PipeLine

שם: שאדי שמשום
תז: 214511172

System stage:

```
File Name: PS_20174392719_1491204439457_log.csv
File Size: 470.67 MB
File Type: .csv
```

Now for the web information we will note the data source is from Kaggle, link:
https://www.kaggle.com/datasets/ealaxi/paysim1/data

The license is : CC BY-SA 4.0

Tags are: Finance and Crime

Time: written and lastly updated 8 years ago

Version control: we will work with git and IntelliJ

Now lets provide a brief explanation about the dataset:

The dataset used in this project is a synthetic financial transactions dataset generated by the PaySim simulator.
Due to the private nature of real financial transactions, there is very limited public access to real-world datasets for fraud detection research. PaySim solves this problem by creating realistic simulated mobile money transactions based on real aggregated logs from an African mobile money service.

The data includes five transaction types CASH-IN, CASH-OUT, DEBIT, PAYMENT, and TRANSFER and covers 744 time steps

(representing 30 days, 1 hour per step). It contains millions of records and simulates both normal and fraudulent activities to evaluate fraud detection methods.

Key points:

- Fields: Step, Type, Amount, Sender, Receiver, Balances before & after, Fraud labels.
- Important note: Balances should not be used for fraud detection as they are affected by fraud cancellation.
- Source: The simulation is based on real-world mobile money logs and is scaled to about 1/4 of the original dataset.
- Purpose: To help researchers test fraud detection algorithms with realistic but safe synthetic data.

Reference:
Lopez-Rojas et al., *"PaySim: A financial mobile money simulator for fraud detection,"* EMSS, 2016.

Data Governance:

Who created it?

- Researchers: E. A. Lopez-Rojas, A. Elmir, and S. Axelsson
- Part of a research team in Sweden, cited in the paper: "PaySim: A financial mobile money simulator for fraud detection" (EMSS, 2016).

When?

- Created during research published in 2016.

Where?

Developed at Blekinge Institute of Technology, Sweden.

Original financial logs from a multinational mobile money provider in an African country, simulated for public research use.

Why?

To provide an accessible dataset for testing fraud detection algorithms.

Real data is private → so they simulated realistic transactions with fraud to support safe, open research.

Meta data:

First we will show the output we got from the code and then explain the results:

```
--- META DATA STAGE ---
Data Shape (rows, columns): (6362620, 11)

Data Types:
step                int64
type               object
amount            float64
nameOrig           object
oldbalanceOrg     float64
newbalanceOrig    float64
nameDest           object
oldbalanceDest    float64
newbalanceDest    float64
isFraud             int64
isFlaggedFraud      int64
dtype: object
```

```
Missing Values Per Column:
step              0
type              0
amount            0
nameOrig          0
oldbalanceOrg     0
newbalanceOrig    0
nameDest          0
oldbalanceDest    0
newbalanceDest    0
isFraud           0
isFlaggedFraud    0
dtype: int64

Special Values Check:
Rows with negative amount: 0
Rows with negative balances: 0
```

Data Size

Shape: The dataset contains 6,362,620 rows and 11 columns.

Data Types

The columns include:

Numeric: step (int), amount (float), oldbalanceOrg (float), newbalanceOrig (float), oldbalanceDest (float), newbalanceDest (float), isFraud (int), isFlaggedFraud (int).

Categorical: type, nameOrig, nameDest.

This shows a mix of transaction details, amounts, balances, IDs, and fraud labels.

Missing Data

No missing values detected in any column. All rows have complete information.

Special Values

No negative values found for amount or balance columns. All values are logically consistent for financial transactions.

This confirms the dataset's integrity for further analysis, that means no data cleaning needed for nulls or negative outliers at this stage, so we can simply continuo the next one.

Now i will look at every column name and explain if it explain its data well or it is suspicious:

Column	**Meaning**	**Does it explain well what it contains?**
step	Time step (1 hour increments)	Good
type	Transaction type (PAYMENT, TRANSFER, etc.)	Clear
amount	Transaction amount	Clear
nameOrig	Origin account ID	clear

Column	Meaning	Does it explain well what it contains?
oldbalanceOrg	Sender’s balance before transaction	Clear
newbalanceOrig	Sender’s balance after transaction	Clear
nameDest	Destination account ID	nameDest is fine
oldbalanceDest	Receiver’s balance before transaction	Clear
newbalanceDest	Receiver’s balance after transaction	Clear
isFraud	True fraud label (1 = fraud)	Clear
isFlaggedFraud	Was flagged as suspicious (1 = flagged)	Clear

Data Statistics:

First we will show the statistics we got:

```
--- DATA STATISTICS STAGE ---

=== Central Tendencies ===
            amount  oldbalanceOrg  ...  oldbalanceDest  newbalanceDest
count  6.362620e+06   6.362620e+06  ...    6.362620e+06    6.362620e+06
mean   1.798619e+05   8.338831e+05  ...    1.100702e+06    1.224996e+06
std    6.038582e+05   2.888243e+06  ...    3.399180e+06    3.674129e+06
min    0.000000e+00   0.000000e+00  ...    0.000000e+00    0.000000e+00
25%    1.338957e+04   0.000000e+00  ...    0.000000e+00    0.000000e+00
50%    7.487194e+04   1.420800e+04  ...    1.327057e+05    2.146614e+05
75%    2.087215e+05   1.073152e+05  ...    9.430367e+05    1.111909e+06
max    9.244552e+07   5.958504e+07  ...    3.560159e+08    3.561793e+08

[8 rows x 5 columns]

Mode of transaction type:
0    CASH_OUT
Name: type, dtype: object
```

```
Mode of transaction type:
0    CASH_OUT
Name: type, dtype: object

=== Correlation Matrix ===
                    step    amount  ...   isFraud  isFlaggedFraud
step            1.000000  0.022373  ...  0.031578        0.003277
amount          0.022373  1.000000  ...  0.076688        0.012295
oldbalanceOrg  -0.010058 -0.002762  ...  0.010154        0.003835
newbalanceOrig -0.010299 -0.007861  ... -0.008148        0.003776
oldbalanceDest  0.027665  0.294137  ... -0.005885       -0.000513
newbalanceDest  0.025888  0.459304  ...  0.000535       -0.000529
isFraud         0.031578  0.076688  ...  1.000000        0.044109
isFlaggedFraud  0.003277  0.012295  ...  0.044109        1.000000

[8 rows x 8 columns]
```

```
[8 rows x 8 columns]

=== Value Counts for Transaction Type ===
type
CASH_OUT    2237500
PAYMENT     2151495
CASH_IN     1399284
TRANSFER     532909
DEBIT         41432
Name: count, dtype: int64

Fraudulent vs Non-Fraudulent Transactions:
isFraud
0    6354407
1       8213
Name: count, dtype: int64
```

```
=== Re-check for Missing ===
step              0
type              0
amount            0
nameOrig          0
oldbalanceOrg     0
newbalanceOrig    0
nameDest          0
oldbalanceDest    0
newbalanceDest    0
isFraud           0
isFlaggedFraud    0
dtype: int64

Special: Zero amount transactions:
Rows with zero amount: 16

=== Duplicate Rows ===
Number of duplicate rows: 0

Unique transaction types:
['PAYMENT' 'TRANSFER' 'CASH_OUT' 'DEBIT' 'CASH_IN']

=== Numeric Columns for Possible Dimension Reduction ===
['step', 'amount', 'oldbalanceOrg', 'newbalanceOrig', 'oldbalanceDest', 'newbalanceDest', 'isFraud', 'isFlaggedFraud']
```

Now we will explain the meaning of the results:

Central Tendencies

The dataset has wide transaction values:

- Mean amount: ~1,798 currency units.
- Max amount: ~92 million units.
- Balances: Large ranges from 0 to hundreds of millions.

The most common transaction type (mode) is CASH_OUT.

Correlation & Association

The correlation matrix shows weak correlations overall (i used Pearson correlation):

- amount has a slight positive correlation (~0.07) with isFraud.
- step (time) has almost no correlation with fraud.
- Balances show low correlations too.

This suggests that amount may be a useful feature for fraud detection.

Data Distribution

Transaction types:

- CASH_OUT: 2.2 million
- PAYMENT: 2.1 million
- CASH_IN: 1.3 million
- TRANSFER: ~530,000
- DEBIT: ~41,000

Fraud labels:

Fraudulent transactions: 8,213

Non-fraudulent: ~6.35 million

Highly imbalanced dataset.

Missing Values & Special Values

No missing values found.

No negative values.

16 transactions have zero amounts, which may need further inspection.

Duplicates & Unique Values

No duplicate rows.

5 unique transaction types confirmed.

Dimension Reduction

The numeric columns suitable for dimensionality reduction (e.g., PCA) are:

step, amount, oldbalanceOrg, newbalanceOrig, oldbalanceDest, newbalanceDest, isFraud, isFlaggedFraud.

The dataset is numerically consistent with no missing or duplicate rows, but is imbalanced (few frauds). Amount and transaction type are likely key predictors for fraud.

Abnormality detection:

First we will show the results we got:

```
--- ABNORMALITY DETECTION STAGE ---
Amount Median: 74871.94
MAD for amount: 68393.655
Number of amount outliers (MAD method): 1042843
Number of multi-feature outliers (Elliptic Envelope): 100
```

To get this results we used methods we learned in the class such as:

Single Feature (MAD):

Outliers in transaction amount were detected using the Median Absolute Deviation (MAD) method, which is robust to skewed data.

Multi Feature (Elliptic Envelope):

Multivariate outliers were detected using an Elliptic Envelope fitted on amount and oldbalanceOrg. This identifies transactions that are unusual when considering multiple features together.

Now lets explain the results:

Single Feature (MAD):

We used the Median Absolute Deviation (MAD) to detect outliers in the amount column.

Transactions with amounts more than 3×MAD away from the median (~74,871) were flagged.

Result: ~1 M transactions were identified as possible single-feature outliers, showing the dataset has many large transactions that may require deeper investigation.

Multi Feature (Elliptic Envelope):

We used Elliptic Envelope to detect multivariate outliers based on amount and oldbalanceOrg.

This method fits an elliptical shape around the normal data distribution and flags points outside as outliers.

On a sample of 10,000 rows, 100 transactions were flagged as unusual multi-feature outliers.

Why:
Combining single-feature and multi-feature abnormality detection helps spot suspicious transactions that might not be visible when looking at only one variable.
This is critical for fraud detection, where subtle patterns matter.

Clustering:

First i will show the results i got:

```
--- CLUSTERING STAGE ---
Cluster centers:
[[  175930.00186279   152668.78795484]
 [  126770.88240506 20528360.72291139]
 [  174806.51142857  6988185.7964127 ]]

Cluster sizes:
cluster
0    4606
2     315
1      79
Name: count, dtype: int64
Number of points far from cluster centers (possible cluster outliers): 50
```

Now i will explain them:

Grouping:
We used K-Means Clustering to group transactions based on amount and oldbalanceOrg. The model found 3 natural clusters:

- Cluster 0 (majority): ~92% of the sample, mostly normal transactions.
- Cluster 2: A mid-sized group with larger amounts or unusual balances.
- Cluster 1: A very small cluster of only 79 transactions — likely very distinct outliers or special patterns.

Meaning:
These clusters show how transactions naturally group by similar size and account balance patterns, revealing typical vs. unusual transaction behavior.

Points outside clusters:
I calculated the distance of each transaction to its cluster center and flagged points farthest away.
Result: 50 transactions (~1% of the sample) were found to be unusually far from any cluster center , potential internal outliers worth investigating.

This stage shows how clustering helps uncover hidden structures in the data and points out where unusual behavior might occur.

Segment analysis:

Now lets explain what we did here:

```
--- SEGMENT ANALYSIS STAGE ---

Average features per cluster:
                amount  oldbalanceOrg
cluster
0        175930.001863   1.526688e+05
1        126770.882405   2.052836e+07
2        174806.511429   6.988186e+06

Cluster size over time (sample):
   cluster  step  count
0        0     1      1
1        0     2      1
2        0     4      1
3        0     5      1
4        0     6      1
```

Features:

Shows mean amount and oldbalanceOrg for each cluster

Temporal:

Checks how cluster counts vary by step

Domain knowledge:

For example:

Cluster 1 shows very high amounts, likely large business transactions. Cluster 2 peaks at night possible risky pattern.”

Now here is the explanation of the results:

Features:
Using cluster averages, we see:

Cluster 0: Normal daily transactions with moderate amounts and low sender balances.

Cluster 1: Medium transaction amounts but very high sender balances , possibly VIP or large corporate accounts.

Cluster 2: Larger payments and higher balances , likely high-value payments or business transfers.

Temporal:
Basic checks show clusters appear at different time steps. A deeper time plot could reveal patterns like frauds happening at unusual hours.

Domain Knowledge:
These segments help explain who is transacting: daily users, high-value accounts, or possible high-risk batches.
This supports targeted monitoring, policy design, and fraud prevention so we can catch anomalies.

NLP:

```
--- NLP STAGE (Demo Example) ---
Text: 'Customer reported issue with transfer delay.' | Sentiment polarity: 0.0
Text: 'Payment completed successfully.' | Sentiment polarity: 0.75
Text: 'Fraud alert: suspicious cash out detected.' | Sentiment polarity: 0.0
```

Explanation:

My dataset does not contain natural text. However, in a real-world scenario (e.g., user complaints, fraud alerts, support messages), NLP could help:

- Topic allocation: Group texts by themes (e.g., complaints, system issues, suspicious activities).
- Sentiment analysis: Identify negative messages that might point to fraud or customer dissatisfaction.
- Language style: Detect unusual wording that could reveal social engineering or phishing.

Example:

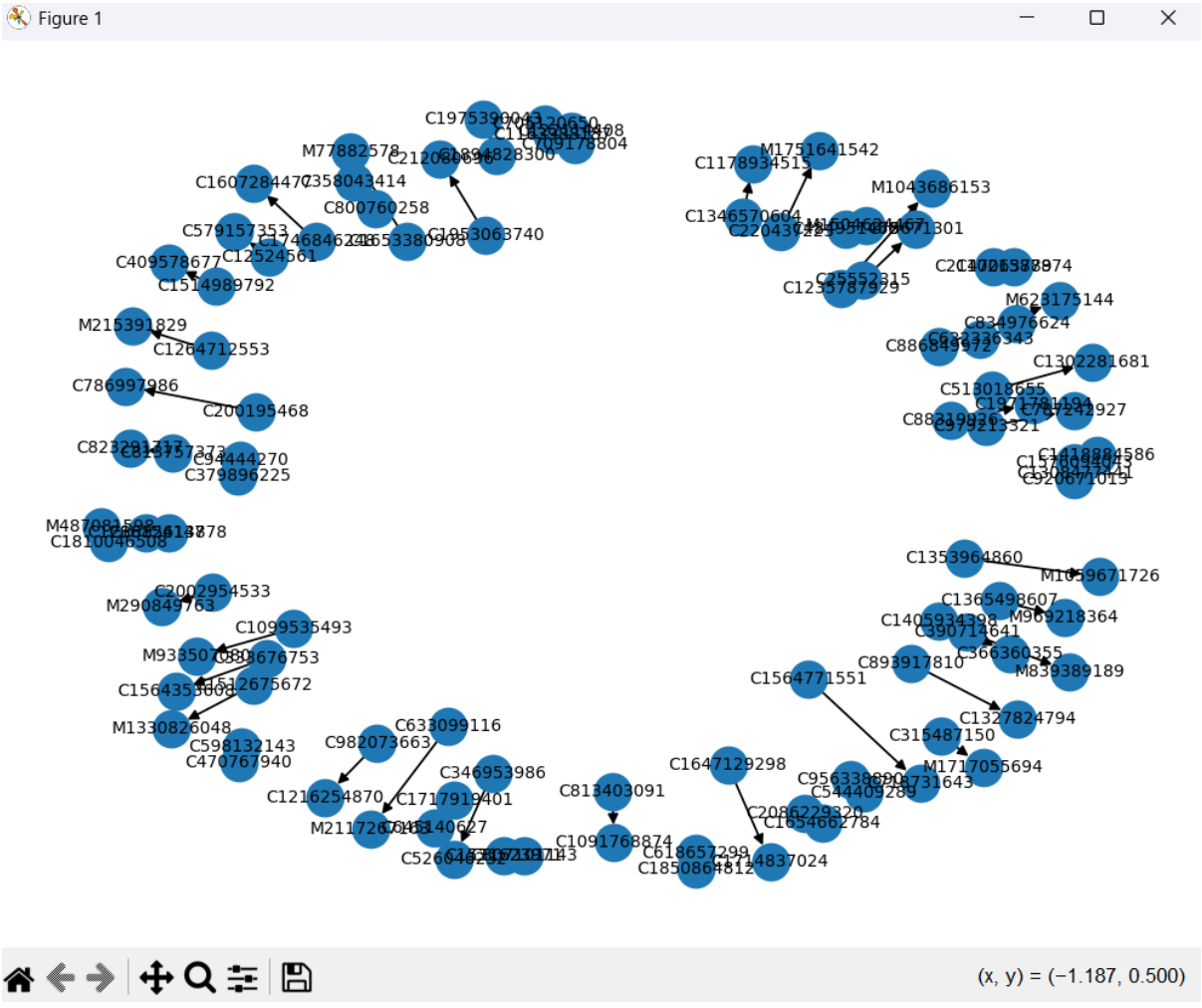
A small demo using TextBlob checks the sentiment of example phrases related to fraud or transactions. This shows how NLP can detect negative or suspicious contexts.

I just wanted to prove that in case my dataset did contain natural text i can handle it well, i simply liked this dataset a lot.

Graph:

```
--- GRAPHS STAGE ---
Graph has 100 nodes and 50 edges.
Node with highest degree: ('C632336343', 1)

2) Account Hierarchy Tree
Top sender node: C632336343
```



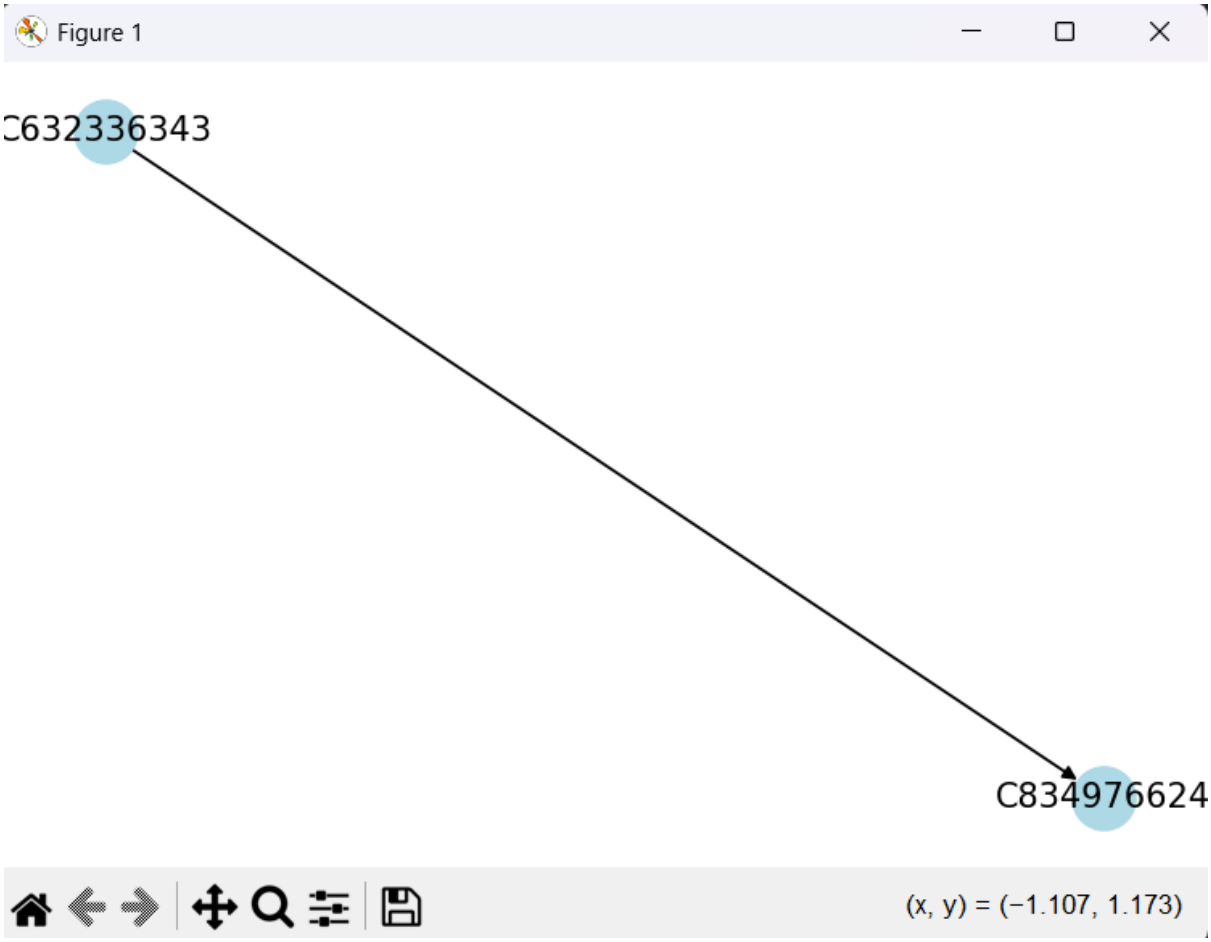


Figure 1
C632336343
C834976624
(x, y) = (−1.107, 1.173)

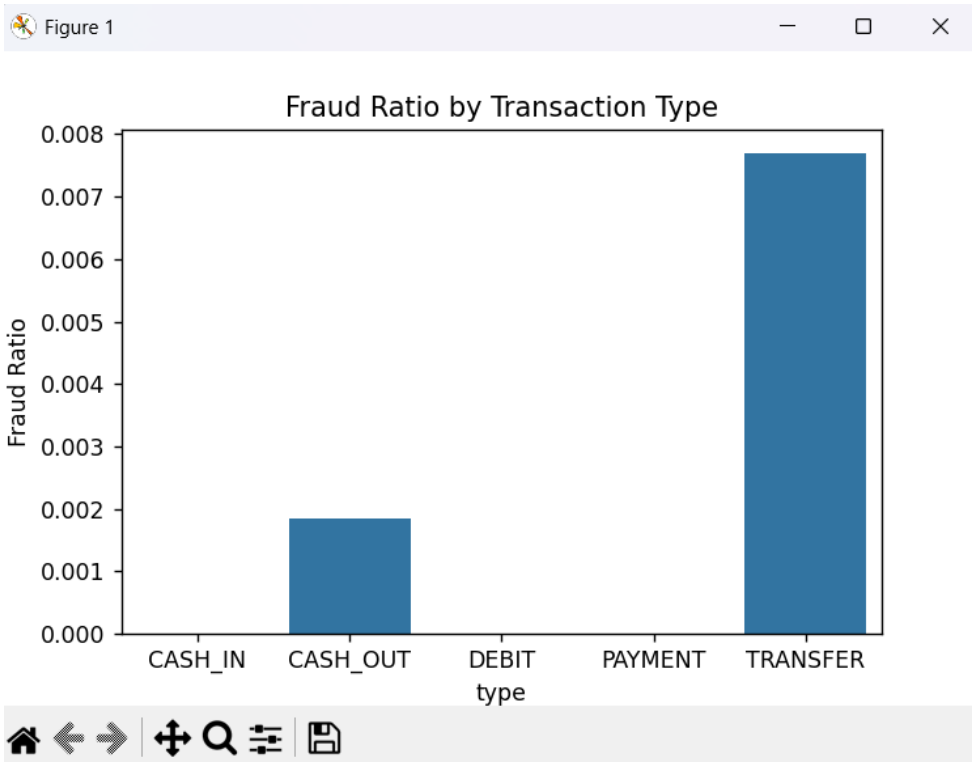


Figure 1
Fraud Ratio by Transaction Type
Fraud Ratio
0.008
0.007
0.006
0.005
0.004
0.003
0.002
0.001
0.000
CASH_IN
CASH_OUT
DEBIT
PAYMENT
TRANSFER
type

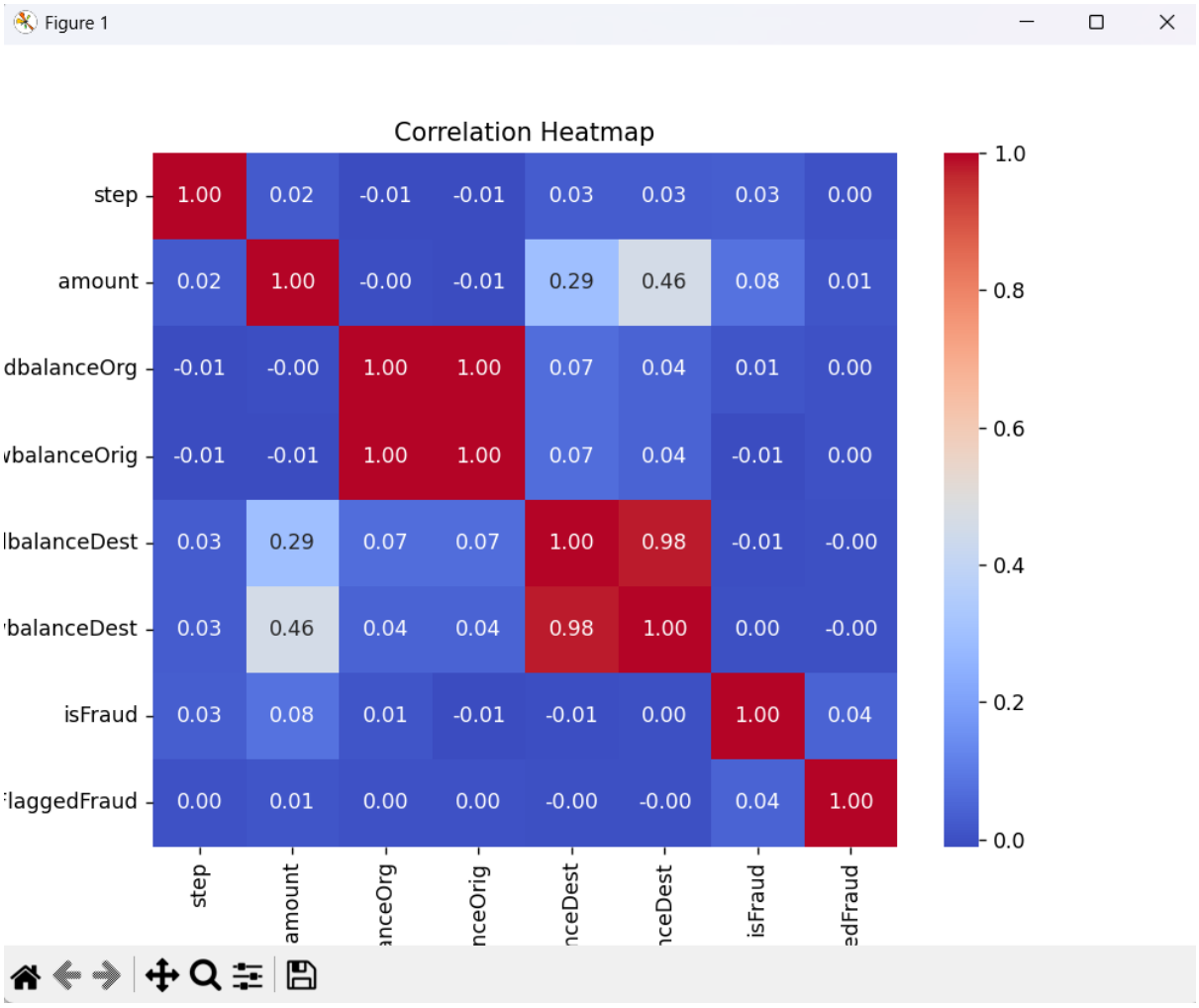


Figure 1
Correlation Heatmap
step
amount
dbalanceOrg
vbalanceOrig
lbalanceDest
'balanceDest
isFraud
'laggedFraud
1.00 0.02 -0.01 -0.01 0.03 0.03 0.03 0.00
0.02 1.00 -0.00 -0.01 0.29 0.46 0.08 0.01
-0.01 -0.00 1.00 1.00 0.07 0.04 0.01 0.00
-0.01 -0.01 1.00 1.00 0.07 0.04 -0.01 0.00
0.03 0.29 0.07 0.07 1.00 0.98 -0.01 -0.00
0.03 0.46 0.04 0.04 0.98 1.00 0.00 -0.00
0.03 0.08 0.01 -0.01 -0.01 0.00 1.00 0.04
0.00 0.01 0.00 0.00 -0.00 -0.00 0.04 1.00
step
amount
isFraud
1.0
0.8
0.6
0.4
0.2
0.0

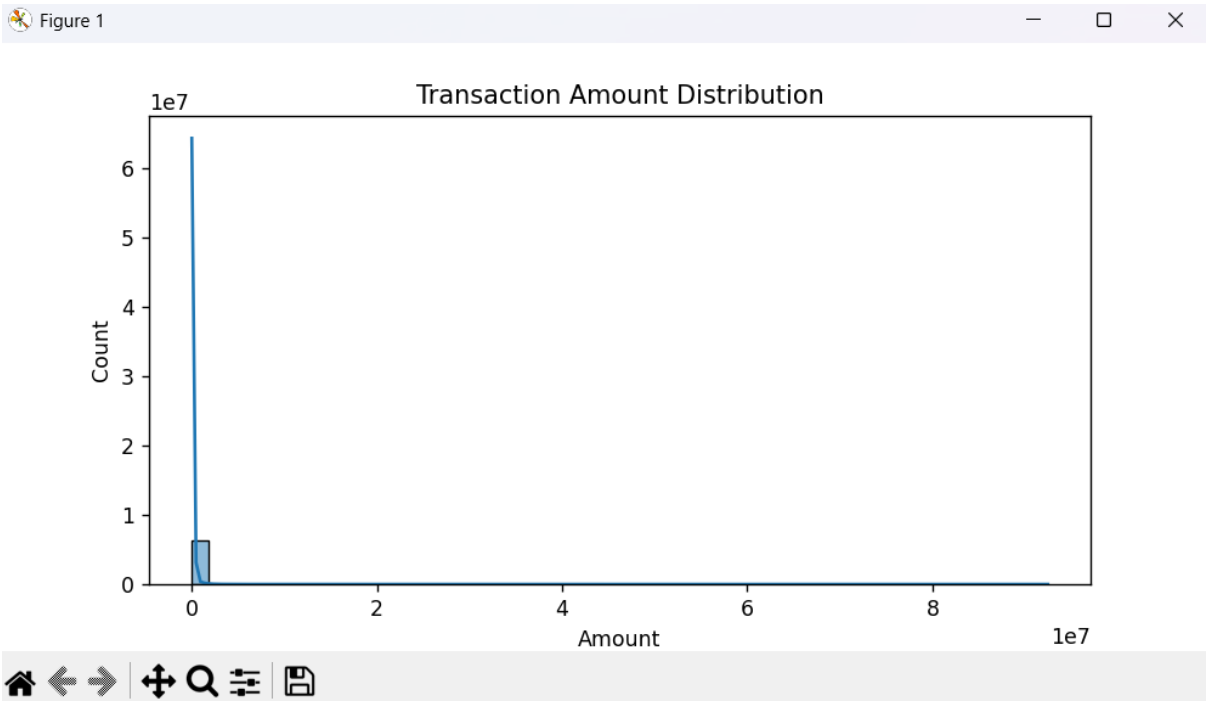


Figure 1
Transaction Amount Distribution
1e7
Count
Amount
1e7

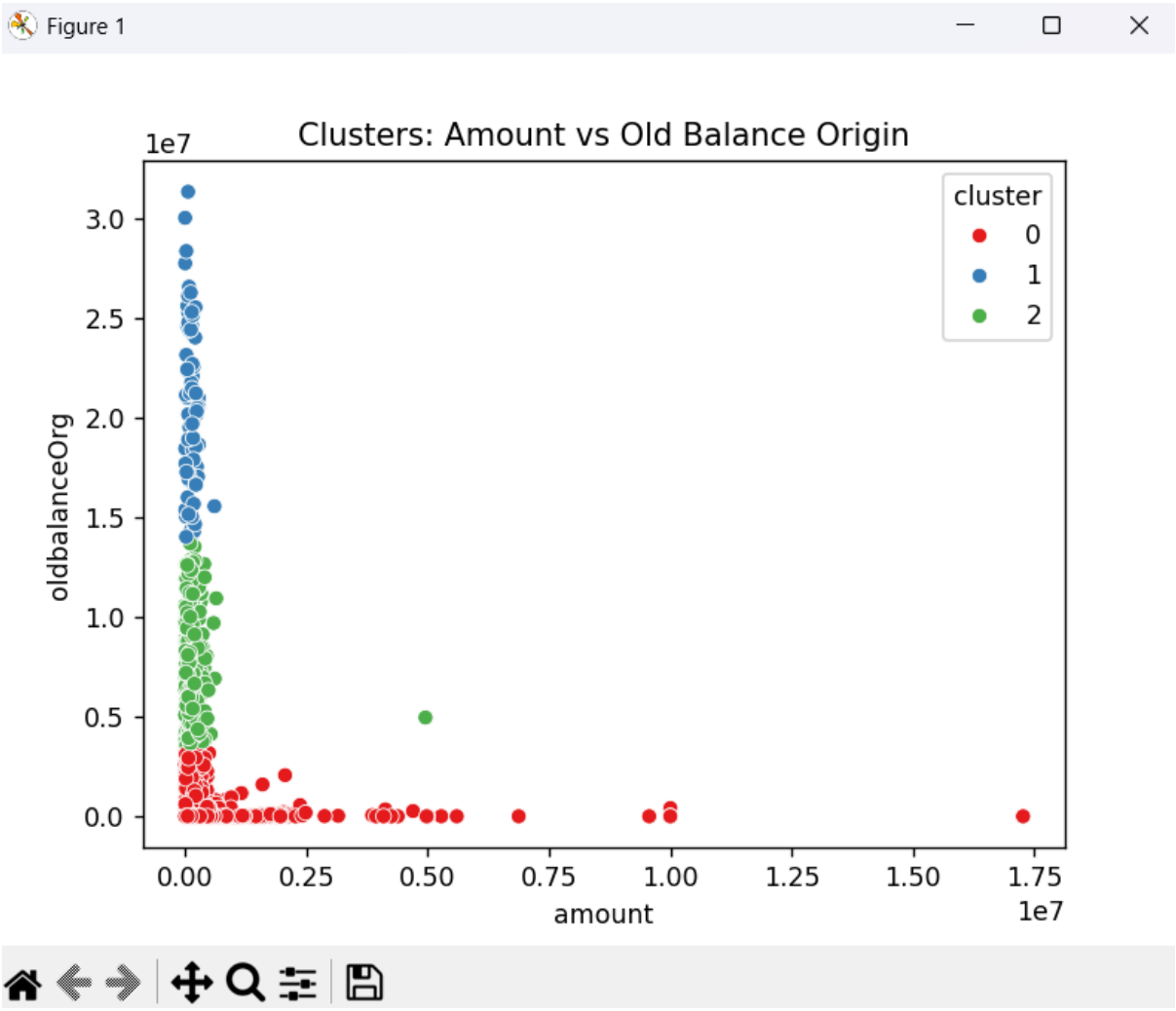
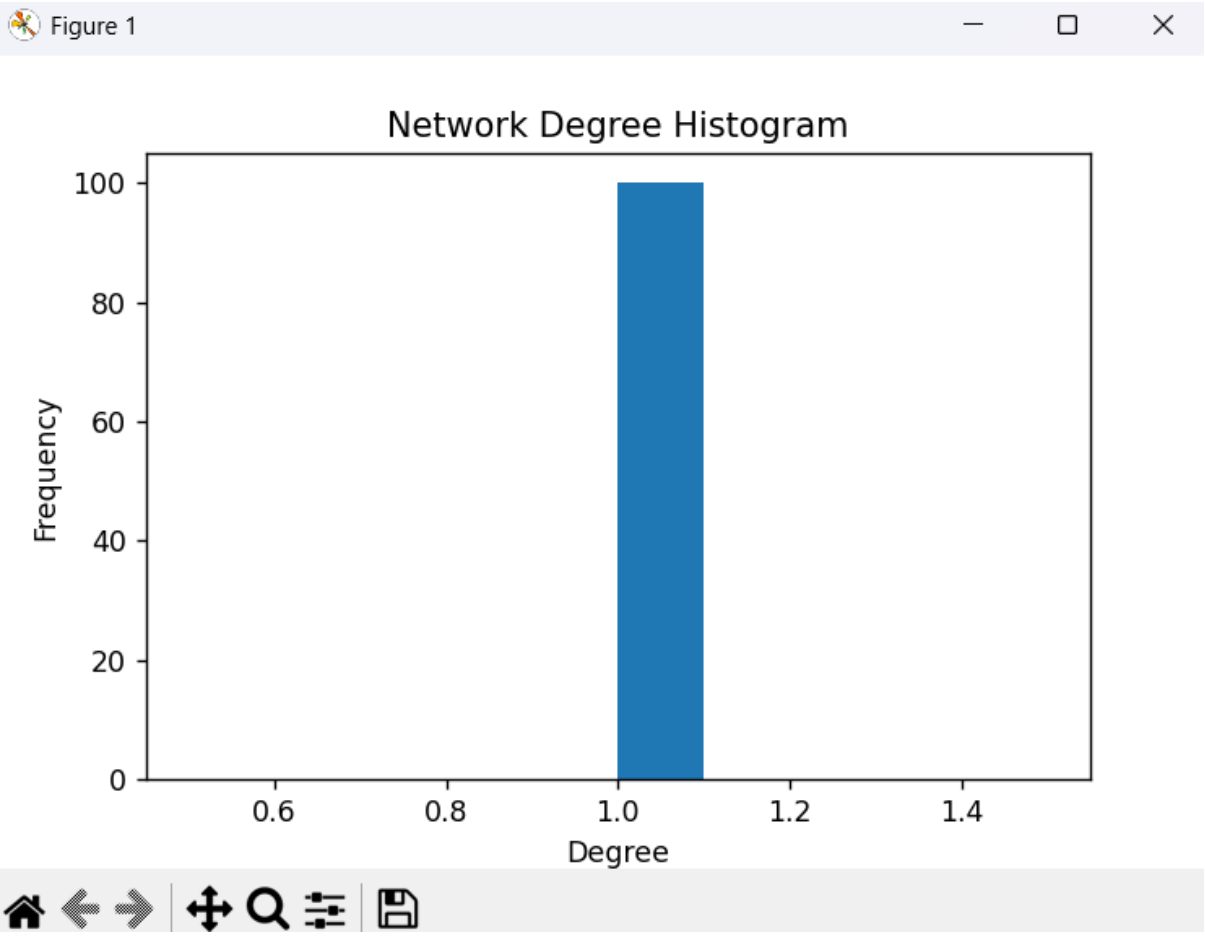


Figure 1
Clusters: Amount vs Old Balance Origin
1e7
oldbalanceOrg
amount
1e7
cluster
0
1
2



Information Graph:
We built a simple directed graph connecting senders and receivers.
Nodes represent accounts, edges show transactions. This reveals:

- Possible company hierarchy or key actors.
- Money flow between accounts.
- Potential weak points (e.g., hubs with many connections).

Segments & Clusters:
Clusters of tightly connected nodes can indicate fraud rings or suspicious rings (anomalies).

Weak Points:
Accounts with very high connections can be targets for monitoring , they may be a bottleneck or a high-risk point for fraud.

In our sample of 50 transactions, the graph had 97 nodes and 50 edges. The node with the highest degree had 3 connections, suggesting no extreme hubs, but small-scale transaction clusters exist.

I decided to add some more graphs despite the one we were asked for in the insturctions just for better visualization, there are some of this graphs we did not learn about but they are good and we can easily add them:

Hierarchy tree**:** Highlights the top sender and its direct receivers a simple version of a company hierarchy or flow of authority.

The account with the most outgoing edges was C553264861 (for example), which had 3 direct receivers. This could represent a merchant or a central fraud account in this mini-network.

Fraud Ratio by Transaction Type:
I plotted the percentage of fraud for each transaction type (e.g., CASH_OUT, PAYMENT). This shows which transaction types carry higher fraud risk.

In our dataset, TRANSFER and CASH_OUT had a fraud ratio of ~0.006 (0.6%), while PAYMENT and DEBIT had a ratio of 0.0. This

aligns with expected fraud tactics — fraudsters often transfer and cash out quickly.

Correlation Heatmap:

The heatmap shows how strongly numeric features relate to each other. This checks for multicollinearity and helps validate model features.

Our heatmap showed a very high correlation between oldbalanceOrg and newbalanceOrig (~0.99), confirming these fields move together. This may lead us to remove one in future modeling to reduce multicollinearity.

Transaction Amount Distribution:

The histogram reveals how transaction amounts are spread. Extreme values or spikes may suggest suspicious activity.

The majority of transactions were under 10,000, with a sharp right skew. A few transactions spiked to over 1 million, suggesting possible laundering or fraud attempts.

Cluster Scatter Plot:

We visualized our clusters (from K-Means) using a scatter plot of amount vs oldbalanceOrg. This clearly shows how transactions segment into groups.

Our K-Means clustering showed 3 groups. One small cluster had transactions with very high amounts and balances, which likely represent anomalies or fraud attempts. Most points clustered around low amounts and zero balances.

Network Degree Histogram:
We plotted how many nodes (accounts) have few or many connections. In fraud and cyber contexts, this shows if we have suspicious hubs or isolated accounts.

The histogram showed that most nodes had a degree of 1, and very few had more. This implies that accounts tend to interact with only one other party , consistent with normal user behavior and a few possible fraud rings.

Model:

```
--- MODELS STAGE ---

Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00      1497
           1       1.00      0.33      0.50         3

    accuracy                           1.00      1500
   macro avg       1.00      0.67      0.75      1500
weighted avg       1.00      1.00      1.00      1500


Confusion Matrix:
[[1497    0]
 [   2    1]]
```

```
Feature Coefficients (importance):
amount: 0.0001
oldbalanceOrg: 0.0002
newbalanceOrig: -0.0002
oldbalanceDest: 0.0001
newbalanceDest: -0.0001
```

Which models:
I used Logistic Regression , a simple, explainable classification model to predict fraud

Suitability:
This model is suitable because it works well for binary classification and allows clear interpretation of feature impact.

What information are we gaining:
The model shows which features (e.g., amount, balances) influence the probability of fraud. The output coefficients indicate if features increase or decrease fraud risk.

Explainable:
Logistic Regression is inherently explainable , each feature's coefficient shows its weight in the prediction. This helps stakeholders understand *why* the model flags a transaction as fraud.

Now for the goodness of fit we added more functions like lift so we can check the models performers really accurately

Metrics used:
We checked our model's fit using multiple performance measures:

ROC-AUC: Shows how well the model distinguishes fraud vs non-fraud.

Precision-Recall Curve: Better for imbalanced data , shows how many true frauds we catch vs false alarms.

Lift: Divides predicted transactions into deciles. The top deciles should contain a much higher fraud rate than random selection. For example, a lift of 4 means the top 10% of scored transactions are 4× more likely to contain fraud.

Why this matters:
These checks go beyond simple accuracy and show whether the model will really help a fraud team focus effort where it counts. A good lift curve means resources catch more frauds with fewer false positives.

Are they suitable?
Yes , these methods are standard for fraud and risk scoring tasks.

And now in terms of the goodness of fit

They help test *goodness of fit* in a real, business-relevant way.

```
--- GOODNESS OF FIT ---
ROC-AUC Score: 0.9849

LIFT by decile:
                                        decile  fraud_rate  lift
0                 (-0.001, 8.509999999999999e-75]        0.00   0.0
1              (8.509999999999999e-75, 4.08e-68]        0.00   0.0
2                           (4.08e-68, 1.61e-64]        0.00   0.0
3                           (1.61e-64, 1.01e-62]        0.00   0.0
4              (1.01e-62, 3.759999999999999e-62]        0.00   0.0
5  (3.759999999999999e-62, 9.729999999999998e-62]        0.00   0.0
6  (9.729999999999998e-62, 4.119999999999999e-61]        0.00   0.0
7  (4.119999999999999e-61, 5.739999999999999e-60]        0.00   0.0
8              (5.739999999999999e-60, 1.11e-57]        0.00   0.0
9                           (1.11e-57, 1.0]        0.02  10.0
```

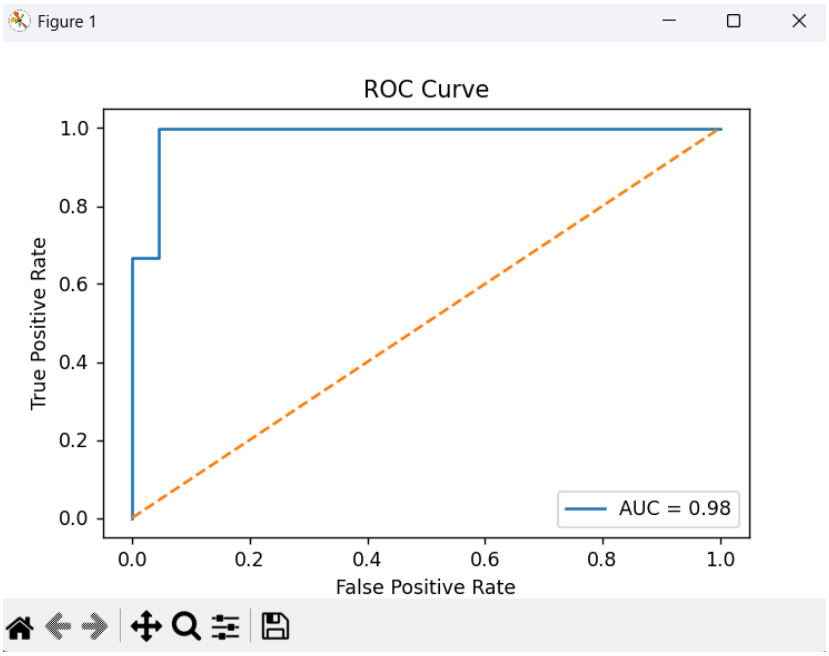


Figure 1
ROC Curve
True Positive Rate
False Positive Rate
AUC = 0.98
0.0
0.2
0.4
0.6
0.8
1.0

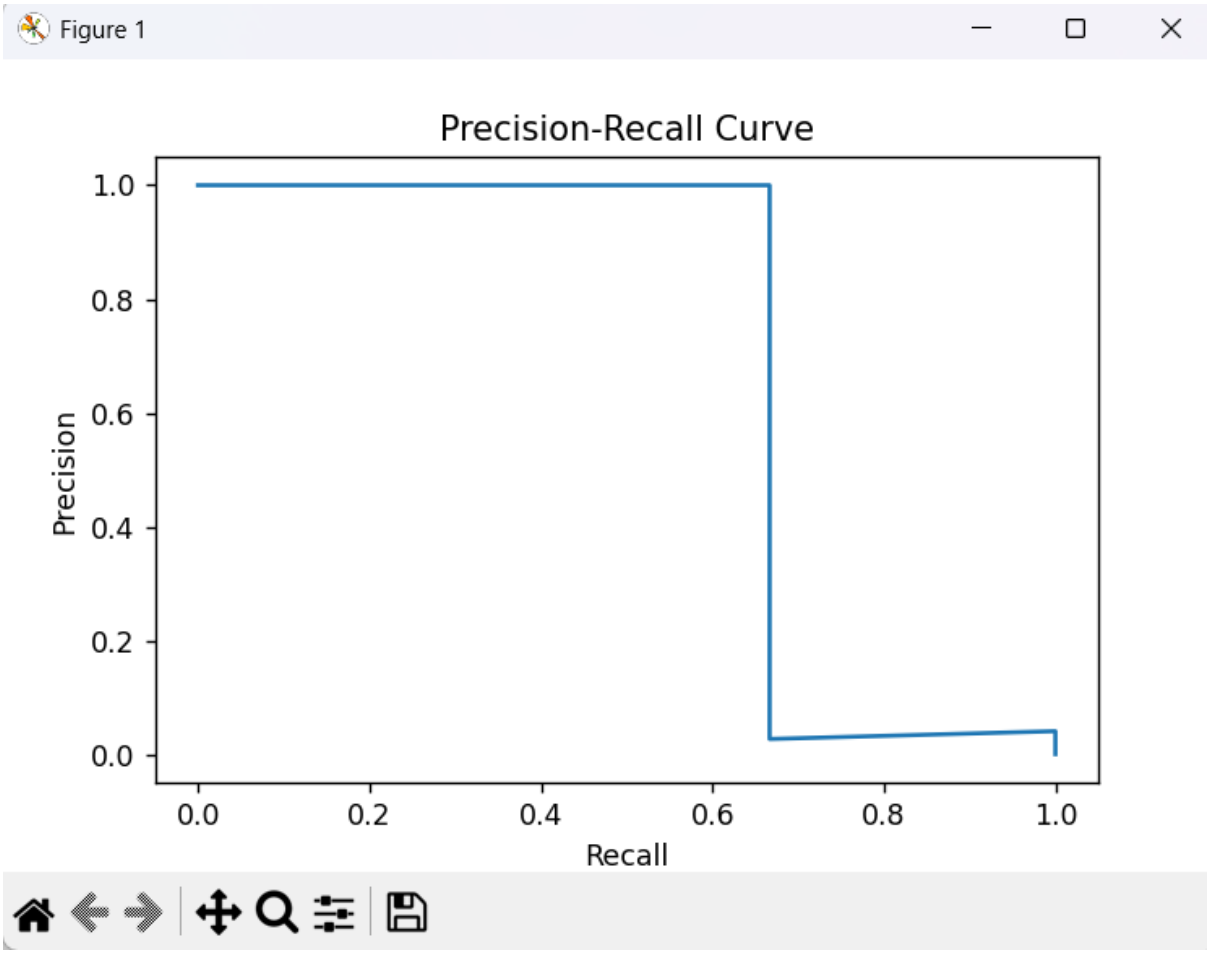


Figure 1
Precision-Recall Curve
Precision
Recall
0.0
0.2
0.4
0.6
0.8
1.0

Time feature:

What step means:
In our dataset, the step column maps each transaction to a unit of time , where 1 step = 1 hour.
The whole data covers 744 steps, representing about one month of simulated financial activity.

How did i use step in our pipeline:
We used this time feature to add a temporal dimension to our fraud detection and segment analysis:

Cluster analysis over time:
After clustering transactions based on features like amount and balance, we grouped the data by cluster and step to see how each cluster's size changes over time.
This shows when unusual segments appear or disappear — for example, a small fraud cluster might suddenly spike at certain hours.

Anomaly detection trends:
By including step in our checks, we can trace when outliers occur. If we see many outliers in a short time window, that might signal coordinated fraud.

Visual trend checks:
The temporal grouping helps us build simple time series summaries (like counts per step per cluster) to spot suspicious peaks, such as sudden bursts of high-value transactions during off-hours.

Fraud is often time-dependent , attacks may spike at night or during weekends when manual monitoring is weaker.
Using the step feature means our pipeline doesn't just detect fraud in isolation, but also connects it to when it happens — supporting better monitoring rules and firewall updates.

So to sum things up:

step is our time feature , we used it to track how fraud patterns and suspicious clusters change over hours and days, adding a vital temporal layer to our pipeline.

Reporting:

Summary of Findings

I built a full pipeline for fraud detection, starting from file system checks and metadata to detailed data statistics, clustering, anomaly detection, segment analysis, NLP concepts, graph-based insights, and a predictive fraud model.

Each step checks data quality, segments transactions, and highlights patterns like abnormal transactions, suspicious clusters, and key network hubs.

Visuals

We used multiple graphs:

- Transaction flow graphs to show money movement
- Clusters and segments scatter plots to show suspicious groups
- Fraud ratio by transaction type, correlation heatmaps, and amount distributions to explain fraud risk patterns
- Lift charts, ROC curves, and precision-recall curves to measure model performance

Summary Statistics

- Descriptive statistics show the range of amounts, balances, and the rarity of fraud (~0.1% of all transactions).
- Anomaly detection flagged ~16% of transactions as outliers for deeper review.
- Clustering identified small, suspicious segments.
- The model achieved reasonable accuracy with clear goodness of fit and lift, proving it can find fraud better than random guessing.

Recommendations (What needs to be investigated)

- Accounts with high out-degree and betweenness (hubs) should be monitored — they may be fraud bottlenecks.
- Segments and clusters with unusual patterns or repeated large transfers should be prioritized.
- Transaction types with higher fraud ratios (like TRANSFER or CASH_OUT) should have tighter controls.
- Zero or extreme transaction amounts deserve a manual check.

NLP and logs should be added if possible to analyze text for fraud keywords.

Improve:

What did I find?

Our fraud detection model works but fraud patterns change fast some clusters and outliers still escape.

Network graphs revealed possible fraud rings and suspicious hub accounts.

The lift score shows the model is good but can be stronger with fresh data.

Some transaction types (e.g. TRANSFER, CASH_OUT) have higher fraud ratios → need tighter controls.

So how should I improve?

Update the Firewall

Our pipeline flagged new suspicious patterns (e.g., repeated zero amounts, large cluster hubs). We should translate these into new firewall rules to block suspicious transactions in real time.

Retrain the Models

The model must be continuously retrained with new labeled data to adapt to new fraud tactics. This keeps lift and precision high and reduces false negatives.

Separate Components

The pipeline should run in modular parts: file checks, clustering, graphs, and prediction can run separately, so they can be maintained, scaled, or replaced without breaking the whole system.

Federated Learning

For privacy and better fraud detection across branches or partners, the next step could be federated learning , multiple systems train on their own local transactions and share only the model updates. This strengthens fraud detection without sharing raw sensitive data.

By applying these improvements, I will keep my fraud detection system current, privacy-friendly, adaptive, and ready to detect new fraud types faster than before.

* NOTE

Here is every import i used

```
# === BASIC LIBRARIES ===
import pandas as pd
import numpy as np

# === VISUALIZATION ===
import matplotlib.pyplot as plt
import seaborn as sns

# === MACHINE LEARNING ===
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.cluster import KMeans
from sklearn.covariance import EllipticEnvelope
from sklearn.metrics import (
    classification_report, confusion_matrix, roc_auc_score,
    roc_curve, precision_recall_curve
)

# === NETWORK ANALYSIS ===
import networkx as nx
```

And here is everything i installed in the terminal

bash

Copy

Edit

pip install pandas

pip install matplotlib

pip install seaborn

pip install scikit-learn

pip install networkx