Department of Statistics & Computer Science,

University of Kelaniya

ACADEMIC YEAR – 2023

Master of Science in Computer Science

Fraud Detection in Financial Transaction

Name: B.H.Medini Lakmali

Index No: MSc-CS-2022-033

Contents

1 Introduction

Financial transactions occur on a vast scale daily, spanning various industries and involving numerous parties. Unfortunately, among these transactions, there exists the threat of fraudulent activities. Fraud in financial transactions refers to any intentional deception or manipulation carried out to gain an unfair or unlawful advantage, causing financial loss to individuals, organizations, or financial institutions.

Fraudsters continuously evolve their techniques, making it challenging to detect fraudulent behavior using traditional methods alone. This is where the application of machine learning (ML) and artificial intelligence (AI) becomes crucial in mitigating the risks associated with fraudulent activities.

1.1 Problem Definition

The problem of Fraud Detection in Financial Transactions can be defined as the task of identifying and preventing fraudulent activities within a vast volume of transactions. This involves creating predictive models or algorithms capable of distinguishing between genuine and fraudulent transactions in real-time or near real-time.

1.2 Objects and Goal

To Achieve High Prediction Accuracy

Develop models that accurately classify transactions, minimizing both false negatives (missed fraud) and false positives (legitimate transactions misclassified as fraud).

To Create Scalable Solutions

Build models and systems that can handle large volumes of transactions efficiently, maintaining high accuracy in fraud detection as the volume of transactions increases.

To Adapt to Evolving Fraud Tactics

Develop models that continuously learn and adapt to new fraud patterns and behaviors, staying effective against emerging fraudulent activities.

1.3 Data Resource

Source: Kaggle (https://www.kaggle.com/)

Description: The dataset is a synthetic representation generated by the PaySim simulator, mimicking mobile money transactions based on a sample of real transactions extracted from one month of financial logs from a mobile money service in an African country. It is scaled down to 1/4 of the original dataset and is specifically created for the Kaggle platform. The dataset is designed to facilitate research in fraud detection within the mobile money transactions domain.

```
[ ] from google.colab import drive
    drive.mount('/content/drive')

    Mounted at /content/drive
```

```
[ ] import pandas as pd
    df = pd.read_csv("/content/drive/MyDrive/Colab_Notebooks/archive (6).zip")
```

```
[ ] df.shape

    (6362620, 11)
```

Features:

step: A unit of time in the simulation, where 1 step equals 1 hour (total steps: 744, simulating 30 days).

type: The type of transaction (CASH-IN, CASH-OUT, DEBIT, PAYMENT, TRANSFER).

amount: The transaction amount in local currency.

nameOrig: The customer initiating the transaction.

oldbalanceOrg: The initial balance before the transaction for the initiating customer.

newbalanceOrig: The new balance after the transaction for the initiating customer.

nameDest: The recipient of the transaction.

oldbalanceDest: The initial balance before the transaction for the.

newbalanceDest: The new balance after the transaction for the recipient (.

isFraud: Indicates transactions made by fraudulent agents within the simulation.

isFlaggedFraud: Flags illegal attempts, such as transferring more than 200,000 in a single transaction, in the simulation.

```
[ ] df.columns

Index(['step', 'type', 'amount', 'nameOrig', 'oldbalanceOrg', 'newbalanceOrig',
       'nameDest', 'oldbalanceDest', 'newbalanceDest', 'isFraud',
       'isFlaggedFraud'],
      dtype='object')
```

```
[ ] df.head(5)
```

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

Data Format: CSV (Comma-Separated Values), with 1 row representing a transaction and columns representing different attributes of the transaction.

```
df
```

	step	type	amount	nameOrig	oldbalanceOrg	newbalanceOrig	nameDest	oldbalanceDest	newbalanceDest	isFraud	isFlaggedFraud
0	1	PAYMENT	9839.64	C1231006815	170136.00	160296.36	M1979787155	0.00	0.00	0	0
1	1	PAYMENT	1864.28	C1666544295	21249.00	19384.72	M2044282225	0.00	0.00	0	0
2	1	TRANSFER	181.00	C1305486145	181.00	0.00	C553264065	0.00	0.00	1	0
3	1	CASH_OUT	181.00	C840083671	181.00	0.00	C38997010	21182.00	0.00	1	0
4	1	PAYMENT	11668.14	C2048537720	41554.00	29885.86	M1230701703	0.00	0.00	0	0
...	...	...	...	...	...	...	...	...	...	...	...
6362615	743	CASH_OUT	339682.13	C786484425	339682.13	0.00	C776919290	0.00	339682.13	1	0
6362616	743	TRANSFER	6311409.28	C1529008245	6311409.28	0.00	C1881841831	0.00	0.00	1	0
6362617	743	CASH_OUT	6311409.28	C1162922333	6311409.28	0.00	C1365125890	68488.84	6379898.11	1	0
6362618	743	TRANSFER	850002.52	C1685995037	850002.52	0.00	C2080388513	0.00	0.00	1	0
6362619	743	CASH_OUT	850002.52	C1280323807	850002.52	0.00	C873221189	6510099.11	7360101.63	1	0

6362620 rows × 11 columns

1.4 Limitation

Data Imbalance: The imbalance between fraudulent and non-fraudulent transactions might affect model training, potentially causing challenges in effectively learning.

Lack of Variability: Due to the nature of a synthetic dataset, there might be limited variability in the transactions, possibly leading to biases or oversimplified representations of transaction patterns.

Limited Contextual Information: The dataset might lack additional contextual information or metadata which could limit the depth of analysis and understanding of fraudulent behavior.

2 Methodology

2.1 Data Preparation

The choice of the isFraud column as the target class for fraud detection in this dataset stems from its direct relevance to the primary objective of identifying fraudulent transactions accurately. In this simulated dataset, the isFraud column explicitly marks transactions conducted by fraudulent agents, providing a clear distinction between fraudulent and non-fraudulent activities. This distinct labeling enables the development of machine learning models aimed at effectively distinguishing between genuine and fraudulent transactions.

Class 0 (Non-Fraudulent Transactions): There are a total of 6,354,407 instances labeled as non-fraudulent transactions (0).

```
df1 = df.copy()
df1[df1['isFraud'] == 0]
```

	step	type	amount	nameOrig	oldbalanceOrg	newbalanceOrig	nameDest	oldbalanceDest	newbalanceDest	isFraud	isFlaggedFraud
0	1	PAYMENT	9839.64	C1231006815	170136.0	160296.36	M1979787155	0.00	0.00	0	0
1	1	PAYMENT	1864.28	C1666544295	21249.0	19384.72	M2044282225	0.00	0.00	0	0
4	1	PAYMENT	11668.14	C2048537720	41554.0	29885.86	M1230701703	0.00	0.00	0	0
5	1	PAYMENT	7817.71	C90045638	53860.0	46042.29	M573487274	0.00	0.00	0	0
6	1	PAYMENT	7107.77	C154988899	183195.0	176087.23	M408069119	0.00	0.00	0	0
...	...	...	...	...	...	...	...	...	...	...	...
6362319	718	PAYMENT	8634.29	C642813806	518802.0	510167.71	M747723689	0.00	0.00	0	0
6362320	718	CASH_OUT	159188.22	C691808084	3859.0	0.00	C1818183087	0.00	159188.22	0	0
6362321	718	CASH_OUT	186273.84	C102120699	168046.0	0.00	C1515639522	24893.67	211167.51	0	0
6362322	718	TRANSFER	82096.45	C614459560	13492.0	0.00	C855350324	0.00	82096.45	0	0
6362323	718	DEBIT	1864.24	C49652609	20426.0	18561.76	C1799009964	188746.00	190610.24	0	0

6354407 rows × 11 columns

Class 1 (Fraudulent Transactions): There are a total of 8,213 instances labeled as fraudulent transactions (1).

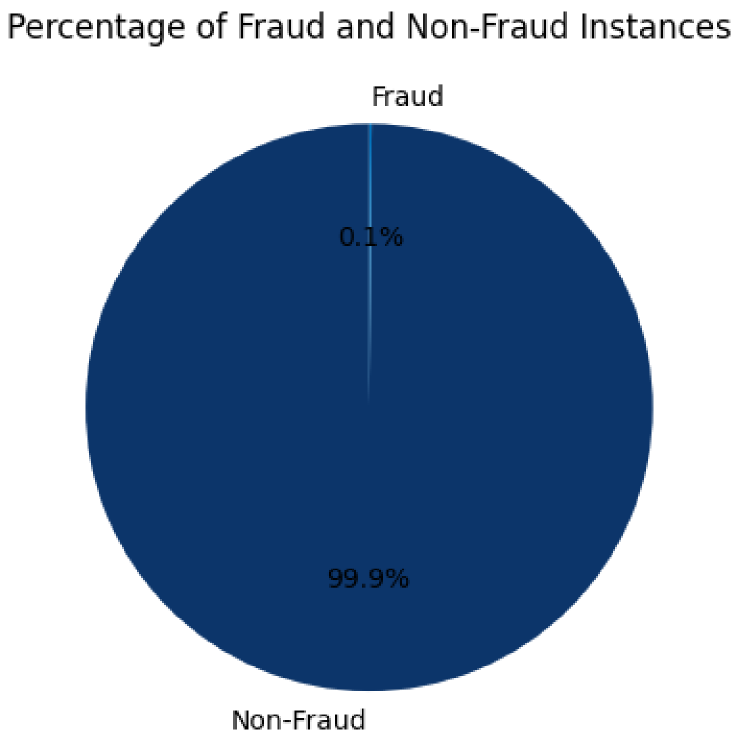
```
[8] df1[df1['isFraud']==1]
```

	step	type	amount	nameOrig	oldbalanceOrg	newbalanceOrig	nameDest	oldbalanceDest	newbalanceDest	isFraud	isFlaggedFraud
2	1	TRANSFER	181.00	C1305486145	181.00	0.0	C553264065	0.00	0.00	1	0
3	1	CASH_OUT	181.00	C840083671	181.00	0.0	C38997010	21182.00	0.00	1	0
251	1	TRANSFER	2806.00	C1420196421	2806.00	0.0	C972765878	0.00	0.00	1	0
252	1	CASH_OUT	2806.00	C2101527076	2806.00	0.0	C1007251739	26202.00	0.00	1	0
680	1	TRANSFER	20128.00	C137533655	20128.00	0.0	C1848415041	0.00	0.00	1	0
...	...	...	...	...	...	...	...	...	...	...	...
6362615	743	CASH_OUT	339682.13	C786484425	339682.13	0.0	C776919290	0.00	339682.13	1	0
6362616	743	TRANSFER	6311409.28	C1529008245	6311409.28	0.0	C1881841831	0.00	0.00	1	0
6362617	743	CASH_OUT	6311409.28	C1162922333	6311409.28	0.0	C1365125890	68488.84	6379898.11	1	0
6362618	743	TRANSFER	850002.52	C1685995037	850002.52	0.0	C2080388513	0.00	0.00	1	0
6362619	743	CASH_OUT	850002.52	C1280323807	850002.52	0.0	C873221189	6510099.11	7360101.63	1	0

8213 rows × 11 columns

This dataset is considered unbalanced due to the significant disparity in the number of instances between the two classes. The vast majority of transactions (represented by Class 0) are non-fraudulent, greatly outnumbering the instances of fraudulent transactions (Class 1). In this case, the ratio between non-fraudulent and fraudulent transactions is heavily skewed towards non-fraudulent instances.


Number of Fraud and Non-Fraud Instances
1e6
6354407
8213
Count
Fraud Status
0
1
2
3
4
5
6


Percentage of Fraud and Non-Fraud Instances
Fraud
0.1%
99.9%
Non-Fraud

2.1.1 Handle Imbalanced Dataset

Approach to Handle Imbalanced Data Set

- Resampling (Uppersampling and Downsampling)
- Synthetic Minority Oversampling Technique
- BalancedBaggingClassifier

Downsampling

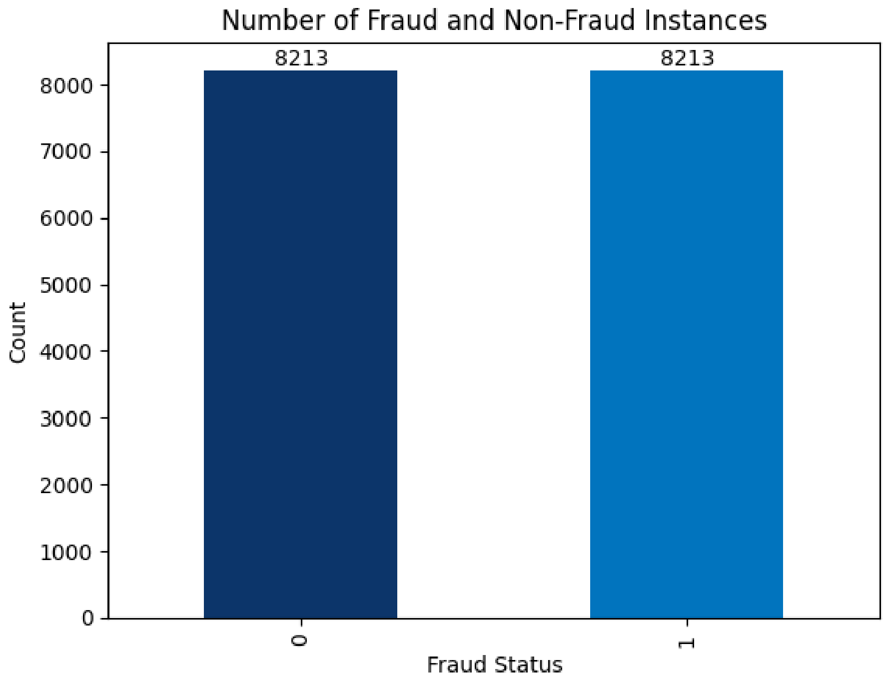
This technique is used to downsample the majority class. When we are using an imbalanced dataset, we can randomly delete rows from the majority class to match them with the minority class which is called undersampling. After sampling the data we can get a balanced dataset for both majority and minority classes. So, when both classes have a similar number of records present in the dataset, we can assume that the classifier will give equal importance to both classes.

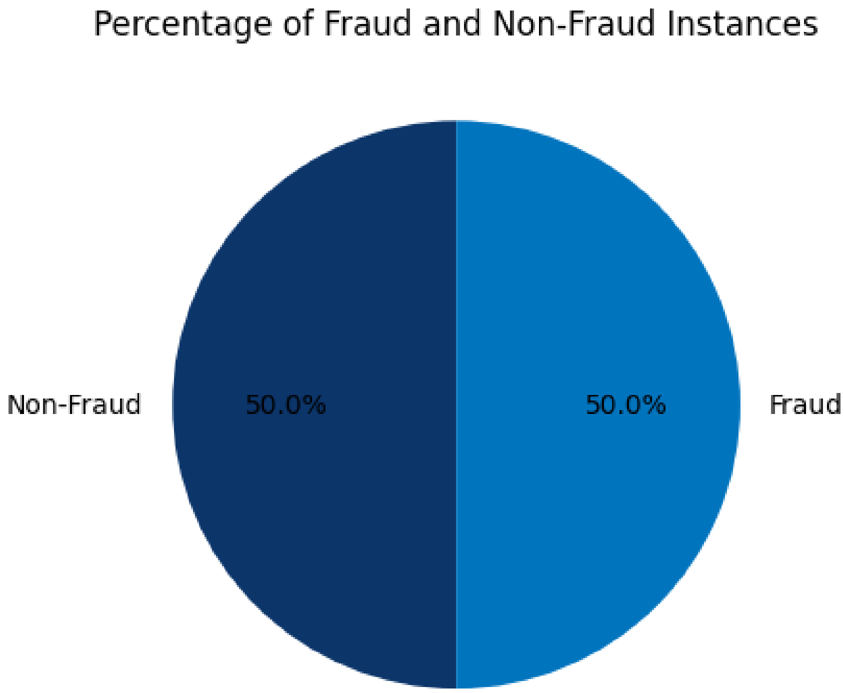
Below technique using the sklearn library's **resample()** is shown below for illustration purposes.

```
from sklearn.utils import resample
#create two different dataframe of majority and minority class
df_majority = df2[(df2['isFraud']==0)]
df_minority = df2[(df2['isFraud']==1)]
# upsample minority class
df_majority_downsampled = resample(df_majority,
                                 replace=False,    # sample with replacement
                                 n_samples= 8213, # to match majority class
                                 random_state=42)  # reproducible results
# Combine majority class with upsampled minority class
df_downsampled = pd.concat([df_majority_downsampled, df_minority])
```

```
df_downsampled['isFraud'].value_counts()
```

```
0    8213
1    8213
Name: isFraud, dtype: int64
```

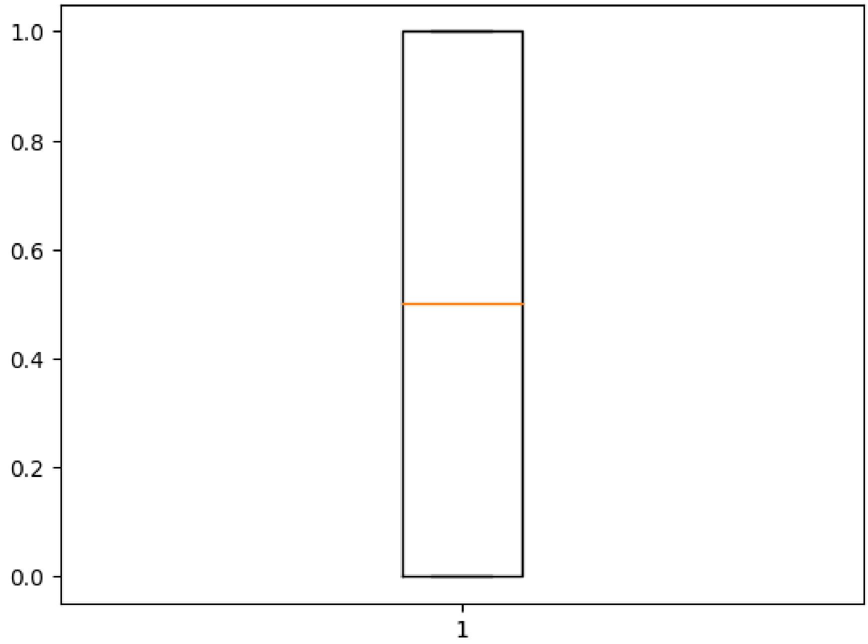

Number of Fraud and Non-Fraud Instances
8213
8213
8000
7000
6000
5000
4000
3000
2000
1000
0
Count
0
1
Fraud Status


Percentage of Fraud and Non-Fraud Instances
Non-Fraud
50.0%
50.0%
Fraud

2.1.2 Data Cleaning

2.1.2.1 Handling Outliers

The absence of outliers in a dataset contributes to a more stable, predictable, and reliable dataset for statistical analysis or modeling.



2.1.2.2 Missing Values Handling

There are no missing values detected across any of the columns in the dataset

```
clean_data.isna().sum()
```

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

2.1.3 Data Preprocessing

2.1.3.1 Data Transform

In the provided dataset, several columns initially held categorical or textual information represented as object data types. These columns, such as **type**, **nameOrig**, and **nameDest**, contained categorical information that needed to be numerically encoded for machine learning algorithms to process effectively.

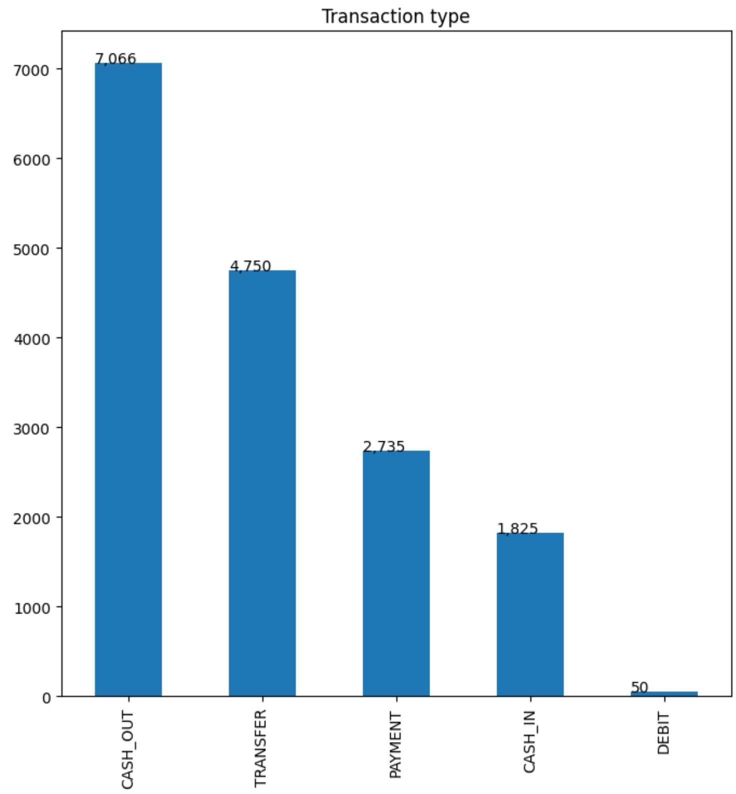
```
clean_data.dtypes
```

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

The 'type' column in the dataset contains categorical information representing different types of transactions. Each transaction type signifies a distinct category of financial activities.

```
[29] clean_data['type'].value_counts()
```

```
CASH_OUT    7066
TRANSFER    4750
PAYMENT     2735
CASH_IN     1825
DEBIT         50
Name: type, dtype: int64
```



The **LabelEncoder** from the **scikit-learn** library was utilized to transform these categorical columns into a format suitable for machine learning models. The **LabelEncoder** assigned a unique numerical label to each unique category or value within the categorical columns.

```
from sklearn.preprocessing import LabelEncoder
my_encoder = LabelEncoder()
df_cln['type'] = my_encoder.fit_transform(df_cln['type'])
#0 for CASH_IN 1 for CASH_OUT 2 for DEBIT 3 for PAYMENT 4 for TRANSFER.
```

```
df_cln.dtypes
```

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

'nameOrig' trnsform in to integer

```
clean_data['nameOrig'].value_counts()
```

```
C691771226      1
C1263272342     1
C1843566745     1
C351713185      1
C58682758       1
               ..
C944685644      1
C736006600      1
C1294515646     1
C73296501       1
C1280323807     1
Name: nameOrig, Length: 16426, dtype: int64
```

```
[37] from sklearn.preprocessing import LabelEncoder
     my_encoder = LabelEncoder()
     df_cln['nameOrig'] = my_encoder.fit_transform(df_cln['nameOrig'])
```

```
df_cln.dtypes
```

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

'nameDest' transform into integer value

```
clean_data['nameDest'].value_counts()
```

```
C1561140816    3
C2020337583    3
C164033249     3
C1875540277    3
C330226144     2
              ..
M1792844041    1
C1299718574    1
C312306105     1
M523692779     1
C2080388513    1
Name: nameDest, Length: 16235, dtype: int64
```

```
from sklearn.preprocessing import LabelEncoder
my_encoder = LabelEncoder()
df_cln['nameDest'] = my_encoder.fit_transform(df_cln['nameDest'])
```

```
[40] df_cln.dtypes
```

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

After the transformation, the categorical columns such as **type**, **nameOrig**, and **nameDest** were converted into integer representations (int64 data type). Each distinct category within these columns was assigned a corresponding unique numerical label

```
df_cln
```

	step	type	amount	nameOrig	oldbalanceOrg	newbalanceOrig	nameDest	oldbalanceDest	newbalanceDest	isFraud	isFlaggedFraud
1777056	162	1	183806.32	13825	19391.00	0.00	2838	382572.19	566378.51	0	0
1350600	137	3	521.37	8934	0.00	0.00	15435	0.00	0.00	0	0
1991933	179	3	3478.18	6065	19853.00	16374.82	15741	0.00	0.00	0	0
5092368	355	3	1716.05	15672	5769.17	4053.13	14056	0.00	0.00	0	0
5066515	354	0	253129.93	8798	1328499.49	1581629.42	9322	2713220.48	2460090.55	0	0
...	...	...	...	...	...	...	...	...	...	...	...
6362615	743	1	339682.13	14602	339682.13	0.00	11940	0.00	339682.13	1	0
6362616	743	4	6311409.28	4586	6311409.28	0.00	6158	0.00	0.00	1	0
6362617	743	1	6311409.28	1423	6311409.28	0.00	2453	68488.84	6379898.11	1	0
6362618	743	4	850002.52	5974	850002.52	0.00	7510	0.00	0.00	1	0
6362619	743	1	850002.52	2449	850002.52	0.00	12628	6510099.11	7360101.63	1	0

16426 rows × 11 columns

2.1.3.2 Data Reduction

In the dataset used for fraud detection, certain columns might not contribute significantly to the detection of fraudulent activities or might contain information that is irrelevant to the model.

```
Correlation Matrix:
                   step      type    amount  nameOrig  oldbalanceOrg  \
step           1.000000  0.097230  0.149111 -0.006061       0.074185
type           0.097230  1.000000  0.112007 -0.000257      -0.071468
amount         0.149111  0.112007  1.000000  0.017550       0.646056
nameOrig      -0.006061 -0.000257  0.017550  1.000000       0.014088
oldbalanceOrg  0.074185 -0.071468  0.646056  0.014088       1.000000
newbalanceOrig -0.022918 -0.183056  0.123465  0.006553       0.824217
nameDest      -0.104483  0.185777 -0.114270 -0.009775      -0.095401
oldbalanceDest -0.006124 -0.132000  0.005105  0.015011       0.008734
newbalanceDest  0.027531 -0.176528  0.256852  0.019683       0.117984
isFraud        0.320576  0.273082  0.345287  0.002854       0.125072
isFlaggedFraud  0.037332  0.040256  0.067676  0.013601       0.063118

                newbalanceOrig  nameDest  oldbalanceDest  newbalanceDest  \
step                 -0.022918 -0.104483       -0.006124        0.027531
type                 -0.183056  0.185777       -0.132000       -0.176528
amount                0.123465 -0.114270        0.005105        0.256852
nameOrig              0.006553 -0.009775        0.015011        0.019683
oldbalanceOrg         0.824217 -0.095401        0.008734        0.117984
newbalanceOrig        1.000000 -0.051151        0.039598        0.005994
nameDest             -0.051151  1.000000       -0.066738       -0.090406
oldbalanceDest        0.039598 -0.066738        1.000000        0.928051
newbalanceDest        0.005994 -0.090406        0.928051        1.000000
isFraud              -0.133095 -0.282582       -0.082357        0.004891
isFlaggedFraud        0.090899 -0.020221       -0.007449       -0.010187

                 isFraud  isFlaggedFraud
step            0.320576        0.037332
type            0.273082        0.040256
amount          0.345287        0.067676
nameOrig        0.002854        0.013601
oldbalanceOrg   0.125072        0.063118
newbalanceOrig -0.133095        0.090899
nameDest       -0.282582       -0.020221
oldbalanceDest -0.082357       -0.007449
newbalanceDest  0.004891       -0.010187
isFraud         1.000000        0.031225
isFlaggedFraud  0.031225        1.000000
```

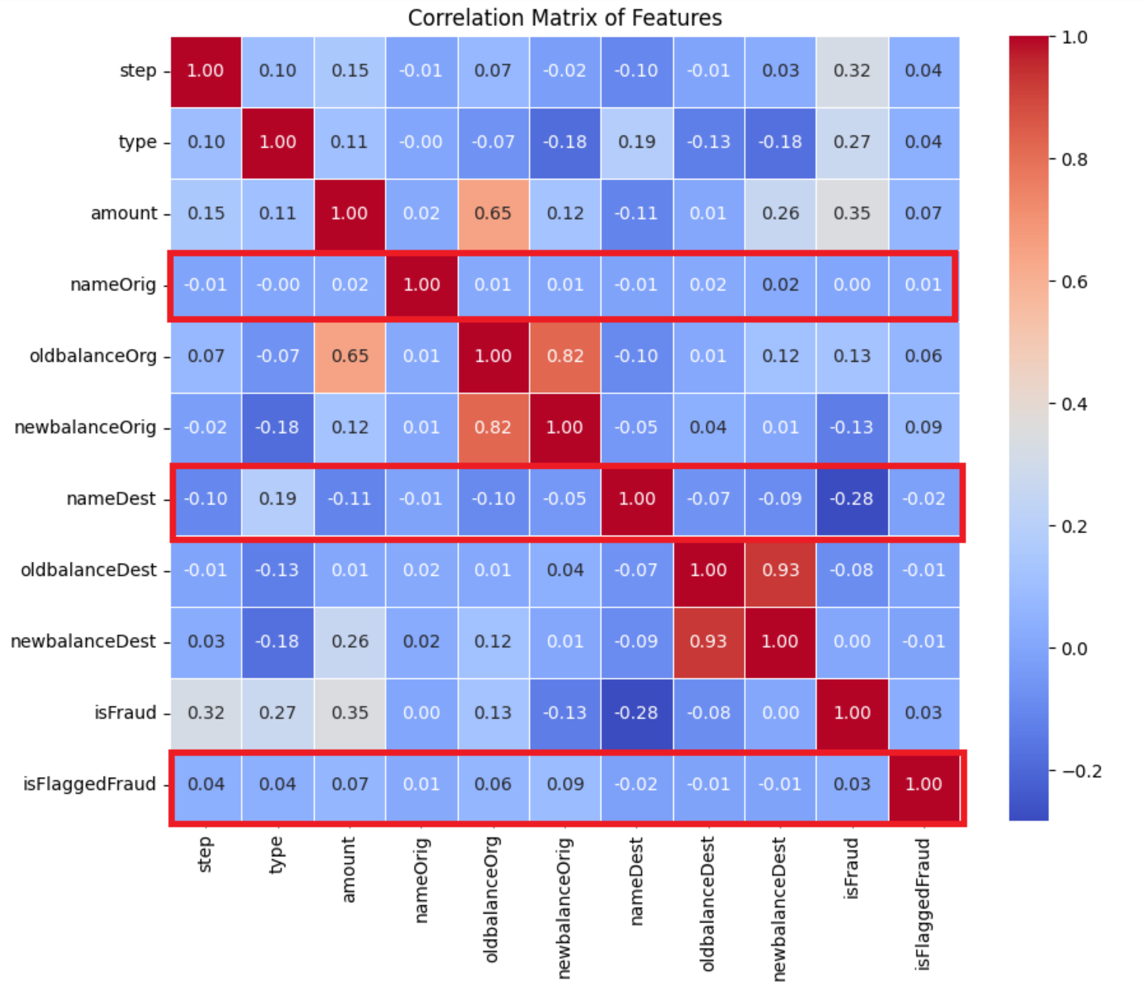
The correlation matrix provides insights into the relationships between different features in the dataset, measuring the strength and direction of their linear associations.

nameOrig and **nameDest**:

Both 'nameOrig' and 'nameDest' exhibit relatively low absolute correlation values (ranging from -0.28 to 0.02) with the target variable 'isFraud.'

isFlaggedFraud:

Correlation Explanation: The 'isFlaggedFraud' column shows a relatively low correlation value (0.03) with the 'isFraud' target variable.



The decision to remove these columns ('nameOrig,' 'nameDest,' and 'isFlaggedFraud') could be considered based on their weaker linear correlations with the target variable 'isFraud.'

```
[45] df_new_reduction
```

	step	type	amount	oldbalanceOrg	newbalanceOrig	oldbalanceDest	newbalanceDest	isFraud
1777056	162	1	183806.32	19391.00	0.00	382572.19	566378.51	0
1350600	137	3	521.37	0.00	0.00	0.00	0.00	0
1991933	179	3	3478.18	19853.00	16374.82	0.00	0.00	0
5092368	355	3	1716.05	5769.17	4053.13	0.00	0.00	0
5066515	354	0	253129.93	1328499.49	1581629.42	2713220.48	2460090.55	0
...	...	...	...	...	...	...	...	...
6362615	743	1	339682.13	339682.13	0.00	0.00	339682.13	1
6362616	743	4	6311409.28	6311409.28	0.00	0.00	0.00	1
6362617	743	1	6311409.28	6311409.28	0.00	68488.84	6379898.11	1
6362618	743	4	850002.52	850002.52	0.00	0.00	0.00	1
6362619	743	1	850002.52	850002.52	0.00	6510099.11	7360101.63	1

16426 rows × 8 columns

2.2 Model Planning

The selection of supervised learning classification for fraud detection stems from its reliance on labeled data, enabling models to distinguish between known classes like fraudulent and non-fraudulent transactions. This approach suits the dataset, which contains labeled instances of fraud ('isFraud') used for model training. Employing Decision Tree, Random Forest, and Naive Bayes, these ML models leverage the labeled data to discern patterns in financial transaction features. The Decision Tree offers transparent decision-making, Random Forest combines multiple trees for better accuracy, while Naive Bayes, despite its simplicity, provides an efficient probabilistic approach. Trained on labeled data, these models aim to accurately categorize transactions, enhancing fraud detection and prevention in financial systems.

3 Implementation

3.1 Model Building

3.1.1.1 Decision Tree

This code segment performs a crucial step in machine learning—dividing the dataset into subsets: a training set used to train the model (70% of the data) and a test set employed to evaluate its performance (30% of the data). These subsets, x_train, x_test, y_train, and y_test, are crucial for training the model on one portion of the data and assessing its predictive ability on unseen data from the test set.

In this specific instance, the model achieves an accuracy score of approximately 98.97%, showcasing its ability to effectively identify fraudulent transactions within the given test data

```
[46] #load libraries
     import pandas as pd
     from sklearn.tree import DecisionTreeClassifier # Import Decision Tree Classification
     from sklearn.model_selection import train_test_split # Import train_test_split
     from sklearn import metrics #import scikit-learn metrics module for accuracy
```

```
#split dataset in features and target variable
feature_cols = ['step', 'type', 'amount', 'oldbalanceOrg', 'newbalanceOrig',
       'oldbalanceDest', 'newbalanceDest', 'isFraud']
x = df_prepared.drop(columns=['isFraud']) # Features
y = df_prepared.isFraud # Target variable
```

```
#Split dataset into training set and test set
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.3, random_state=42)
```

```
x_train.shape
```

```
(11498, 7)
```

```
[53] x_test.shape
```

```
(4928, 7)
```

```
#Create Decision Tree classifer object
clf = DecisionTreeClassifier(random_state=42)

#Train Decision Tree Classifer
clf = clf.fit(x_train,y_train)
```

```
#Predict the response for test dataset
y_pred = clf.predict(x_test)
```

```
#Model Accuracy, how often is the classifier correct
print("Accuracy:",metrics.accuracy_score(y_test, y_pred))
```

```
Accuracy: 0.989650974025974
```

3.1.1.2 Random Forest Tree

A RandomForestClassifier object is instantiated and trained using the training data, with 70% allocated for training and 30% for testing.The model is then employed to predict instances of fraud in the test dataset. Finally, the model's accuracy is assessed by comparing its predictions against the actual labels, providing an accuracy score of approximately 99.17%.

```
[62] from sklearn.ensemble import RandomForestClassifier
     from sklearn.metrics import accuracy_score
```

```
#split dataset in features and target variable
feature_cols = ['step', 'type', 'amount', 'oldbalanceOrg', 'newbalanceOrig',
       'oldbalanceDest', 'newbalanceDest', 'isFraud']
X = df_prepared1.drop(columns=['isFraud']) # Features
Y = df_prepared1.isFraud # Target variable
```

```
#Split dataset into training set and test set
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.3, random_state=42)
```

```
X_train.shape
```

```
(11498, 7)
```

```
X_test.shape
```

```
(4928, 7)
```

```
#Create Decision Tree classifer object
rfc = RandomForestClassifier(n_estimators=100, random_state=42)

#Train Decision Tree Classifer
rfc = rfc.fit(X_train,Y_train)
```

```
#Predict the response for test dataset
rf_pred = rfc.predict(X_test)
```

```
#Model accuracy how often is the classifier correct?
print("Accuracy:",accuracy_score(Y_test,rf_pred))
```

```
Accuracy: 0.9916801948051948
```

3.1.1.3 Multinomial Naive Bayes

This code initializes and utilizes a Multinomial Naive Bayes classifier from scikit-learn. Initially, the dataset is divided into features and the target variable ('isFraud'). Subsequently, the dataset is split into training and testing subsets, dedicating 70% of the data for training the model and 30% for testing its performance.

the Multinomial Naive Bayes classifier achieves an accuracy score of approximately 74.63%

```
from sklearn.naive_bayes import MultinomialNB
```

```
#split dataset in features and target variable
feature_cols = ['step', 'type', 'amount', 'oldbalanceOrg', 'newbalanceOrig',
       'oldbalanceDest', 'newbalanceDest', 'isFraud']
NB_X = df_prepared2.drop(columns=['isFraud']) # Features
NB_Y = df_prepared2.isFraud # Target variable
```

```
#Split dataset into training set and test set
NB_X_train, NB_X_test, NB_Y_train, NB_Y_test = train_test_split(NB_X, NB_Y, test_size=0.3, random_state=42)
```

```
[78] NB_X_train.shape
```

```
(11498, 7)
```

```
[79] NB_X_test.shape
```

```
(4928, 7)
```

```
#instantiant the model
mnb = MultinomialNB()

#fit the model
mnb.fit(NB_X_train, NB_Y_train)
```

```
▾ MultinomialNB
MultinomialNB()
```

```
NB_Y_pred = mnb.predict(NB_X_test)
```

```
#Model accuracy how often is the classifier correct?
print("Accuracy:",accuracy_score(NB_Y_test,NB_Y_pred))
```

```
Accuracy: 0.7463474025974026
```

4 Conclusion

The performance evaluation of the machine learning models reveals varying levels of accuracy. The Random Forest Tree model demonstrates the highest accuracy of 99.17%, followed closely by the Decision Tree model with an accuracy of 98.97%. In comparison, the Multinomial Naive Bayes model achieves a comparatively lower accuracy of 74.63%.

The Random Forest Tree and Decision Tree models showcase significantly higher accuracy rates compared to the Multinomial Naive Bayes model. Therefore, for this specific fraud detection task, the ensemble-based Random Forest Tree and the individual Decision Tree models exhibit superior performance in accurately classifying fraudulent transactions when compared to the Multinomial Naive Bayes model.