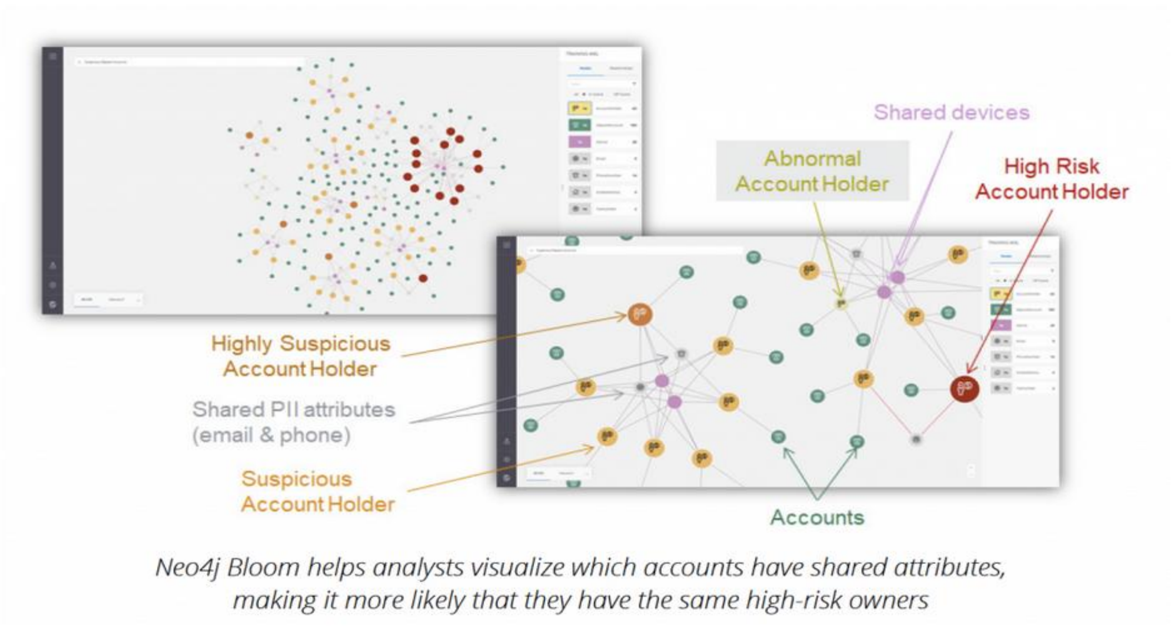
Transaction ID	Customer ID	Amount	Location	Time	Suspicious Score
TXN001	CUST123	5000	USA	10:30 AM	0.2
TXN002	CUST456	100000	UAE	2:15 PM	0.9
TXN003	CUST789	75000	India	5:45 PM	0.8
TXN004	CUST234	2500	UK	8:00 AM	0.1
TXN005	CUST567	200000	China	11:10 AM	0.95
TXN006	CUST890	15000	Germany	4:30 PM	0.4
TXN007	CUST345	120000	Russia	9:20 AM	0.85
TXN008	CUST678	9800	Canada	1:45 PM	0.3
TXN009	CUST901	500000	UAE	7:00 PM	0.99
TXN010	CUST112	60000	Brazil	6:15 AM	0.7

3.3 Sample Graph Analysis

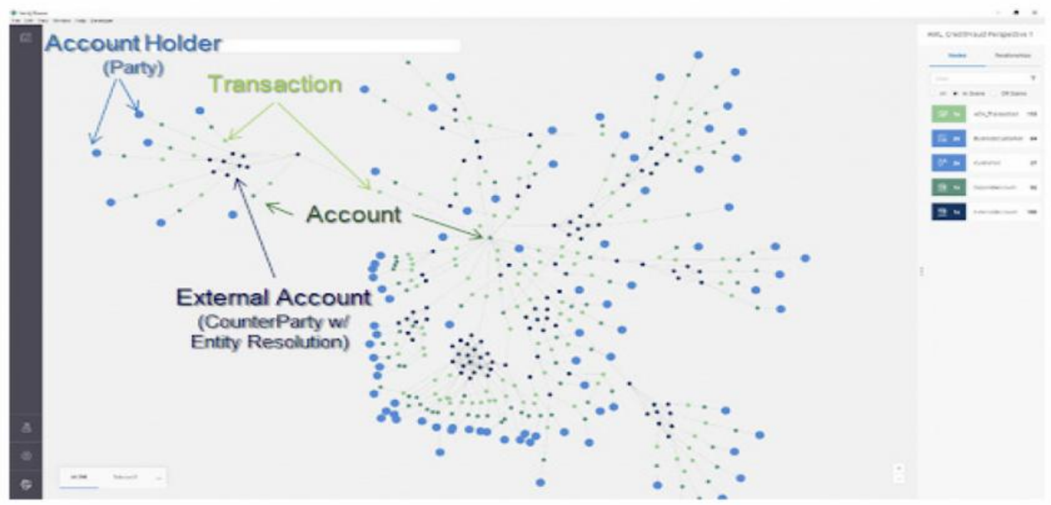
- **Nodes:** Represent individual customers/accounts.
- **Edges:** Represent transactions between accounts.
- **Suspicious nodes:** Highlight high-risk accounts in red.

Operations
Ops-Drill-down
Hub by Team
Productivity
Financials
HR
Alert Volume by Scenario & Product Type
Alert Volume by AML Scenario
Volume of Alerts by Resolution Status
Age of Alerts
Volume of Alerts
Trigger of Hits v/s Count of Alerts

The Neo4j BI Connector can be used to create AML dashboards such as this one from Tableau.


Shared devices
Abnormal Account Holder
High Risk Account Holder
Highly Suspicious Account Holder
Shared PII attributes (email & phone)
Suspicious Account Holder
Accounts

Neo4j Bloom helps analysts visualize which accounts have shared attributes, making it more likely that they have the same high-risk owners



Neo4j Bloom helps bankers and regulators detect and visualize payment chains and identify suspicious entities and events.

4. Conclusion

Money laundering detection using **Big Data analytics** is crucial for ensuring financial security and preventing illicit financial activities. Criminals use complex schemes, such as structuring transactions, using offshore accounts, and creating shell companies, to disguise illegally obtained funds. Traditional rule-based detection systems often fail to keep up with evolving fraud techniques, necessitating advanced analytical solutions.

A **Big Data-driven fraud detection system** integrates **real-time transaction monitoring, machine learning models, and graph analytics** to identify suspicious activities. **Apache Kafka** enables high-speed transaction ingestion, while **HDFS and NoSQL databases** store vast financial datasets. **Apache Spark** processes transaction data at scale, using **AI-driven anomaly detection models** to flag unusual fund movements. **Graph-based analytics (Neo4j)** uncovers hidden connections between fraudulent entities, exposing layered money-laundering networks.

Financial institutions can leverage **Power BI and Tableau dashboards** for visualizing transaction trends and compliance metrics. **Regulatory API integrations** ensure compliance with AML (Anti-Money Laundering) laws, allowing seamless reporting to authorities. Automated alerts and case management streamline investigations, improving response times.

This proactive model enhances **fraud detection, risk mitigation, and regulatory compliance**, strengthening global banking transparency and security. By adopting **Big Data analytics**, financial institutions can safeguard their networks against financial crimes while maintaining customer trust and regulatory integrity.

Foundations of Data Science

Digital Assignment 2

Jeeth M Shah

22BDS0183

Code Steps

Data Preprocessing

```
from sklearn.preprocessing import LabelEncoder, MinMaxScaler

# Handle missing values
df.fillna(df.median(), inplace=True)

# Encode categorical variables
label_encoder = LabelEncoder()
df["Transaction_Type_Encoded"] = label_encoder.fit_transform(df["Transaction_Type"])

# Normalize numerical features
scaler = MinMaxScaler()
df[["Transaction_Amount", "Account_Balance", "Customer_Risk_Score"]] = 
scaler.fit_transform(
    df[["Transaction_Amount", "Account_Balance", "Customer_Risk_Score"]]
)
```

EDA

```
import matplotlib.pyplot as plt
import seaborn as sns

# Fraud distribution
plt.figure(figsize=(6, 4))
```

```
sns.barplot(x=df["Fraudulent"].value_counts().index,
y=df["Fraudulent"].value_counts(normalize=True) * 100)

plt.xticks([0, 1], ["Legit", "Fraud"])

plt.title("Fraud vs Legit Transactions")

plt.show()


# Heatmap

plt.figure(figsize=(8, 6))

sns.heatmap(df.corr(), annot=True, cmap="coolwarm", fmt=".2f")

plt.title("Feature Correlation Heatmap")

plt.show()
```

Feature Engineering

```
# Transaction count per customer

df["Transaction_Count"] = df.groupby("Customer_ID")["Transaction_ID"].transform("count")


# Rolling transaction amount average

df["Rolling_Transaction_Avg"] =
df.groupby("Customer_ID")["Transaction_Amount"].transform(lambda x: x.rolling(5,
min_periods=1).mean())


# High-risk transaction flag

df["High_Risk_Transaction"] = (df["Customer_Risk_Score"] > 0.8).astype(int)
```

Machine Learning Model Training

```
from sklearn.model_selection import train_test_split

from sklearn.ensemble import RandomForestClassifier

from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score
```

```
# Define features and target

X = df.drop(columns=["Fraudulent", "Transaction_ID", "Customer_ID",
"Transaction_Timestamp"])

y = df["Fraudulent"]


# Train-test split

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


# Train Random Forest

model = RandomForestClassifier(n_estimators=100, random_state=42)

model.fit(X_train, y_train)


# Predictions & Evaluation

y_pred = model.predict(X_test)

print("Accuracy:", accuracy_score(y_test, y_pred))

print("Precision:", precision_score(y_test, y_pred))

print("Recall:", recall_score(y_test, y_pred))

print("F1 Score:", f1_score(y_test, y_pred))
```

Handling Class Imbalance (Manual Oversampling)

```
from sklearn.utils import resample


# Separate fraud and legit cases

df_fraud = df[df["Fraudulent"] == 1]

df_legit = df[df["Fraudulent"] == 0]


# Oversample fraud cases
```

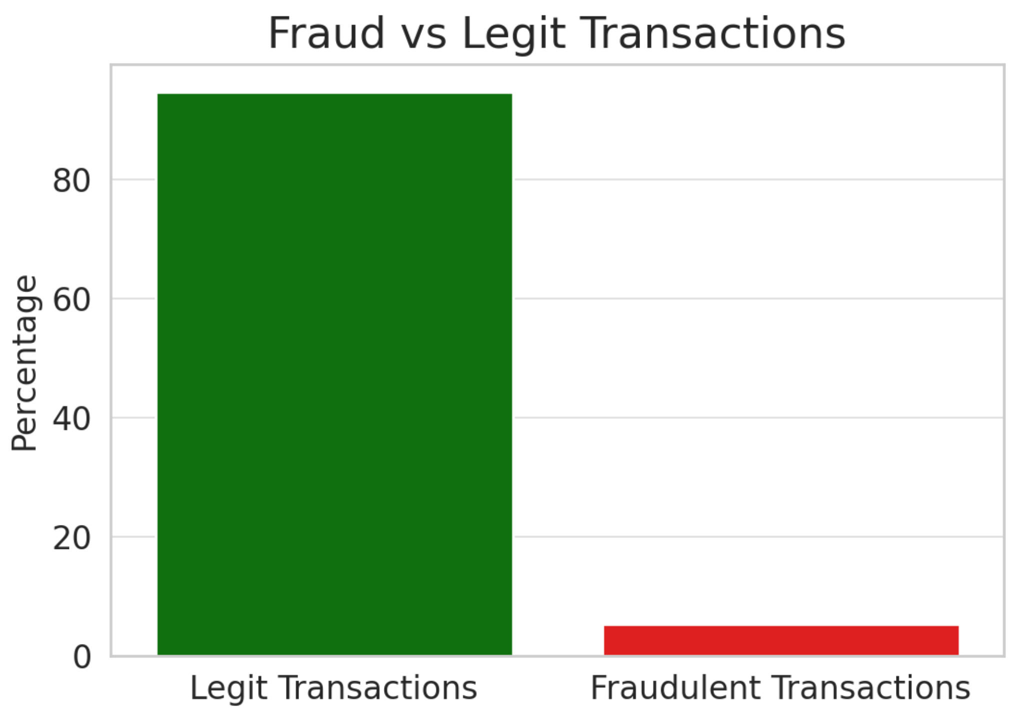
```
df_fraud_upsampled = resample(df_fraud, replace=True, n_samples=len(df_legit),
random_state=42)

# Combine both classes
df_balanced = pd.concat([df_legit, df_fraud_upsampled])

# Train model again with balanced data
X_balanced = df_balanced.drop(columns=["Fraudulent"])
y_balanced = df_balanced["Fraudulent"]

model.fit(X_balanced, y_balanced)
```

Visualization

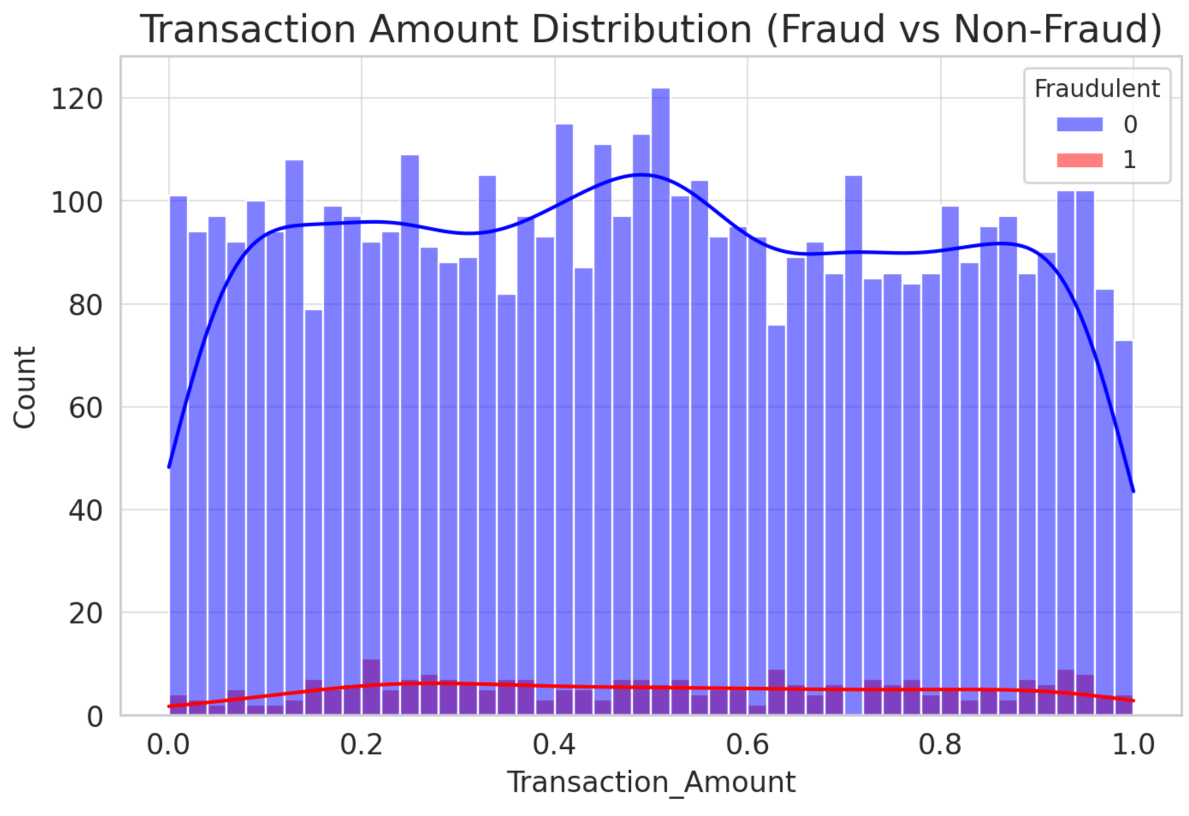
```
# Install & Load Required Packages
install.packages(c("ggplot2", "dplyr", "readr", "corrplot"), dependencies = TRUE)

library(ggplot2)
library(dplyr)
library(readr)
library(corrplot)

# Load the Dataset
df <- read_csv("C:/Users/jeeth/Downloads/feature_engineered_transactions.csv")

# Convert Fraudulent column to factor
df$Fraudulent <- as.factor(df$Fraudulent)

# View First Few Rows
print(head(df))
```

```r
# Fraud vs Legit Transactions Distribution
ggplot(df, aes(x = Fraudulent, fill = Fraudulent)) +
  geom_bar() +
  scale_fill_manual(values = c("green", "red")) +
  labs(title = "Fraud vs Legit Transactions", x = "Transaction Type", y = "Count") +
  theme_minimal()

# Transaction Amount Distribution (Fraud vs Non-Fraud)
ggplot(df, aes(x = Transaction_Amount, fill = Fraudulent)) +
  geom_density(alpha = 0.5) +
  scale_fill_manual(values = c("blue", "red")) +
  labs(title = "Transaction Amount Distribution", x = "Transaction Amount", y = "Density") +
  theme_minimal()

# Boxplot of Transaction Amount by Fraudulent Status
ggplot(df, aes(x = Fraudulent, y = Transaction_Amount, fill = Fraudulent)) +
  geom_boxplot() +
  scale_fill_manual(values = c("green", "red")) +
  labs(title = "Transaction Amount Distribution by Fraudulent Status", x = "Transaction Type", y =
"Transaction Amount") +
  theme_minimal()

# Correlation Heatmap
df_numeric <- df %>% select_if(is.numeric)  # Convert categorical variables to numeric
cor_matrix <- cor(df_numeric, use = "complete.obs")  # Compute correlation matrix

corrplot(cor_matrix, method = "color", type = "lower", tl.col = "black", tl.srt = 45)

# Number of Transactions per Customer
ggplot(df, aes(x = Transaction_Count)) +
  geom_histogram(fill = "blue", bins = 50, alpha = 0.7) +
  labs(title = "Number of Transactions per Customer", x = "Transaction Count", y = "Frequency") +
  theme_minimal()

# Rolling Transaction Average Distribution
ggplot(df, aes(x = Rolling_Transaction_Avg, fill = Fraudulent)) +
  geom_density(alpha = 0.5) +
  scale_fill_manual(values = c("blue", "red")) +
  labs(title = "Rolling Transaction Average Distribution", x = "Rolling Transaction Average", y = 
"Density") +
  theme_minimal()

# High-Risk Transactions (Customer Risk Score)
ggplot(df, aes(x = Customer_Risk_Score, fill = Fraudulent)) +
  geom_histogram(bins = 50, alpha = 0.7) +
  scale_fill_manual(values = c("green", "red")) +
  labs(title = "Customer Risk Score Distribution", x = "Customer Risk Score", y = "Count") +
  theme_minimal()
```

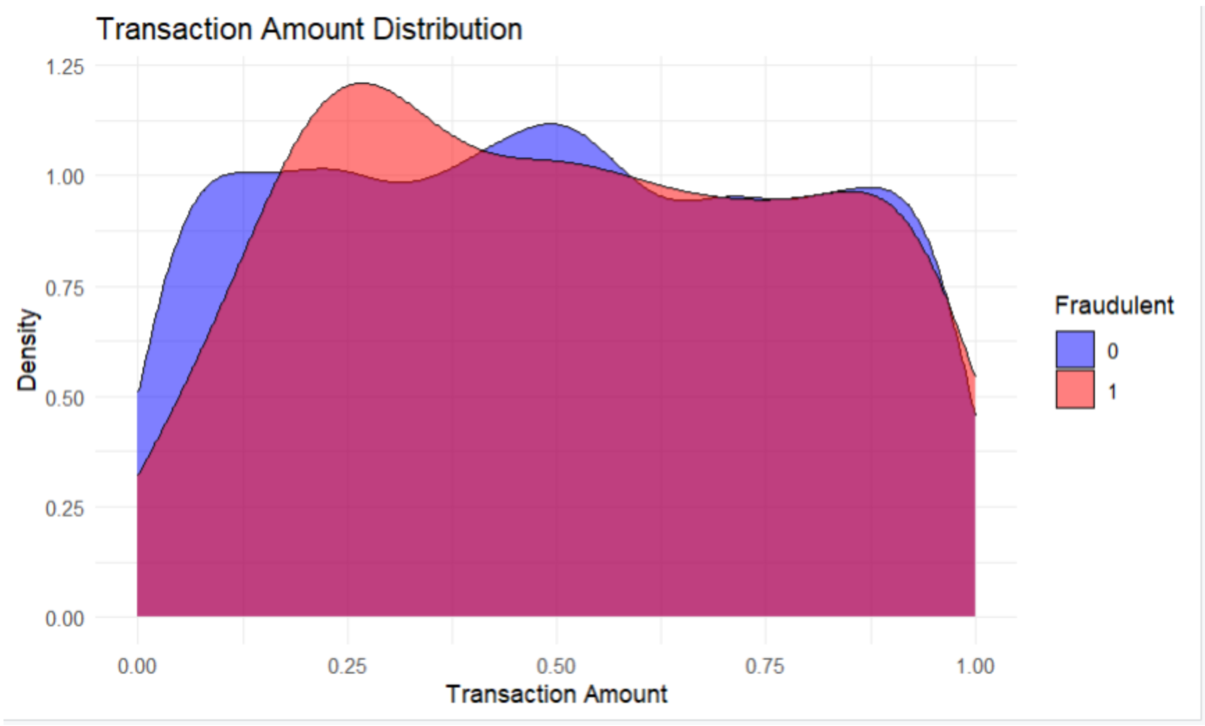
Visualization using Tableau

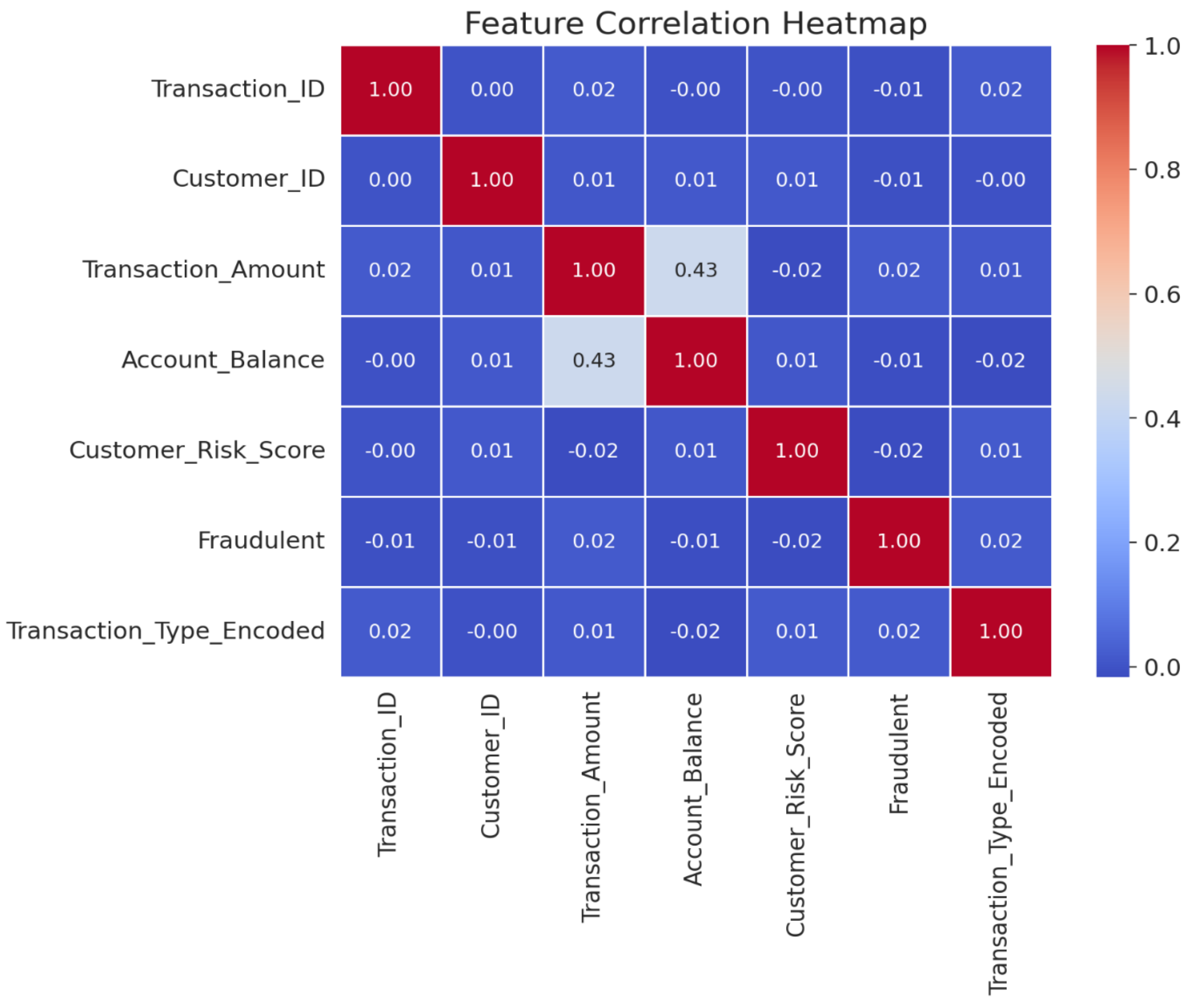


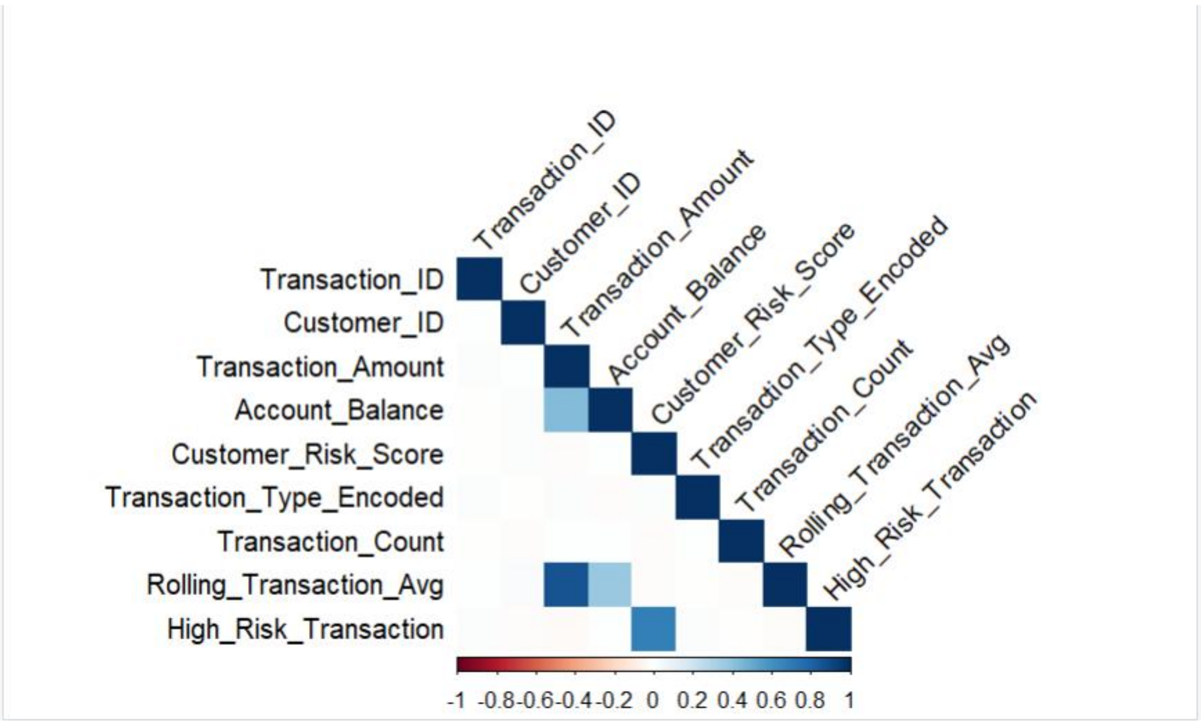
Fraud vs Legit Transactions
Percentage
0
20
40
60
80
Legit Transactions
Fraudulent Transactions

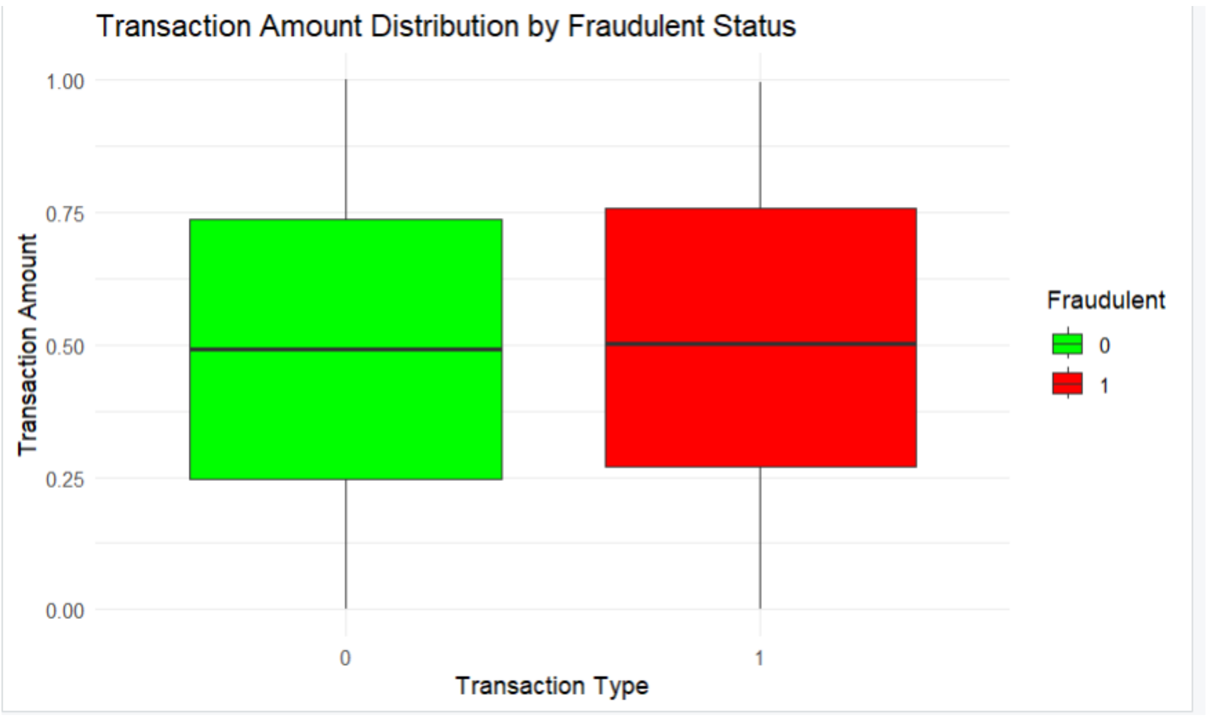


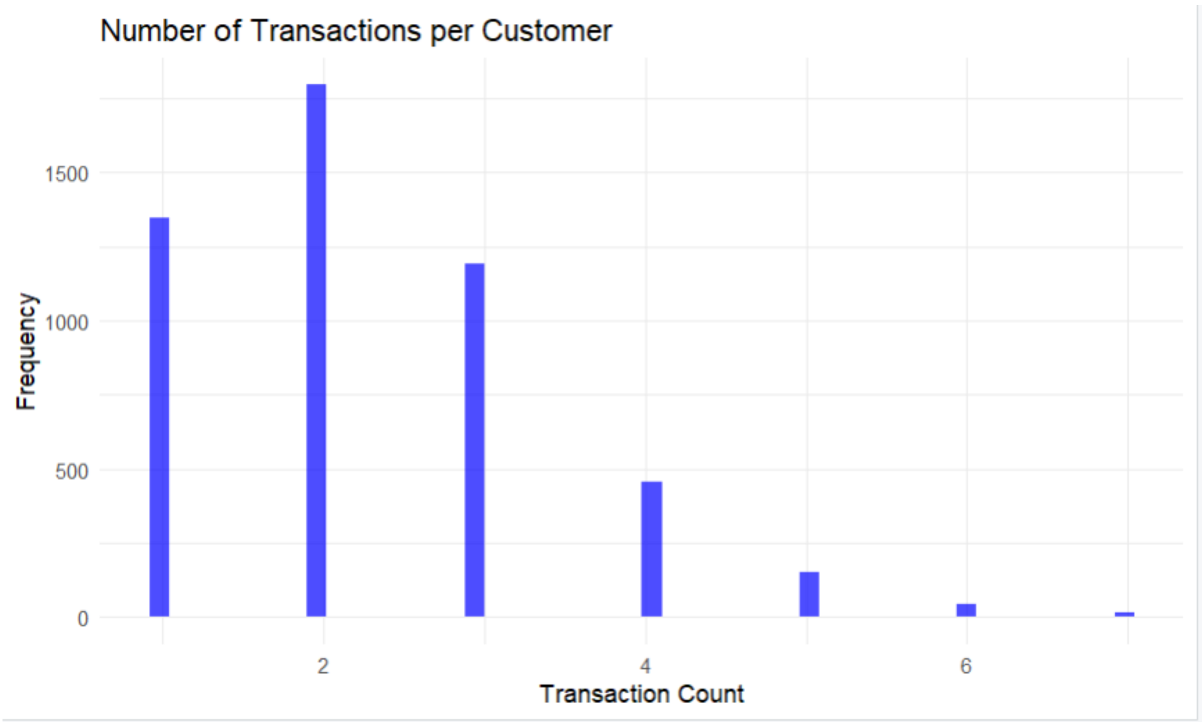
Transaction Amount Distribution (Fraud vs Non-Fraud)
Count
0
20
40
60
80
100
120
0.0
0.2
0.4
0.6
0.8
1.0
Transaction_Amount
Fraudulent
0
1

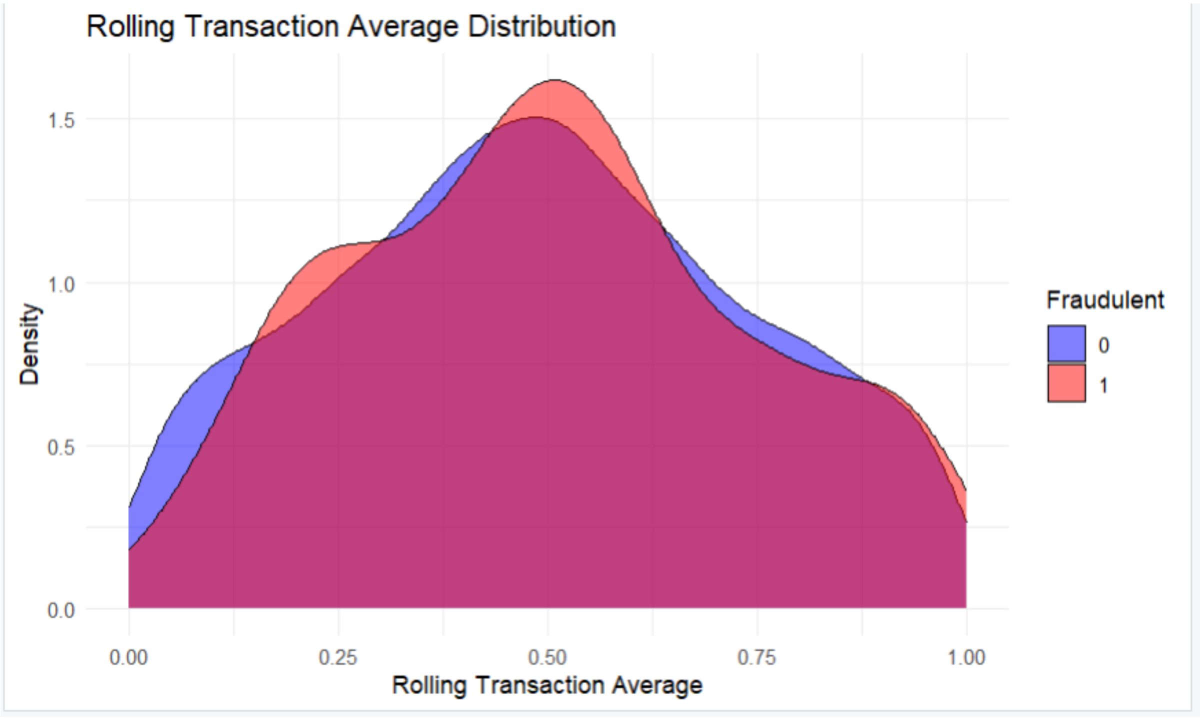

Transaction Amount Distribution
Density
1.25
1.00
0.75
0.50
0.25
0.00
0.00
0.25
0.50
0.75
1.00
Transaction Amount
Fraudulent
0
1

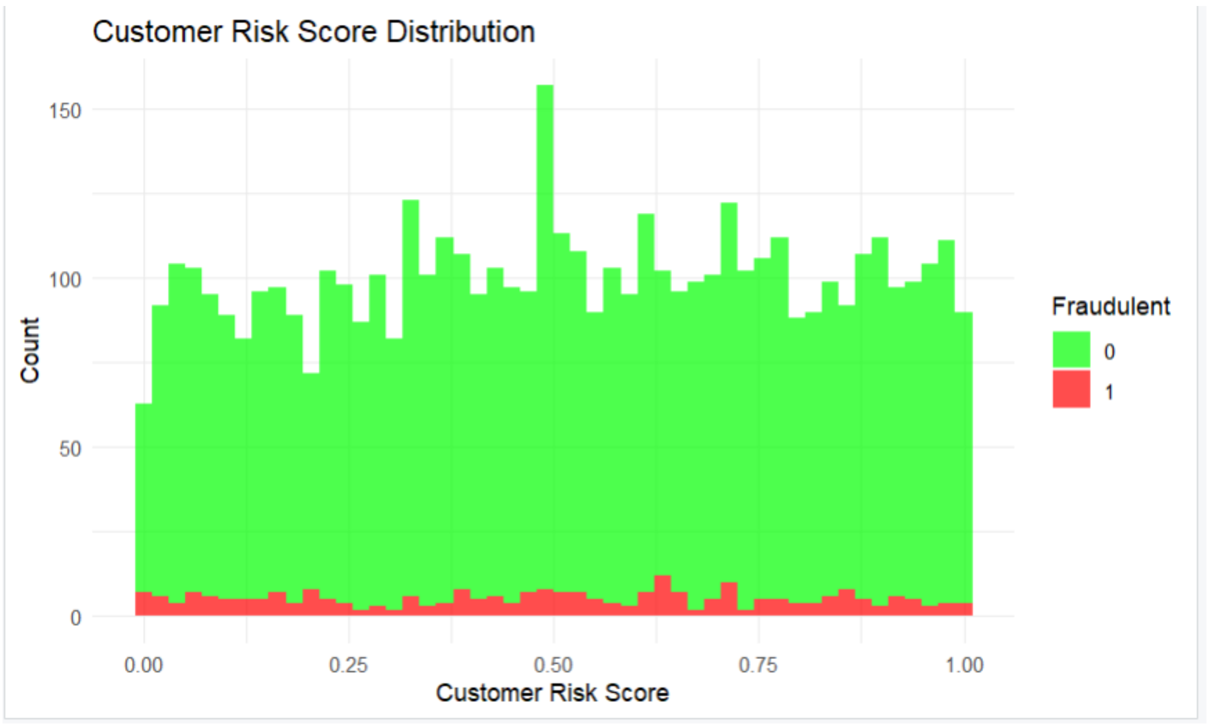

Feature Correlation Heatmap
Transaction_ID 1.00 0.00 0.02 -0.00 -0.00 -0.01 0.02
Customer_ID 0.00 1.00 0.01 0.01 0.01 -0.01 -0.00
Transaction_Amount 0.02 0.01 1.00 0.43 -0.02 0.02 0.01
Account_Balance -0.00 0.01 0.43 1.00 0.01 -0.01 -0.02
Customer_Risk_Score -0.00 0.01 -0.02 0.01 1.00 -0.02 0.01
Fraudulent -0.01 -0.01 0.02 -0.01 -0.02 1.00 0.02
Transaction_Type_Encoded 0.02 -0.00 0.01 -0.02 0.01 0.02 1.00
Transaction_ID
Customer_ID
Transaction_Amount
Account_Balance
Customer_Risk_Score
Fraudulent
Transaction_Type_Encoded
1.0
0.8
0.6
0.4
0.2
0.0


Transaction_ID
Customer_ID
Transaction_Amount
Account_Balance
Customer_Risk_Score
Transaction_Type_Encoded
Transaction_Count
Rolling_Transaction_Avg
High_Risk_Transaction
-1 -0.8 -0.6 -0.4 -0.2 0 0.2 0.4 0.6 0.8 1


Transaction Amount Distribution by Fraudulent Status
1.00
0.75
0.50
0.25
0.00
Transaction Amount
0
1
Transaction Type
Fraudulent
0
1


Number of Transactions per Customer
Frequency
0
500
1000
1500
2
4
6
Transaction Count


Rolling Transaction Average Distribution
Density
0.0
0.5
1.0
1.5
0.00
0.25
0.50
0.75
1.00
Rolling Transaction Average
Fraudulent
0
1



CSV Files

synthetic_transactions.csv

processed_transactions.csv

feature_engineered_transactions.csv

https://drive.google.com/file/d/1JL15Sv41KIkmFaP01BWDgS0UfC84E4Lx/view?usp=drive_link

https://drive.google.com/file/d/1VO48nmMOL3IteMwE2PMUZpV83xX7Hw6u/view?usp=drive_link

https://drive.google.com/file/d/1FxghMiCXd0qXtMLeaPQWLH8vW9toY4in/view?usp=drive_link