Air University Aerospace and Aviation Campus Kamra

DATA SCIENCE

ASSIGNMENT 01

Submitted On: October 18, 2025

Name: Ahmad Faraz

Registration No: 215154

Department: BSCS-VII

Instructor: Mr. Ghulam Ali

Table of Contents

1 Introduction

1.1 Background

The Ethereum blockchain is the base of the world's most blockchains, providing a foundation for building Dapps (Decentralized Applications). These Dapps are used in finance, supply chain management, real estate, voting, and many other applications. These Dapps are fueled by cryptocurrency, which serves as an incentive for transaction verifiers to add transactions to the blockchain.

As the blockchain is a distributed ledger where every transaction is transparent, that's why it's really the main attraction for the users, that they can verify their ownership, see other transactions, even verify the entire history, logs, and balance of others on the blockchain. The blockchain uses a hexadecimal address, which is the only identity of someone, can be traced by anyone on the chain, but cannot share the actual name, location, etc.

1.2 Problem

The Dapps are security critical; any vulnerability, mismanagement of access control, and lack of required knowledge can lead to scams, loss of assets, and an entire protocol hack by an attacker. The hackers exploit users, split the hacked cryptocurrency to different addresses by using the microtransactions strategy, which makes it even more difficult to trace the attacker in the giant ecosystem of blockchain manually. Companies hire the investigators who manually trace these addresses, which is time-consuming, costly, and usually end empty-handed.

1.3 Role of Data Science

Data science will help very well in this field to automate and make it cost-effective in address verification. Even a naive user can verify the legitimacy of an address before interacting with it or making any financial decisions.

2 Problem Statement

Ethereum chain fraud accounts for ~18% of the entire blockchain ecosystem in 2024, and lost US $228.63 million. Tracing fraudulent transactions on blockchain is cost-effective and time-consuming. Required experts to identify the fraudulent transactions associated with an address manually.

3 Key Questions

1. Can the model classify whether a given blockchain address is fraudulent or legitimate?
2. Which transaction-related features contribute most to identifying fraudulent behavior?
3. Can the model predict the risk level of a new or unseen address before any loss occurs?
4. How accurately and efficiently can the model detect suspicious activity compared to manual investigation?
5. Can the model reduce the cost and time of blockchain investigation while improving detection reliability?

4 Data Collection Process

During my data collection process, I encountered a dataset listed below, which helps to identify fraudulent transactions. This dataset is available on the Kaggle platform.

4.1 Sources

https://www.kaggle.com/datasets/rupakroy/ethereum-fraud-detection/data

4.2 Methodology

I used the secondary data collection method because the data is available on a public platform called Kaggle.

4.3 Challenges Faced

No challenges faced

5 Preprocessing Steps

5.1 Data Cleaning

5.1.1 Missing Values

5.1.1.1 Int and Float Type

Column Name	NaN Values
Total ERC20 tnxs	829
ERC20 total Ether received	829
ERC20 total ether sent	829
ERC20 total Ether sent contract	829
ERC20 uniq sent addr	829

ERC20 uniq rec addr	829
ERC20 uniq sent addr.1	829
ERC20 uniq rec contract addr	829
ERC20 avg time between sent tnx	829
ERC20 avg time between rec tnx	829
ERC20 avg time between rec 2 tnx	829
ERC20 avg time between contract tnx	829
ERC20 min val rec	829
ERC20 max val rec	829
ERC20 avg val rec	829
ERC20 min val sent	829
ERC20 max val sent	829
ERC20 avg val sent	829
ERC20 min val sent contract	829
ERC20 max val sent contract	829
ERC20 avg val sent contract	829
ERC20 uniq sent token name	829
ERC20 uniq rec token name	829

829 out of 9841 = 8.42% missing values.

Cleaning
All columns with the prefix ERC20 represent quantitative transaction-based attributes (e.g., total transfers, unique addresses, minimum/maximum/average token values).
In cases where these values are missing (NaN), it indicates that no ERC20 transactions occurred for that address.
Since the absence of an ERC20 transaction inherently means:

- No tokens were sent or received.
- No transactional averages or totals can be computed.

It is semantically correct to replace missing (NaN) entries with 0. This replacement does not distort the data; instead, it accurately encodes "no activity" for those addresses

5.1.1.2 Object Type

Column Name	0 values	Empty strings	Null (NaN) values	Valid values	% Empty strings	% Null (NaN)
ERC20 most rec token type	4399	21	871	4550	0.21%	8.85%
ERC20 most sent token type	4399	1191	2697	1554	12.10%	27.40%
Address	0	0	0	9841	0.00%	0.00%

Cleaning

In both “ERC20_most_rec_token_type” and “ERC20_most_sent_token_type”, the majority of values are '0', which are of string type. Both columns are intended to record the type of token, such as USDT, wBTC, or any other. However, when the token name is unknown, '0' is used. Since we may encounter unknown token names represented as '0', null, or empty strings, it is best to convert all such values to '0' to avoid confusion.

5.1.2 Record Duplication

No duplicate records are found in the overall dataset. Although 25 addresses appear twice, the entire records are not duplicates.

5.2 Data Integration

A single dataset is used.

5.3 Data Transformation

It will be performed before data training.

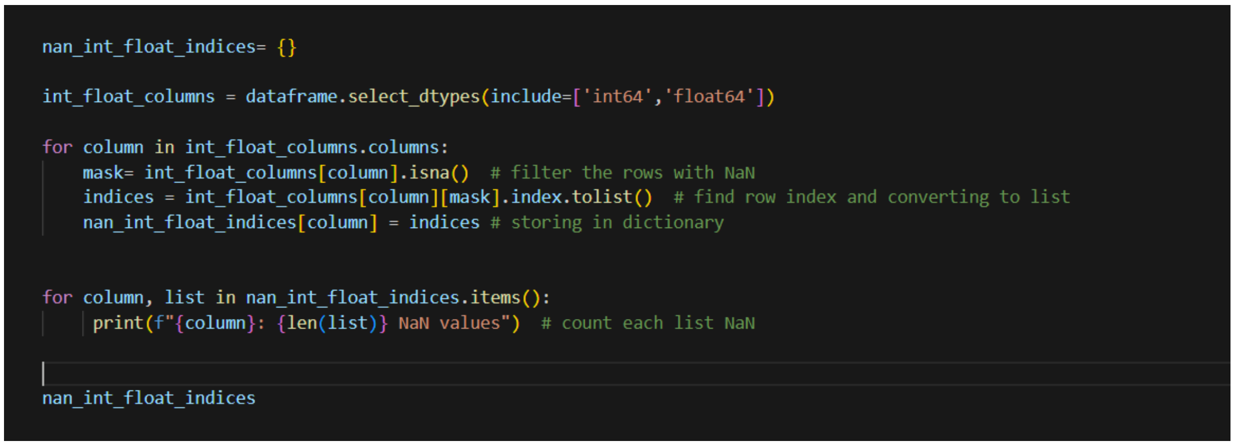
5.4 Data Reduction

Some records may be removed during the model training process.

6 Code Snippets

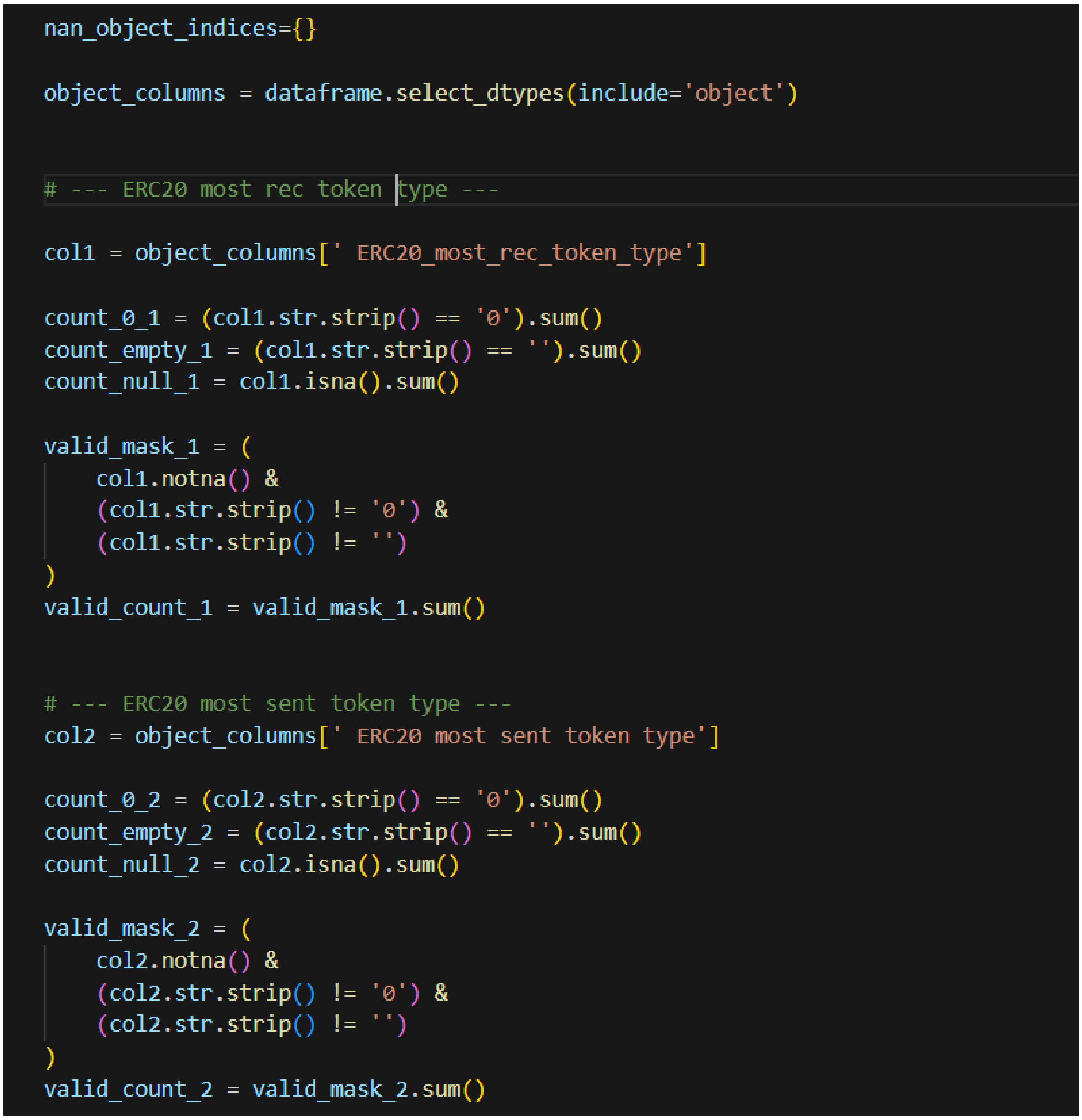
6.1 Missing Values

6.1.1 Int and Float Missing Values



```
nan_int_float_indices= {}

int_float_columns = dataframe.select_dtypes(include=['int64','float64'])

for column in int_float_columns.columns:
    mask= int_float_columns[column].isna()  # filter the rows with NaN
    indices = int_float_columns[column][mask].index.tolist()  # find row index and converting to list
    nan_int_float_indices[column] = indices # storing in dictionary


for column, list in nan_int_float_indices.items():
     print(f"{column}: {len(list)} NaN values")  # count each list NaN


nan_int_float_indices
```

6.1.2 Object Missing Value



```python
nan_object_indices={}

object_columns = dataframe.select_dtypes(include='object')


# --- ERC20 most rec token type ---

col1 = object_columns[' ERC20_most_rec_token_type']

count_0_1 = (col1.str.strip() == '0').sum()
count_empty_1 = (col1.str.strip() == '').sum()
count_null_1 = col1.isna().sum()

valid_mask_1 = (
    col1.notna() &
    (col1.str.strip() != '0') &
    (col1.str.strip() != '')
)
valid_count_1 = valid_mask_1.sum()


# --- ERC20 most sent token type ---
col2 = object_columns[' ERC20 most sent token type']

count_0_2 = (col2.str.strip() == '0').sum()
count_empty_2 = (col2.str.strip() == '').sum()
count_null_2 = col2.isna().sum()

valid_mask_2 = (
    col2.notna() &
    (col2.str.strip() != '0') &
    (col2.str.strip() != '')
)
valid_count_2 = valid_mask_2.sum()
```

```
# --- Address ---
col_addr = object_columns['Address']

count_0_addr = (col_addr.str.strip() == '0').sum()
count_empty_addr = (col_addr.str.strip() == '').sum()

count_null_addr = col_addr.isna().sum()

valid_mask_addr = (
    col_addr.notna() &
    (col_addr.str.strip() != '0') &
    (col_addr.str.strip() != '')
)
valid_count_addr = valid_mask_addr.sum()
```

6.1.3 The ‘FLAG’ column

The FLAG column is used to indicate whether a transaction is fraudulent or not. Therefore, checking for null values in this column is important to ensure accurate labeling.

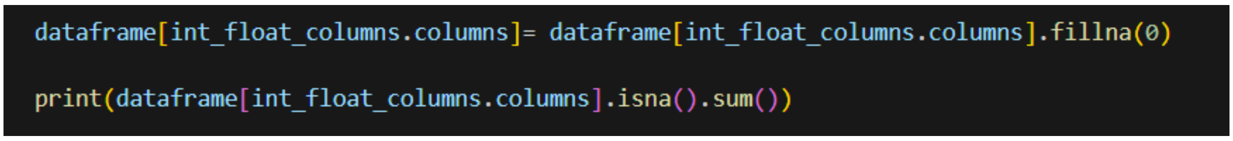
```
flag_column = dataframe['FLAG']
print(flag_column.isna().sum()) # 0
```

6.1.4 Checking Duplicate Values in the entire dataset

```
duplicate = dataframe.duplicated()
duplicate.sum()

# no duplicate record found
```

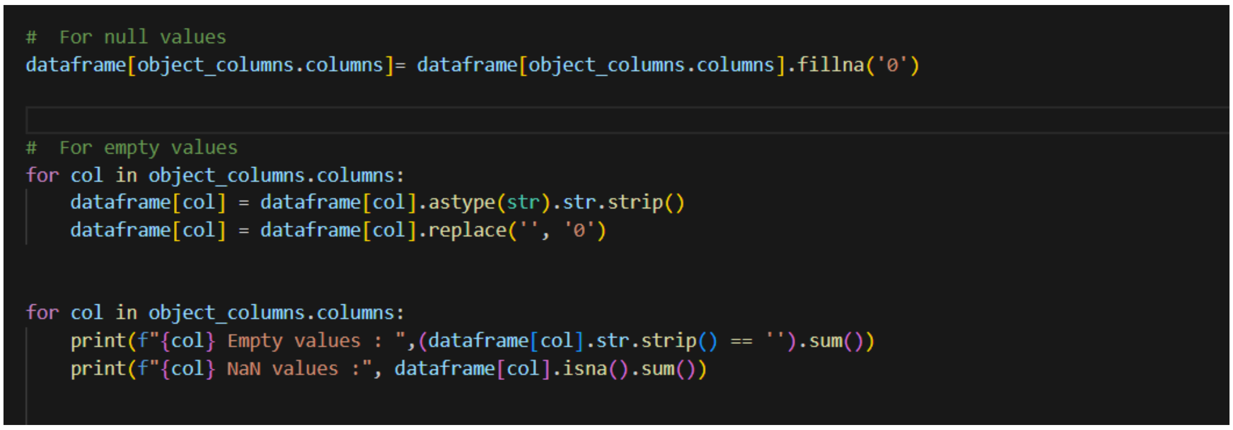
6.2 Handling Missing Values

6.2.1 Int and Float Values



```
dataframe[int_float_columns.columns]= dataframe[int_float_columns.columns].fillna(0)

print(dataframe[int_float_columns.columns].isna().sum())
```

6.2.2 Object Values



```
#  For null values
dataframe[object_columns.columns]= dataframe[object_columns.columns].fillna('0')


#  For empty values
for col in object_columns.columns:
    dataframe[col] = dataframe[col].astype(str).str.strip()
    dataframe[col] = dataframe[col].replace('', '0')


for col in object_columns.columns:
    print(f"{col} Empty values : ",(dataframe[col].str.strip() == '').sum())
    print(f"{col} NaN values :", dataframe[col].isna().sum())
```

7 Observations and Insights

7.1 Data Quality

- No duplicate records found.
- Some missing or empty values were identified in object and numeric columns, which were replaced with 0 for consistency.
- Extra whitespace and inconsistent string formatting cleaned (e.g., in token type columns).

7.2 Data Consistency

- Columns such as “ERC20 most rec token type” and “ERC20 most sent token type” contained unknown or missing token names represented by '0'.
- Numeric columns with NaN values were filled with zeros to maintain model compatibility.

7.3 Address Analysis

- Although 25 addresses appeared more than once, their overall records were unique (no full-row duplicates).

7.4 Flag Column

- The Flag column indicates whether a transaction is fraudulent or not.
- No missing values found; this is crucial since it’s the target variable for classification.

7.5 General Data Characteristics

- The dataset contains both numeric and categorical variables.
- Distribution and imbalance (if any) will need to be checked before model training

8 Conclusion

This Assignment aimed to explore how data science techniques can enhance the detection of fraudulent Ethereum addresses by analyzing transactional data. Through a systematic preprocessing approach, the dataset was cleaned, standardized, and prepared for modeling. Missing numeric values were replaced with zeros to accurately represent the ERC20 transactions state, while categorical inconsistencies were unified by converting unknown or null entries to a standard format. No duplicate records were found, ensuring the integrity and reliability of the data.

Overall, the preprocessing phase established a strong foundation for further model training and fraud classification. The cleaned dataset can now be used to build predictive models capable of identifying suspicious activity and assessing risk levels across blockchain addresses. In the future, machine learning algorithms can be applied to enhance the accuracy, efficiency, and automation of blockchain fraud detection, significantly reducing manual investigation efforts and improving trust in decentralized systems.

9 References

1. https://www.communicationstoday.co.in/over-1-2-billion-lost-in-crypto-hacks-most-targetted-blockchains-of-2024-revealed
2. https://www.bbc.com/news/technology-41011658
3. https://www.datacamp.com/blog/data-preprocessing
4. https://doi.org/10.1007/s41870-022-00864-6