



Data Science and AI Stream

Homework 2 – Week3

Real-Life Fraud Detection Analysis

Hind Abdallah Al Araimi

30-Apr-2025

Dataset Link: <https://www.kaggle.com/datasets/shenba/time-series-datasets?select=daily-minimum-temperatures-in-me.csv>

1. Business Context and Objectives

Fraud has become a major challenge in today's banking sector, leading to significant financial losses and reputational damage for both financial institutions and their customers. As fraudulent schemes grow more complex and sophisticated, traditional rule-based detection methods are proving less effective at identifying and stopping such activities. Consequently, there is an increasing need for advanced solutions like machine learning to improve fraud detection and prevention in the banking industry.

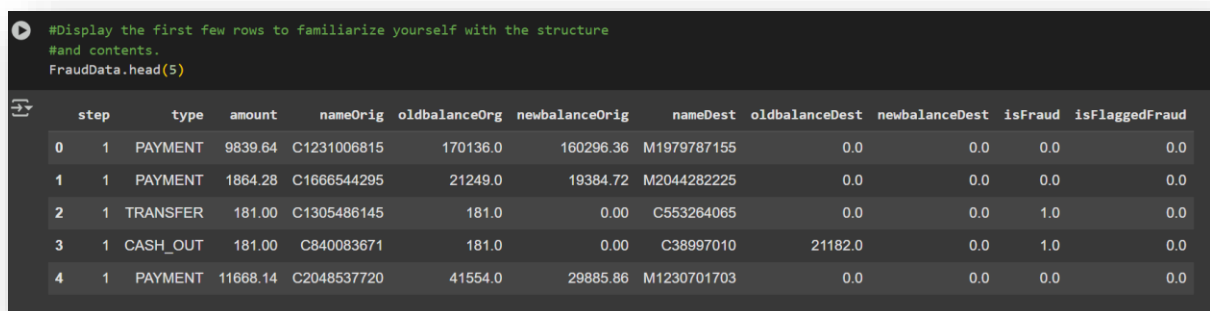
To better understand fraud behaviour, this analysis investigates key questions:

- What types of transactions are most frequently associated with fraud?
- Are there specific patterns in the timing (step) or the amount of transactions that indicate fraudulent activity?

2. Data Preparation

1. Loading and Inspecting the Dataset:

To get started, I loaded the dataset into a pandas DataFrame and took a quick look at the first few rows to understand its structure.



	step	type	amount	nameOrig	oldbalanceOrig	newbalanceOrig	nameDest	oldbalanceDest	newbalanceDest	isFraud	isFlaggedFraud
0	1	PAYMENT	9839.64	C1231006815	170136.0	160296.36	M1979787155	0.0	0.0	0.0	0.0
1	1	PAYMENT	1864.28	C1666544295	21249.0	19384.72	M2044282225	0.0	0.0	0.0	0.0
2	1	TRANSFER	181.00	C1305486145	181.0	0.00	C553264065	0.0	0.0	1.0	0.0
3	1	CASH_OUT	181.00	C840083671	181.0	0.00	C38997010	21182.0	0.0	1.0	0.0
4	1	PAYMENT	11668.14	C2048537720	41554.0	29885.86	M1230701703	0.0	0.0	0.0	0.0

Figure 1: list of the columns in the dataset

2. Column Overview and Relevance:

I used `info()` to get list the columns and datatype.

```
#List the columns and explain the importance of each in the context
#of fraud detection.
FraudData.info()

<class 'pandas.core.frame.DataFrame'>
Index: 97224 entries, 0 to 97223
Data columns (total 11 columns):
#   Column              Non-Null Count  Dtype
---  -
0   step                97224 non-null  int64
1   type                97224 non-null  object
2   amount              97224 non-null  float64
3   nameOrig             97224 non-null  object
4   oldbalanceOrg        97224 non-null  float64
5   newbalanceOrig       97224 non-null  float64
6   nameDest             97224 non-null  object
7   oldbalanceDest       97224 non-null  float64
8   newbalanceDest       97224 non-null  float64
9   isFraud              97224 non-null  float64
10  isFlaggedFraud       97224 non-null  bool
dtypes: bool(1), float64(6), int64(1), object(3)
memory usage: 8.3+ MB
```

Figure 2: Column overview and relevance

Below is a list of the columns in the dataset, along with a brief explanation of each in the context of fraud detection:

Column Name	Description
step	Represents time (in hours). It helps spot when frauds are more likely to happen.
type	Type of transaction like PAYMENT, TRANSFER, etc. Some types may be riskier than others.
amount	The transaction amount — unusually large values might signal fraud.
nameOrig / nameDest	The sender and receiver. Useful for spotting repeat offenders.
oldbalanceOrg newbalanceOrig	/ Sender's balance before and after the transaction. Helps check if money movement makes sense.
oldbalanceDest newbalanceDest	/ Receiver's balance changes — sudden jumps could be suspicious.
isFraud	Tells us if the transaction is actually fraud (1) or not (0).

3. Data Cleaning & Exploration

1. Initial Review:

I checked for missing values and found one row with several NaNs. Since it had too many gaps and the dataset is large, I dropped it — it wouldn't affect the results.

```
[6] FraudData[FraudData.isnull().any(axis=1)]

#Insights:
#Only one row with missing values.
#The row has many nulls, not just one.
#dataset is large, so dropping one row has no significant impact.
```

	step	type	amount	nameOrig	oldbalanceOrg	newbalanceOrig	nameDest	oldbalanceDest	newbalanceDest	isFraud	isFlaggedFraud
	97224	10 PAYMENT	6733.59	C708911726		0.0	0.0	M2034482538	0.0	NaN	NaN

```
[7] FraudData.dropna(inplace=True)

[8] FraudData[FraudData.isnull().any(axis=1)]
# No null values.
```

	step	type	amount	nameOrig	oldbalanceOrg	newbalanceOrig	nameDest	oldbalanceDest	newbalanceDest	isFraud	isFlaggedFraud
--	------	------	--------	----------	---------------	----------------	----------	----------------	----------------	---------	----------------

Figure 3: dropping null values

After reviewing all the columns, I found that each one contributes meaningfully to fraud detection — whether it's tracking transaction patterns, monitoring account behaviour, or labelling fraud. Therefore, I chose to keep all of them for analysis

2. Exploratory Analysis

1- Total Transactions: Counted all transactions and grouped them by type (e.g., PAYMENT, TRANSFER).

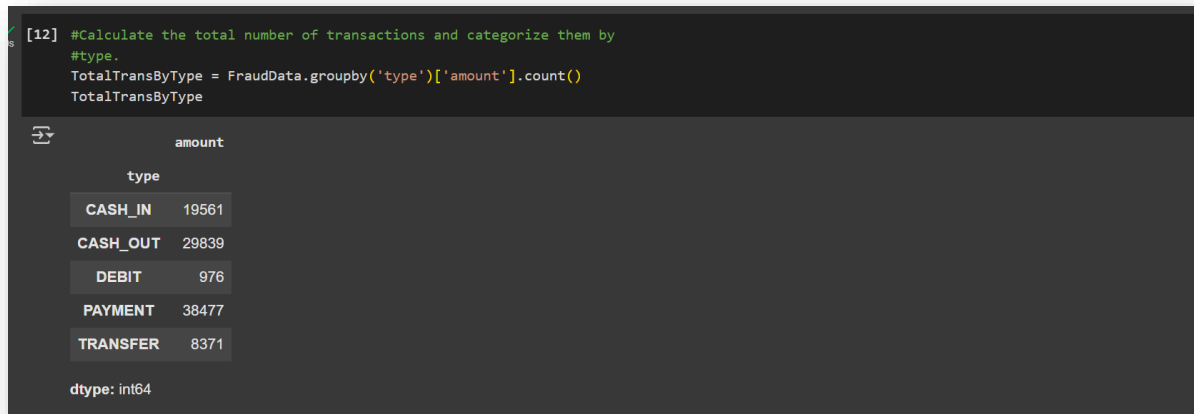


Figure 4: Total Transaction by type

2- Fraud Percentage: Calculated what percentage of transactions were fraud, then broke it down by type to see which are riskier.

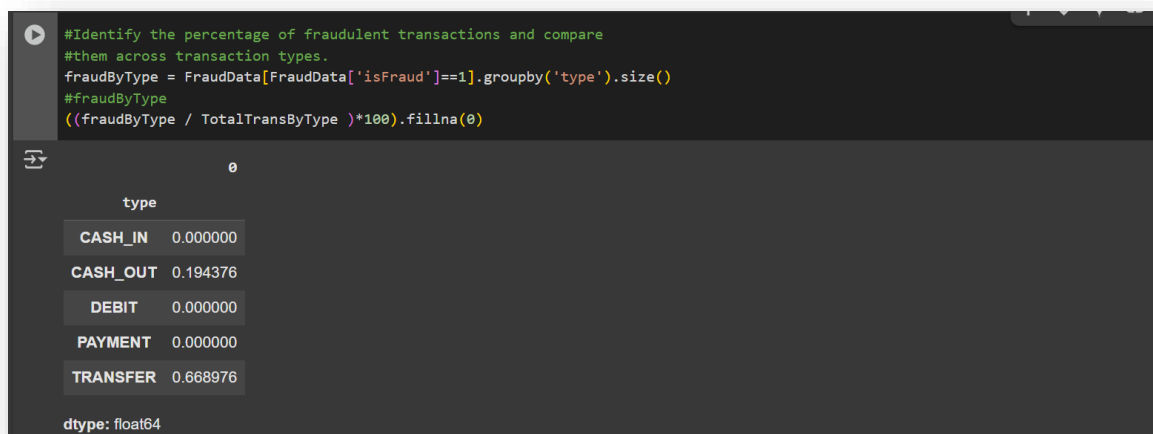


Figure 5: Percentage of Fraud transaction by type

Findings:

After analysing the data, I found that most transaction types had **no fraud at all**, especially CASH_IN, DEBIT, and PAYMENT.

However, two types stood out:

- **TRANSFER** transactions had the **highest fraud rate**, with around **0.67%** of them being fraudulent.
- **CASH_OUT** transactions came next, with a fraud rate of **0.19%**.

This shows that fraud is mainly happening in money transfers and cash withdrawals, which makes sense since both involve moving funds between users and accounts.

3- Amount Distribution: For both fraudulent and non-fraudulent transactions, I compared:

- Mean
- Median
- Standard deviation

This helped understand if frauds tend to involve unusually high amounts.

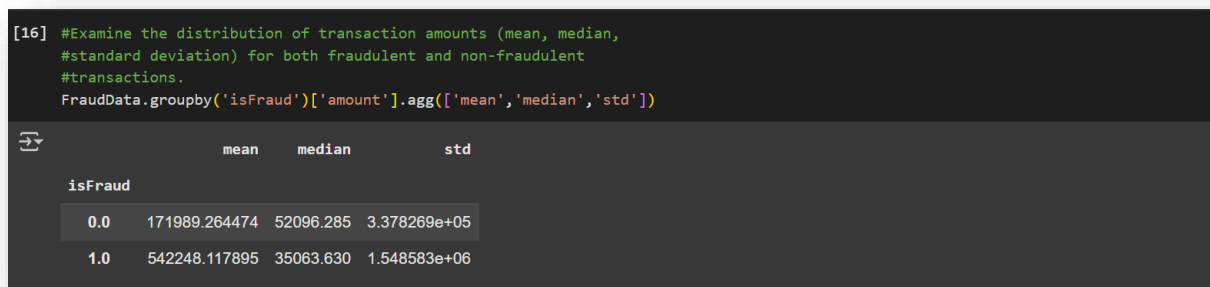


Figure 6: Distribution of transaction amounts (mean, median, Standard deviation)

4. Real-Life Fraud Detection Analysis

1. Detecting Suspicious Patterns

To identify high-risk transactions, I flagged any transaction where the amount exceeded 200,000 as suspicious..

```
[18] #Identify and flag transactions exceeding the legal limit (amount >
#200,000) as potentially fraudulent (isFlaggedFraud).

FraudData['isFlaggedFraud'] = FraudData['amount'] > 200000
```

	step	type	amount	nameOrig	oldbalanceOrig	newbalanceOrig	nameDest	oldbalanceDest	newbalanceDest	isFraud	isFlaggedFraud
0	1	PAYMENT	9839.64	C1231006815	170136.0	160296.36	M1979787155	0.0	0.0	0.0	False
1	1	PAYMENT	1864.28	C1666544295	21249.0	19384.72	M2044282225	0.0	0.0	0.0	False
2	1	TRANSFER	181.00	C1305486145	181.0	0.00	C553264065	0.0	0.0	1.0	False
3	1	CASH_OUT	181.00	C840083671	181.0	0.00	C38997010	21182.0	0.0	1.0	False
4	1	PAYMENT	11668.14	C2048537720	41554.0	29885.86	M1230701703	0.0	0.0	0.0	False
...
97219	10	PAYMENT	10811.91	C504389296	1009.0	0.00	M1318927100	0.0	0.0	0.0	False

Figure 7: Identify risky transaction where amount exceed 200k

I then focused on fraud-related transactions ($\text{isFraud} = 1$) to uncover patterns based on the time step, type, and transaction amount

- **Time Step:** Most fraudulent transactions appeared clustered around specific steps, suggesting that fraud might be more likely at certain operational hours or days.



Figure 8: number of transaction per step

- **Transaction Type:** Fraud occurred almost entirely in TRANSFER and CASH_OUT types, indicating that these are the most abused methods.

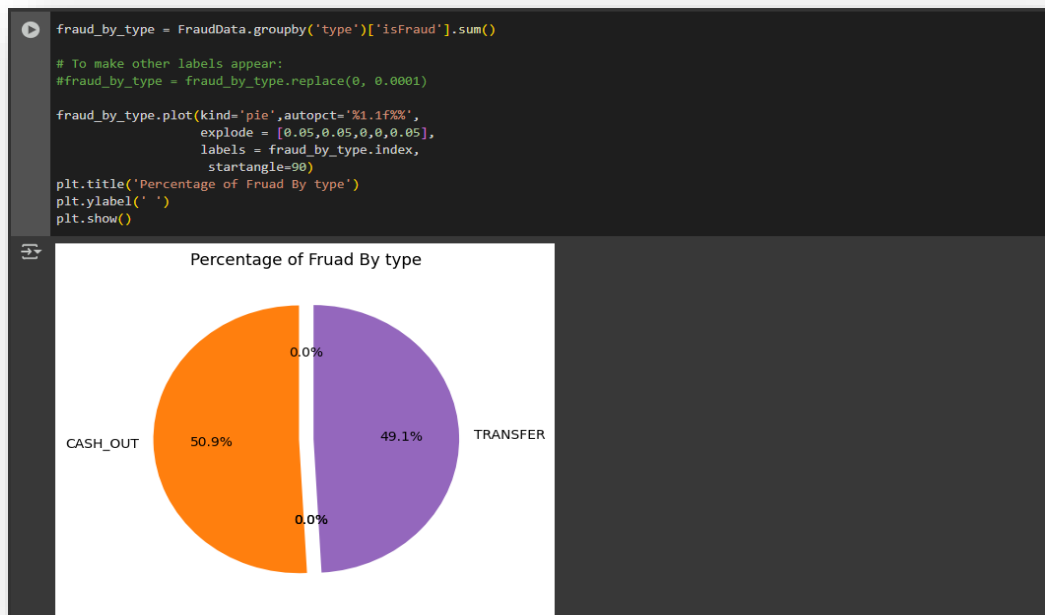


Figure 9: Percentage of fraud by type of transaction

2. Group Analysis:

- Group transactions by type and identify which types have the highest volume and value.

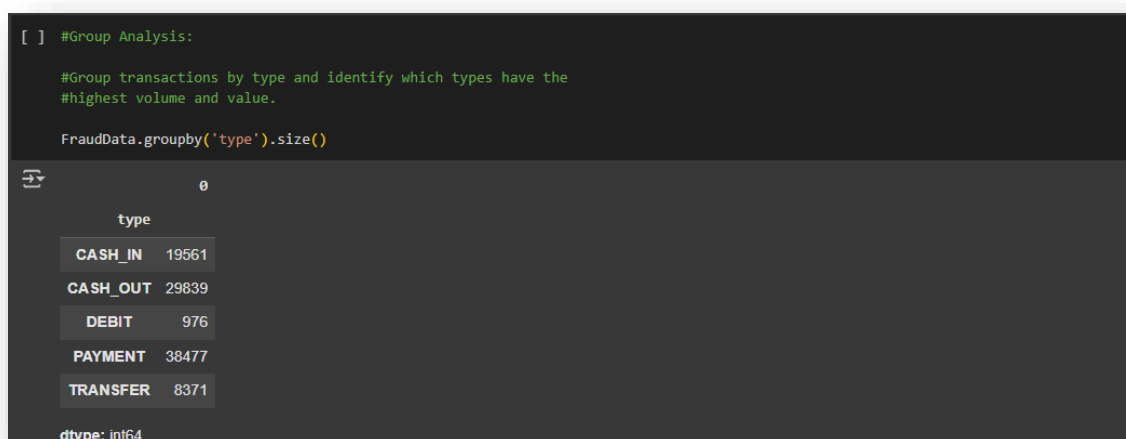


Figure 10: Group of transaction type by highest volume



Figure 11: Group of transaction types by highest value

- Examine whether certain customers (nameOrig or nameDest) are repeatedly involved in flagged or fraudulent transactions.

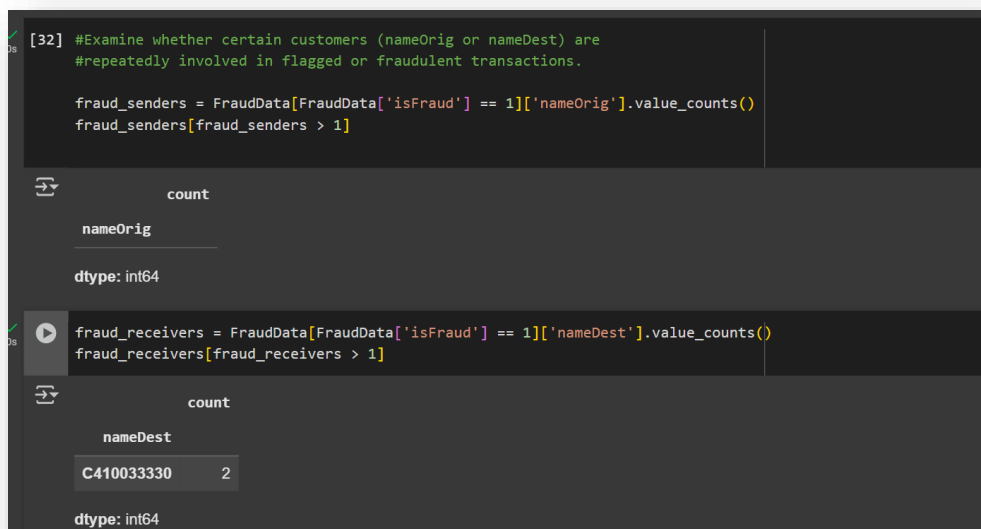


Figure 12: Number of Times Fraudulent Sender and Receiver Accounts Appear More Than Once

After analyzing the nameOrig (sender) and nameDest (receiver) fields in fraudulent transactions:

- No sender (nameOrig) was involved in more than one fraudulent transaction indicating that most fraudulent senders only appear once.
- However, one receiver account (C410033330) appeared twice as a destination in fraudulent transactions. This repetition may suggest a suspicious or high-risk account, worth flagging for further investigation.

3. Critical Thinking Task:

Legitimate Transactions That Might Look Fraudulent:

Sometimes, a user might send a large amount (like over 200,000) to their own account, maybe for savings or business. It looks suspicious but is actually fine.

To avoid false alarms, the system should check if both accounts belong to the same person or if it's something the user does often.

5. Insights and Recommendations

Insights:

Although fraud only occurs in a small portion of total transactions, it's almost entirely concentrated in (TRANSFER) and (CASH_OUT) types. These are also the types where money is moved directly between users, making them riskier. Fraudulent transactions tend to involve higher amounts and often appear during specific time steps, suggesting that some fraud patterns follow a routine. Additionally, one receiver account appeared multiple times in fraud cases, which could indicate suspicious activity.

Recommendations:

1. **Monitor High-Risk Transaction Types:** Focus fraud detection efforts on TRANSFER and CASH_OUT transactions, especially large ones.
2. **Use Time-Based Alerts:** Apply stricter fraud checks during time steps with higher fraud activity.
3. **Flag Repeat Accounts:** Watch for receiver or sender accounts involved in more than one fraud case, as they may signal ongoing fraud.

References:

Iwuchukwu, V., & Onyeabor, C. A. (2024). *Machine learning approach for fraud detection system in financial institution: A web-based application*. ResearchGate.

Singh, S., & Agrawal, S. (2024). *Fraud detection using machine learning techniques in banking sector*. *World Journal of Advanced Research and Reviews*, 21(1), 393–398.