Importing required Python modules

In [22]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import time
import os
```

Ingesting dataset for analysis

In [2]:
```
os.chdir('.')

file_path = "../data/onlinefraud.csv"
try:
    df = pd.read_csv(file_path, encoding='latin1')
except UnicodeDecodeError:
    df = pd.read_csv(file_path, encoding='iso-8859-1')
except UnicodeDecodeError:
    df = pd.read_csv(file_path, encoding='cp1252')

df.head()
```

Out[2]:

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

Observing shape of the dataset

In [55]:
```
print(f"This dataset has {df.shape[0]} rows")
print(f"This dataset has {df.shape[1]} columns")
```

```
This dataset has 6362620 rows
This dataset has 11 columns
```

Observing data types

In [56]:
```
pd.DataFrame(df.dtypes, columns=["DataType"])
```

Out[56]:

	DataType
step	int64
type	object
amount	float64
nameOrig	object
oldbalanceOrg	float64
newbalanceOrig	float64
nameDest	object
oldbalanceDest	float64
newbalanceDest	float64
isFraud	int64
isFlaggedFraud	int64

1. Determining differences between the rows with isFlaggedFraud=0 and isFlaggedFraud=1

1A. Getting counts of rows for each case

In [34]:
```
print(f"The number of rows in the dataset having isFlaggedFraud=0 are {df['isFlagge
print(f"The number of rows in the dataset having isFlaggedFraud=1 are {df['isFlagge
```

```
The number of rows in the dataset having isFlaggedFraud=0 are 6362604
The number of rows in the dataset having isFlaggedFraud=1 are 16
```

1B. Separating out rows based on the column isFlaggedFraud

In [35]:
```
flagged_frauds_df = pd.DataFrame(df[df['isFlaggedFraud']==1])
not_flagged_frauds_df = pd.DataFrame(df[df['isFlaggedFraud']==0])
```

1C. Comparing the minimum & maximum amounts between both cases

In [37]:
```
print(f"The minimum amount in the dataset within the non flagged transactions subse
print(f"The minimum amount in the dataset within the flagged transactions subset is
```

```
The minimum amount in the dataset within the non flagged transactions subset is 0.0
The minimum amount in the dataset within the flagged transactions subset is 353874.2
2
```

In [38]:
```
print(f"The maximum amount in the dataset within the non flagged transactions subse
print(f"The maximum amount in the dataset within the flagged transactions subset is
```

```
The maximum amount in the dataset within the non flagged transactions subset is 9244
5516.64
The maximum amount in the dataset within the flagged transactions subset is 1000000
0.0
```

1D. What is the relationship between isFlaggedFraud and isFraud column ?

In [39]:
```
flagged_frauds_df
```

Out[39]:

	step	type	amount	nameOrig	oldbalanceOrg	newbalanceOrig	n
2736446	212	TRANSFER	4953893.08	C728984460	4953893.08	4953893.08	C6
3247297	250	TRANSFER	1343002.08	C1100582606	1343002.08	1343002.08	C11
3760288	279	TRANSFER	536624.41	C1035541766	536624.41	536624.41	C11
5563713	387	TRANSFER	4892193.09	C908544136	4892193.09	4892193.09	C8
5996407	425	TRANSFER	10000000.00	C689608084	19585040.37	19585040.37	C13
5996409	425	TRANSFER	9585040.37	C452586515	19585040.37	19585040.37	C11
6168499	554	TRANSFER	3576297.10	C193696150	3576297.10	3576297.10	C4
6205439	586	TRANSFER	353874.22	C1684585475	353874.22	353874.22	C17
6266413	617	TRANSFER	2542664.27	C786455622	2542664.27	2542664.27	C6
6281482	646	TRANSFER	10000000.00	C19004745	10399045.08	10399045.08	C18
6281484	646	TRANSFER	399045.08	C724693370	10399045.08	10399045.08	C19
6296014	671	TRANSFER	3441041.46	C917414431	3441041.46	3441041.46	C10
6351225	702	TRANSFER	3171085.59	C1892216157	3171085.59	3171085.59	C13
6362460	730	TRANSFER	10000000.00	C2140038573	17316255.05	17316255.05	C13
6362462	730	TRANSFER	7316255.05	C1869569059	17316255.05	17316255.05	C18
6362584	741	TRANSFER	5674547.89	C992223106	5674547.89	5674547.89	C13

All 16 flagged transactions were fraud. None of the transactions involved any merchants.

All amounts were higher than 200,000. These transactions were canceled by the system.

The documentation says that these transactions should not be considered for further analytics

Removing the rows with isFlaggedFraud=1 (as recommeded by the dataset's creator)

In [3]:
```
df = df[df['isFlaggedFraud'] !=1]
df
```

Out[3]:

	step	type	amount	nameOrig	oldbalanceOrg	newbalanceOrig	n
0	1	PAYMENT	9839.64	C1231006815	170136.00	160296.36	M19
1	1	PAYMENT	1864.28	C1666544295	21249.00	19384.72	M20
2	1	TRANSFER	181.00	C1305486145	181.00	0.00	C5
3	1	CASH_OUT	181.00	C840083671	181.00	0.00	C
4	1	PAYMENT	11668.14	C2048537720	41554.00	29885.86	M12
...	...	...	...	...	...	...	
6362615	743	CASH_OUT	339682.13	C786484425	339682.13	0.00	C7
6362616	743	TRANSFER	6311409.28	C1529008245	6311409.28	0.00	C18
6362617	743	CASH_OUT	6311409.28	C1162922333	6311409.28	0.00	C13
6362618	743	TRANSFER	850002.52	C1685995037	850002.52	0.00	C20
6362619	743	CASH_OUT	850002.52	C1280323807	850002.52	0.00	C8

6362604 rows × 11 columns

In [4]:
```
not_frauds_df = pd.DataFrame(df[df['isFraud']==0])
frauds_df = pd.DataFrame(df[df['isFraud']==1])
```

In [6]:
```
df['orgCustomerType'] = df['nameOrig'].str[0]
df['destCustomerType'] = df['nameDest'].str[0]
```

In [71]:
```
df['orgCustomerType'].value_counts()
```

Out[71]:
```
orgCustomerType
C    6362604
Name: count, dtype: int64
```

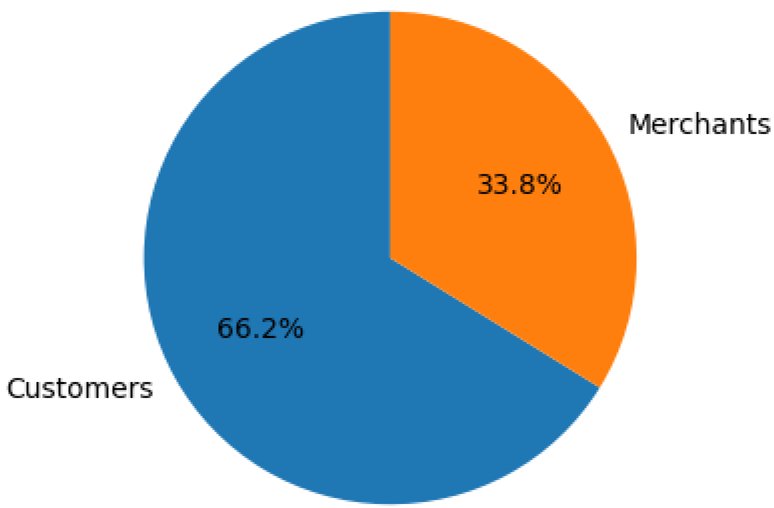
In [7]:
```
df['destCustomerType'].value_counts()
```

Out[7]:
```
destCustomerType
C    4211109
M    2151495
Name: count, dtype: int64
```

In [36]:
```
n_counts_c = df['destCustomerType'].value_counts()[0]
n_counts_m = df['destCustomerType'].value_counts()[1]

# Creating dataset
labels = ['Customers', 'Merchants']
```

```
data = [n_counts_c, n_counts_m]

# Creating plot
fig = plt.figure(figsize=(5, 4))
plt.pie(data, labels=labels, autopct='%1.1f%%', startangle=90)

# show plot
plt.show()
```

```
C:\Users\ketan\AppData\Local\Temp\ipykernel_1872\3859683366.py:1: FutureWarning: Ser
ies.__getitem__ treating keys as positions is deprecated. In a future version, integ
er keys will always be treated as labels (consistent with DataFrame behavior). To ac
cess a value by position, use `ser.iloc[pos]`
  n_counts_c = df['destCustomerType'].value_counts()[0]
C:\Users\ketan\AppData\Local\Temp\ipykernel_1872\3859683366.py:2: FutureWarning: Ser
ies.__getitem__ treating keys as positions is deprecated. In a future version, integ
er keys will always be treated as labels (consistent with DataFrame behavior). To ac
cess a value by position, use `ser.iloc[pos]`
  n_counts_m = df['destCustomerType'].value_counts()[1]
```



In [10]:
```
frauds_df['orgCustomerType'] = frauds_df['nameOrig'].str[0]
frauds_df['destCustomerType'] = frauds_df['nameDest'].str[0]

frauds_df
```

Out[10]:

	step	type	amount	nameOrig	oldbalanceOrg	newbalanceOrig	na
2	1	TRANSFER	181.00	C1305486145	181.00	0.0	C55
3	1	CASH_OUT	181.00	C840083671	181.00	0.0	C3
251	1	TRANSFER	2806.00	C1420196421	2806.00	0.0	C97
252	1	CASH_OUT	2806.00	C2101527076	2806.00	0.0	C100
680	1	TRANSFER	20128.00	C137533655	20128.00	0.0	C184
...	...	...	...	...	...	...	
6362615	743	CASH_OUT	339682.13	C786484425	339682.13	0.0	C77
6362616	743	TRANSFER	6311409.28	C1529008245	6311409.28	0.0	C188
6362617	743	CASH_OUT	6311409.28	C1162922333	6311409.28	0.0	C136
6362618	743	TRANSFER	850002.52	C1685995037	850002.52	0.0	C208
6362619	743	CASH_OUT	850002.52	C1280323807	850002.52	0.0	C87

8197 rows × 13 columns

In [11]:

```
frauds_df['orgCustomerType'].value_counts()
```

Out[11]:

```
orgCustomerType
C    8197
Name: count, dtype: int64
```

In [12]:

```
frauds_df['destCustomerType'].value_counts()
```

Out[12]:

```
destCustomerType
C    8197
Name: count, dtype: int64
```

In [13]:

```
not_frauds_df['orgCustomerType'] = not_frauds_df['nameOrig'].str[0]
not_frauds_df['destCustomerType'] = not_frauds_df['nameDest'].str[0]

not_frauds_df
```

Out[13]:

	step	type	amount	nameOrig	oldbalanceOrg	newbalanceOrig	na
0	1	PAYMENT	9839.64	C1231006815	170136.0	160296.36	M197
1	1	PAYMENT	1864.28	C1666544295	21249.0	19384.72	M204
4	1	PAYMENT	11668.14	C2048537720	41554.0	29885.86	M123
5	1	PAYMENT	7817.71	C90045638	53860.0	46042.29	M57
6	1	PAYMENT	7107.77	C154988899	183195.0	176087.23	M40
...	...	...	...	...	...	...	
6362319	718	PAYMENT	8634.29	C642813806	518802.0	510167.71	M74
6362320	718	CASH_OUT	159188.22	C691808084	3859.0	0.00	C181
6362321	718	CASH_OUT	186273.84	C102120699	168046.0	0.00	C151
6362322	718	TRANSFER	82096.45	C614459560	13492.0	0.00	C85
6362323	718	DEBIT	1864.24	C49652609	20426.0	18561.76	C179

6354407 rows × 13 columns

In [14]:
```
not_frauds_df['orgCustomerType'].value_counts()
```

Out[14]:
```
orgCustomerType
C    6354407
Name: count, dtype: int64
```

In [15]:
```
not_frauds_df['destCustomerType'].value_counts()
```

Out[15]:
```
destCustomerType
C    4202912
M    2151495
Name: count, dtype: int64
```

In [16]:
```
not_frauds_df = not_frauds_df[not_frauds_df['destCustomerType'] != "M"]
not_frauds_df
```

Out[16]:

	step	type	amount	nameOrig	oldbalanceOrg	newbalanceOrig	na
9	1	DEBIT	5337.77	C712410124	41720.0	36382.23	C195
10	1	DEBIT	9644.94	C1900366749	4465.0	0.00	C997
15	1	CASH_OUT	229133.94	C905080434	15325.0	0.00	C476
19	1	TRANSFER	215310.30	C1670993182	705.0	0.00	C1100
21	1	DEBIT	9302.79	C1566511282	11299.0	1996.21	C1973
...	...	...	...	...	...	...	
6362317	718	CASH_OUT	317177.48	C857156502	170.0	0.00	C784
6362320	718	CASH_OUT	159188.22	C691808084	3859.0	0.00	C1818
6362321	718	CASH_OUT	186273.84	C102120699	168046.0	0.00	C1515
6362322	718	TRANSFER	82096.45	C614459560	13492.0	0.00	C855
6362323	718	DEBIT	1864.24	C49652609	20426.0	18561.76	C1799

4202912 rows × 13 columns

In [17]:

```
frauds_df['diff_new_bals'] = frauds_df['newbalanceDest'] - frauds_df['oldbalanceDes
not_frauds_df['diff_new_bals'] = not_frauds_df['newbalanceDest'] - not_frauds_df['o

print(f"The average value of diffs in Non-Fraudulent transactions are : {not_frauds
print(f"The average value of diffs in Fraudulent transactions are : {frauds_df['dif
```

```
The average value of diffs in Non-Fraudulent transactions are : 186727.63768505031
The average value of diffs in Fraudulent transactions are : 736893.5632743686
```

```
C:\Users\ketan\AppData\Local\Temp\ipykernel_1872\1740282389.py:2: SettingWithCopyWar
ning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/u
ser_guide/indexing.html#returning-a-view-versus-a-copy
  not_frauds_df['diff_new_bals'] = not_frauds_df['newbalanceDest'] - not_frauds_df
['oldbalanceDest']
```

In [18]:

```
frauds_df['diff_new_bals'] = frauds_df['newbalanceDest'] - frauds_df['newbalanceOri
not_frauds_df['diff_new_bals'] = not_frauds_df['newbalanceDest'] - not_frauds_df['n

print(f"The average value of diffs in Non-Fraudulent transactions are : {not_frauds
print(f"The average value of diffs in Fraudulent transactions are : {frauds_df['dif
```

```
The average value of diffs in Non-Fraudulent transactions are : 589481.0665288282
The average value of diffs in Fraudulent transactions are : 1104697.313473222
```

```
C:\Users\ketan\AppData\Local\Temp\ipykernel_1872\470572971.py:2: SettingWithCopyWarn
ing:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/u
ser_guide/indexing.html#returning-a-view-versus-a-copy
  not_frauds_df['diff_new_bals'] = not_frauds_df['newbalanceDest'] - not_frauds_df
['newbalanceOrig']
```

In [19]:

```
frauds_df['diff_new_bals'] = frauds_df['newbalanceOrig'] - frauds_df['oldbalanceOrg
not_frauds_df['diff_new_bals'] = not_frauds_df['newbalanceOrig'] - not_frauds_df['o

print(f"The average value of diffs in Non-Fraudulent transactions are : {not_frauds
print(f"The average value of diffs in Fraudulent transactions are : {frauds_df['dif
```

```
The average value of diffs in Non-Fraudulent transactions are : 38253.207304656884
The average value of diffs in Fraudulent transactions are : -1460119.477911431
```

```
C:\Users\ketan\AppData\Local\Temp\ipykernel_1872\632296387.py:2: SettingWithCopyWarn
ing:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/u
ser_guide/indexing.html#returning-a-view-versus-a-copy
  not_frauds_df['diff_new_bals'] = not_frauds_df['newbalanceOrig'] - not_frauds_df
['oldbalanceOrg']
```

In [ ]:

Now observing the shape of updated dataframe

In [20]:

```
print(f"This dataset has {df.shape[0]} rows after removing the FlaggedFraud transac
print(f"This dataset has {df.shape[1]} columns after removing the FlaggedFraud tran
```

```
This dataset has 6362604 rows after removing the FlaggedFraud transactions
This dataset has 13 columns after removing the FlaggedFraud transactions
```

2. Determining differences between the rows with isFraud=0 and isFraud=1

2A. Getting counts of rows for each case

In [81]:

```
print(f"The number of rows in the dataset having isFraud=0 are {df['isFraud'].value
print(f"The number of rows in the dataset having isFraud=1 are {df['isFraud'].value
```

```
The number of rows in the dataset having isFraud=0 are 6354407
The number of rows in the dataset having isFraud=1 are 8197
```

2B. Separating out rows based on the column isFraud

In [82]:
```
not_frauds_df = pd.DataFrame(df[df['isFraud']==0])
frauds_df = pd.DataFrame(df[df['isFraud']==1])
```

2C. Observing the unique values in TYPE column in each case

In [83]:
```
print(f"The unique values of TYPE column in Non-Fradulent transactions are : {not_f
print(f"The unique values of TYPE column in Fradulent transactions are : {frauds_df
```

```
The unique values of TYPE column in Non-Fradulent transactions are : ['PAYMENT' 'DEB
IT' 'CASH_OUT' 'TRANSFER' 'CASH_IN']
The unique values of TYPE column in Fradulent transactions are : ['TRANSFER' 'CASH_O
UT']
```

2D. Observing the amount stats in both cases

In [84]:
```
print(f"The average value of executed transactions in Non-Fraudulent transactions a
print(f"The average value of executed transactions in Fraudulent transactions are :
```

```
The average value of executed transactions in Non-Fraudulent transactions are : 1781
97.04172740763
The average value of executed transactions in Fraudulent transactions are : 1461343.
157758936
```

In [85]:
```
print(f"The maximum value of executed transactions in Non-Fraudulent transactions a
print(f"The maximum value of executed transactions in Fraudulent transactions are :
```

```
The maximum value of executed transactions in Non-Fraudulent transactions are : 9244
5516.64
The maximum value of executed transactions in Fraudulent transactions are : 1000000
0.0
```

In [86]:
```
print(f"The minimum value of executed transactions in Non-Fraudulent transactions a
print(f"The minimum value of executed transactions in Fraudulent transactions are :
```

```
The minimum value of executed transactions in Non-Fraudulent transactions are : 0.01
The minimum value of executed transactions in Fraudulent transactions are : 0.0
```

In [87]:
```
print(f"The standard deviation between executed transactions in Non-Fraudulent tran
print(f"The standard deviation between executed transactions in Fraudulent transact
```

```
The standard deviation between executed transactions in Non-Fraudulent transactions
are : 596236.9813471774
The standard deviation between executed transactions in Fraudulent transactions are
: 2397046.563628217
```

In [88]:
```
print(f"The coefficient of variance of executed transactions in Non-Fraudulent tran
print(f"The coefficient of variance of executed transactions in Fraudulent transact
```

```
The coefficient of variance of executed transactions in Non-Fraudulent transactions
are : 3.345942085050187
The coefficient of variance of executed transactions in Fraudulent transactions are
: 1.6403036828832473
```

2D. Observing the oldbalanceOrg stats in both cases

In [89]:
```
print(f"The average value of oldbalanceOrg in Non-Fraudulent transactions are : {no
print(f"The average value of oldbalanceOrg in Fraudulent transactions are : {frauds
```

```
The average value of oldbalanceOrg in Non-Fraudulent transactions are : 832828.71172
72632
The average value of oldbalanceOrg in Fraudulent transactions are : 1637627.68592411
84
```

In [90]:
```
print(f"The maximum value of oldbalanceOrg in Non-Fraudulent transactions are : {no
print(f"The maximum value of oldbalanceOrg in Fraudulent transactions are : {frauds
```

```
The maximum value of oldbalanceOrg in Non-Fraudulent transactions are : 43818855.3
The maximum value of oldbalanceOrg in Fraudulent transactions are : 59585040.37
```

In [91]:
```
print(f"The minimum value of oldbalanceOrg in Non-Fraudulent transactions are : {no
print(f"The minimum value of oldbalanceOrg in Fraudulent transactions are : {frauds
```

```
The minimum value of oldbalanceOrg in Non-Fraudulent transactions are : 0.0
The minimum value of oldbalanceOrg in Fraudulent transactions are : 0.0
```

In [92]:
```
print(f"The standard deviation between oldbalanceOrg in Non-Fraudulent transactions
print(f"The standard deviation between oldbalanceOrg in Fraudulent transactions are
```

```
The standard deviation between oldbalanceOrg in Non-Fraudulent transactions are : 28
87144.030332925
The standard deviation between oldbalanceOrg in Fraudulent transactions are : 352809
9.5182469925
```

In [93]:
```
print(f"The coefficient of variance of oldbalanceOrg in Non-Fraudulent transactions
print(f"The coefficient of variance of oldbalanceOrg in Fraudulent transactions are
```

```
The coefficient of variance of oldbalanceOrg in Non-Fraudulent transactions are : 3.
4666720655500334
The coefficient of variance of oldbalanceOrg in Fraudulent transactions are : 2.1543
965997717454
```

2E. Observing the newbalanceOrig stats in both cases

In [94]:
```
print(f"The average value of newbalanceOrig in Non-Fraudulent transactions are : {n
print(f"The average value of newbalanceOrig in Fraudulent transactions are : {fraud
```

```
The average value of newbalanceOrig in Non-Fraudulent transactions are : 855970.2281
088118
The average value of newbalanceOrig in Fraudulent transactions are : 177508.20801268
757
```

In [95]:
```
print(f"The maximum value of newbalanceOrig in Non-Fraudulent transactions are : {n
print(f"The maximum value of newbalanceOrig in Fraudulent transactions are : {fraud
```

```
The maximum value of newbalanceOrig in Non-Fraudulent transactions are : 43686616.33
The maximum value of newbalanceOrig in Fraudulent transactions are : 49585040.37
```

In [96]:
```
print(f"The minimum value of newbalanceOrig in Non-Fraudulent transactions are : {n
print(f"The minimum value of newbalanceOrig in Fraudulent transactions are : {fraud
```

```
The minimum value of newbalanceOrig in Non-Fraudulent transactions are : 0.0
The minimum value of newbalanceOrig in Fraudulent transactions are : 0.0
```

In [97]:
```
print(f"The standard deviation between newbalanceOrig in Non-Fraudulent transaction
print(f"The standard deviation between newbalanceOrig in Fraudulent transactions ar
```

```
The standard deviation between newbalanceOrig in Non-Fraudulent transactions are : 2
924986.964649587
The standard deviation between newbalanceOrig in Fraudulent transactions are : 19153
77.8465069544
```

In [98]:
```
print(f"The coefficient of variance of newbalanceOrig in Non-Fraudulent transaction
print(f"The coefficient of variance of newbalanceOrig in Fraudulent transactions ar
```

```
The coefficient of variance of newbalanceOrig in Non-Fraudulent transactions are :
3.4171596962105553
The coefficient of variance of newbalanceOrig in Fraudulent transactions are : 10.79
0362135648685
```

2F. Observing the oldbalanceDest stats in both cases

In [99]:
```
print(f"The average value of oldbalanceDest in Non-Fraudulent transactions are : {n
print(f"The average value of oldbalanceDest in Fraudulent transactions are : {fraud
```

```
The average value of oldbalanceDest in Non-Fraudulent transactions are : 1101420.874
5693793
The average value of oldbalanceDest in Fraudulent transactions are : 545311.95821154
08
```

In [100…
```
print(f"The maximum value of oldbalanceDest in Non-Fraudulent transactions are : {n
print(f"The maximum value of oldbalanceDest in Fraudulent transactions are : {fraud
```

```
The maximum value of oldbalanceDest in Non-Fraudulent transactions are : 356015889.3
5
The maximum value of oldbalanceDest in Fraudulent transactions are : 236230516.82
```

In [101…
```
print(f"The minimum value of oldbalanceDest in Non-Fraudulent transactions are : {n
print(f"The minimum value of oldbalanceDest in Fraudulent transactions are : {fraud
```

```
The minimum value of oldbalanceDest in Non-Fraudulent transactions are : 0.0
The minimum value of oldbalanceDest in Fraudulent transactions are : 0.0
```

In [102…
```
print(f"The standard deviation between oldbalanceDest in Non-Fraudulent transaction
print(f"The standard deviation between oldbalanceDest in Fraudulent transactions ar
```

```
The standard deviation between oldbalanceDest in Non-Fraudulent transactions are : 3
399201.793378541
The standard deviation between oldbalanceDest in Fraudulent transactions are : 33395
89.253916916
```

In [103…
```
print(f"The coefficient of variance of oldbalanceDest in Non-Fraudulent transaction
print(f"The coefficient of variance of oldbalanceDest in Fraudulent transactions ar
```

```
The coefficient of variance of oldbalanceDest in Non-Fraudulent transactions are :
3.0861969950474393
The coefficient of variance of oldbalanceDest in Fraudulent transactions are : 6.124
181220726873
```

2G. Observing the newbalanceDest stats in both cases

In [104…
```
print(f"The average value of newbalanceDest in Non-Fraudulent transactions are : {n
print(f"The average value of newbalanceDest in Fraudulent transactions are : {fraud
```

```
The average value of newbalanceDest in Non-Fraudulent transactions are : 1224925.684
5631592
The average value of newbalanceDest in Fraudulent transactions are : 1282205.5214859
096
```

In [105…
```
print(f"The maximum value of newbalanceDest in Non-Fraudulent transactions are : {n
print(f"The maximum value of newbalanceDest in Fraudulent transactions are : {fraud
```

```
The maximum value of newbalanceDest in Non-Fraudulent transactions are : 356179278.9
2
The maximum value of newbalanceDest in Fraudulent transactions are : 236726494.66
```

In [106…
```
print(f"The minimum value of newbalanceDest in Non-Fraudulent transactions are : {n
print(f"The minimum value of newbalanceDest in Fraudulent transactions are : {fraud
```

```
The minimum value of newbalanceDest in Non-Fraudulent transactions are : 0.0
The minimum value of newbalanceDest in Fraudulent transactions are : 0.0
```

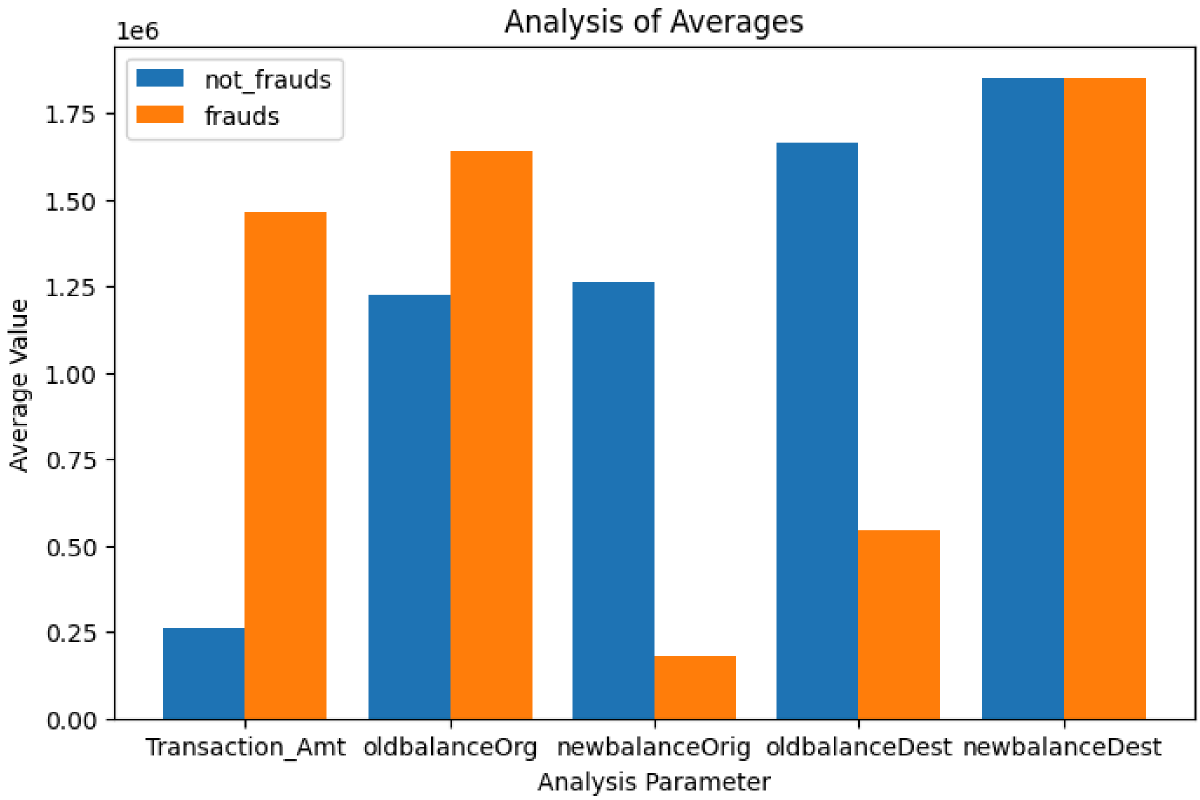
In [107…
```
print(f"The standard deviation between newbalanceDest in Non-Fraudulent transaction
print(f"The standard deviation between newbalanceDest in Fraudulent transactions ar
```

```
The standard deviation between newbalanceDest in Non-Fraudulent transactions are : 3
673815.7099226634
The standard deviation between newbalanceDest in Fraudulent transactions are : 39122
20.649584355
```

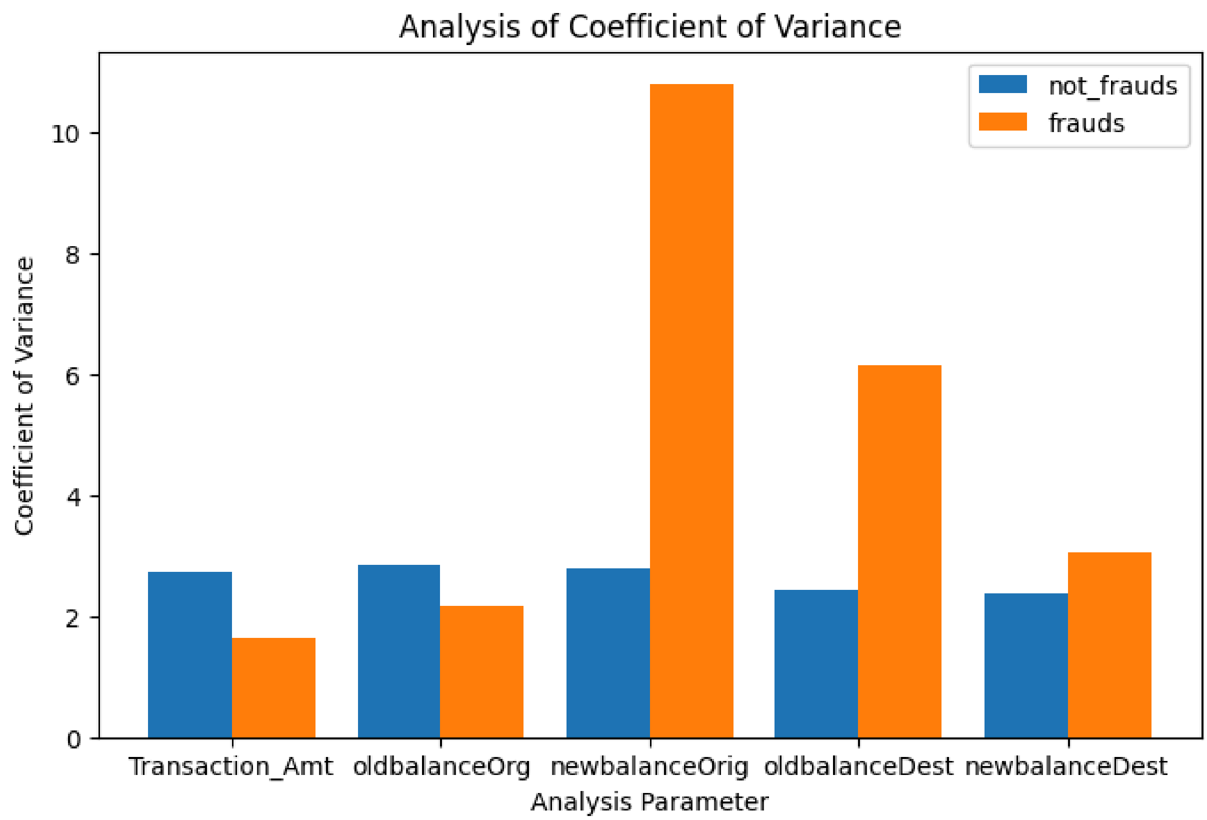
In [108…
```
print(f"The coefficient of variance of newbalanceDest in Non-Fraudulent transaction
print(f"The coefficient of variance of newbalanceDest in Fraudulent transactions ar
```

```
The coefficient of variance of newbalanceDest in Non-Fraudulent transactions are :
2.9992151819666044
The coefficient of variance of newbalanceDest in Fraudulent transactions are : 3.051
165030899726
```

In [32]:
```
X = ['Transaction_Amt','oldbalanceOrg','newbalanceOrig','oldbalanceDest', 'newbalan
not_frauds = [not_frauds_df['amount'].mean(), not_frauds_df['oldbalanceOrg'].mean()
frauds = [frauds_df['amount'].mean(),frauds_df['oldbalanceOrg'].mean(),frauds_df['n

X_axis = np.arange(len(X))
plt.figure(figsize=(8, 5))
plt.bar(X_axis - 0.2, not_frauds, 0.4, label = 'not_frauds')
plt.bar(X_axis + 0.2, frauds, 0.4, label = 'frauds')
plt.xticks(X_axis, X)
plt.xlabel("Analysis Parameter")
plt.ylabel("Average Value")
plt.title("Analysis of Averages")
plt.legend()
plt.show()
```



In [31]:
```
X = ['Transaction_Amt','oldbalanceOrg','newbalanceOrig','oldbalanceDest', 'newbalan
not_frauds = [not_frauds_df['amount'].std()/not_frauds_df['amount'].mean(),
              not_frauds_df['oldbalanceOrg'].std()/not_frauds_df['oldbalanceOrg'].m
              not_frauds_df['newbalanceOrig'].std()/not_frauds_df['newbalanceOrig']
              not_frauds_df['oldbalanceDest'].std()/not_frauds_df['oldbalanceDest']
              not_frauds_df['newbalanceDest'].std()/not_frauds_df['newbalanceDest']
             ]
frauds = [frauds_df['amount'].std()/frauds_df['amount'].mean(),
              frauds_df['oldbalanceOrg'].std()/frauds_df['oldbalanceOrg'].mean(),
              frauds_df['newbalanceOrig'].std()/frauds_df['newbalanceOrig'].mean(),
              frauds_df['oldbalanceDest'].std()/frauds_df['oldbalanceDest'].mean(),
              frauds_df['newbalanceDest'].std()/frauds_df['newbalanceDest'].mean()
             ]

X_axis = np.arange(len(X))
plt.figure(figsize=(8, 5))
plt.bar(X_axis - 0.2, not_frauds, 0.4, label = 'not_frauds')
plt.bar(X_axis + 0.2, frauds, 0.4, label = 'frauds')
plt.xticks(X_axis, X)
plt.xlabel("Analysis Parameter")
plt.ylabel("Coefficient of Variance")
plt.title("Analysis of Coefficient of Variance")
plt.legend()
plt.show()
```



3. Analyzing Fraudulent Customers

In [109…

```
frauds_df
```

Out[109…

	step	type	amount	nameOrig	oldbalanceOrg	newbalanceOrig	na
2	1	TRANSFER	181.00	C1305486145	181.00	0.0	C55
3	1	CASH_OUT	181.00	C840083671	181.00	0.0	C3
251	1	TRANSFER	2806.00	C1420196421	2806.00	0.0	C97
252	1	CASH_OUT	2806.00	C2101527076	2806.00	0.0	C10C
680	1	TRANSFER	20128.00	C137533655	20128.00	0.0	C184
...	...	...	...	...	...	...	
6362615	743	CASH_OUT	339682.13	C786484425	339682.13	0.0	C77
6362616	743	TRANSFER	6311409.28	C1529008245	6311409.28	0.0	C188
6362617	743	CASH_OUT	6311409.28	C1162922333	6311409.28	0.0	C136
6362618	743	TRANSFER	850002.52	C1685995037	850002.52	0.0	C208
6362619	743	CASH_OUT	850002.52	C1280323807	850002.52	0.0	C87

8197 rows × 13 columns

In [110…

```
frauds_df.shape
```

Out[110…

```
(8197, 13)
```

In [111…

```
frauds_df.groupby("type").count()
```

Out[111…

type	step	amount	nameOrig	oldbalanceOrg	newbalanceOrig	nameDest	oldbala
CASH_OUT	4116	4116	4116	4116	4116	4116	
TRANSFER	4081	4081	4081	4081	4081	4081	

Data Cleaning

Removing flagged rows

In [3]:

```
df = df[df['isFlaggedFraud'] !=1]
df
```

Out[3]:

	step	type	amount	nameOrig	oldbalanceOrg	newbalanceOrig	n
0	1	PAYMENT	9839.64	C1231006815	170136.00	160296.36	M19
1	1	PAYMENT	1864.28	C1666544295	21249.00	19384.72	M20
2	1	TRANSFER	181.00	C1305486145	181.00	0.00	C5
3	1	CASH_OUT	181.00	C840083671	181.00	0.00	C
4	1	PAYMENT	11668.14	C2048537720	41554.00	29885.86	M12
...	...	...	...	...	...	...	
6362615	743	CASH_OUT	339682.13	C786484425	339682.13	0.00	C7
6362616	743	TRANSFER	6311409.28	C1529008245	6311409.28	0.00	C18
6362617	743	CASH_OUT	6311409.28	C1162922333	6311409.28	0.00	C13
6362618	743	TRANSFER	850002.52	C1685995037	850002.52	0.00	C20
6362619	743	CASH_OUT	850002.52	C1280323807	850002.52	0.00	C8

6362604 rows × 11 columns

Removing Merchants

In [4]:
```
df['destCustomerType'] = df['nameDest'].str[0]
df = df[df['destCustomerType'] != "M"]

df
```

Out[4]:

	step	type	amount	nameOrig	oldbalanceOrg	newbalanceOrig	na
2	1	TRANSFER	181.00	C1305486145	181.00	0.00	C55
3	1	CASH_OUT	181.00	C840083671	181.00	0.00	C3
9	1	DEBIT	5337.77	C712410124	41720.00	36382.23	C19
10	1	DEBIT	9644.94	C1900366749	4465.00	0.00	C99
15	1	CASH_OUT	229133.94	C905080434	15325.00	0.00	C47
...	...	...	...	...	...	...	
6362615	743	CASH_OUT	339682.13	C786484425	339682.13	0.00	C77
6362616	743	TRANSFER	6311409.28	C1529008245	6311409.28	0.00	C188
6362617	743	CASH_OUT	6311409.28	C1162922333	6311409.28	0.00	C136
6362618	743	TRANSFER	850002.52	C1685995037	850002.52	0.00	C208
6362619	743	CASH_OUT	850002.52	C1280323807	850002.52	0.00	C87

4211109 rows × 12 columns

In [5]:
```
df['isFraud'].value_counts()
```

Out[5]:
```
isFraud
0    4202912
1       8197
Name: count, dtype: int64
```

In [7]:
```
target_rate = df['isFraud'].mean() * 100
target_rate
```

Out[7]:
```
np.float64(0.19465181262228073)
```

Target rate is 0.19%

Removing the aux columns

In [8]:
```
df = df.drop(['step', 'nameOrig', 'destCustomerType', 'nameDest', 'isFlaggedFraud']
df
```

Out[8]:

	type	amount	oldbalanceOrg	newbalanceOrig	oldbalanceDest	newbala
2	TRANSFER	181.00	181.00	0.00	0.00	
3	CASH_OUT	181.00	181.00	0.00	21182.00	
9	DEBIT	5337.77	41720.00	36382.23	41898.00	4
10	DEBIT	9644.94	4465.00	0.00	10845.00	1!
15	CASH_OUT	229133.94	15325.00	0.00	5083.00	!
...	...	...	...	...	...	
6362615	CASH_OUT	339682.13	339682.13	0.00	0.00	3:
6362616	TRANSFER	6311409.28	6311409.28	0.00	0.00	
6362617	CASH_OUT	6311409.28	6311409.28	0.00	68488.84	63
6362618	TRANSFER	850002.52	850002.52	0.00	0.00	
6362619	CASH_OUT	850002.52	850002.52	0.00	6510099.11	73

4211109 rows × 7 columns

In [10]:

```
# Replace values in the 'transaction_type' column
replacement_dict = {
    'TRANSFER': 2,
    'CASH_OUT': 4,
    'DEBIT': 6,
    'CASH_IN': 8
}

df['transaction_type'] = df['type'].replace(replacement_dict)
df = df.drop(['type'], axis=1)

df
```

```
C:\Users\ketan\AppData\Local\Temp\ipykernel_2152\1011219186.py:9: FutureWarning: Dow
ncasting behavior in `replace` is deprecated and will be removed in a future versio
n. To retain the old behavior, explicitly call `result.infer_objects(copy=False)`. T
o opt-in to the future behavior, set `pd.set_option('future.no_silent_downcasting',
True)`
  df['transaction_type'] = df['type'].replace(replacement_dict)
```

Out[10]:

	amount	oldbalanceOrg	newbalanceOrig	oldbalanceDest	newbalanceDest	isF
2	181.00	181.00	0.00	0.00	0.00	
3	181.00	181.00	0.00	21182.00	0.00	
9	5337.77	41720.00	36382.23	41898.00	40348.79	
10	9644.94	4465.00	0.00	10845.00	157982.12	
15	229133.94	15325.00	0.00	5083.00	51513.44	
...	...	...	...	...	...	
6362615	339682.13	339682.13	0.00	0.00	339682.13	
6362616	6311409.28	6311409.28	0.00	0.00	0.00	
6362617	6311409.28	6311409.28	0.00	68488.84	6379898.11	
6362618	850002.52	850002.52	0.00	0.00	0.00	
6362619	850002.52	850002.52	0.00	6510099.11	7360101.63	

4211109 rows × 7 columns

Feature Generation

In [11]:
```
df['diff_newbalanceDest_oldbalanceDest'] = df['newbalanceDest'] - df['oldbalanceDes
df['diff_newbalanceDest_newbalanceOrig'] = df['newbalanceDest'] - df['newbalanceOri
df['diff_newbalanceOrig_oldbalanceOrg'] = df['newbalanceOrig'] - df['oldbalanceOrg'
```

In [12]:
```
df
```

Out[12]:

	amount	oldbalanceOrg	newbalanceOrig	oldbalanceDest	newbalanceDest	isF
2	181.00	181.00	0.00	0.00	0.00	
3	181.00	181.00	0.00	21182.00	0.00	
9	5337.77	41720.00	36382.23	41898.00	40348.79	
10	9644.94	4465.00	0.00	10845.00	157982.12	
15	229133.94	15325.00	0.00	5083.00	51513.44	
...	...	...	...	...	...	
6362615	339682.13	339682.13	0.00	0.00	339682.13	
6362616	6311409.28	6311409.28	0.00	0.00	0.00	
6362617	6311409.28	6311409.28	0.00	68488.84	6379898.11	
6362618	850002.52	850002.52	0.00	0.00	0.00	
6362619	850002.52	850002.52	0.00	6510099.11	7360101.63	

4211109 rows × 10 columns

In [21]:
```
df.to_csv("../data/frauds_feats.csv", index=False)
```

Data Splitting

In [3]:
```
df['isFraud'].value_counts()
```

Out[3]:
```
isFraud
0    4202912
1       8197
Name: count, dtype: int64
```

In [4]:
```
not_frauds = df[df['isFraud']==0]
frauds = df[df['isFraud']==1]
```

In [5]:
```
not_frauds_test = not_frauds.sample(frac=0.2, random_state=1576023)
not_frauds_train = not_frauds.drop(not_frauds_test.index)
```

In [8]:
```
frauds_test = frauds.sample(frac=0.2, random_state=4014)
frauds_train = frauds.drop(frauds_test.index)
```

In [9]:
```
training_df = pd.concat([not_frauds_train, frauds_train], ignore_index=True)
training_df = training_df.sample(frac=1)

training_df
```

Out[9]:

	amount	oldbalanceOrg	newbalanceOrig	oldbalanceDest	newbalanceDest	isFr
1298219	7758.19	0.00	0.00	354325.90	362084.10	
1366762	123387.17	12616828.00	12740215.17	183116.89	59729.72	
1005718	139038.19	0.00	0.00	5537247.72	5676285.91	
1640726	176770.24	0.00	0.00	974197.63	1150967.87	
940041	133556.12	0.00	0.00	575465.10	709021.23	
...	...	...	...	...	...	
1409038	216099.80	200600.00	416699.80	0.00	0.00	
913902	70014.07	308427.00	238412.93	61140.34	131154.40	
2333897	48690.81	9455616.92	9504307.73	128225.02	79534.21	
3007983	8190.44	12368.00	4177.56	0.00	8190.44	
98446	23904.98	1578.00	0.00	0.00	23904.98	

3368888 rows × 10 columns

In [10]:

```
out_of_sample_validation_df = pd.concat([not_frauds_test, frauds_test], ignore_inde
out_of_sample_validation_df = out_of_sample_validation_df.sample(frac=1)

out_of_sample_validation_df
```

Out[10]:

	amount	oldbalanceOrg	newbalanceOrig	oldbalanceDest	newbalanceDest	isFra
227843	278065.16	0.0	0.00	944658.21	1222723.37	
661152	141449.48	104.0	0.00	112551.16	254000.63	
438917	236523.18	92191.0	0.00	878301.14	1114824.32	
61278	234387.18	3477569.8	3711956.98	5725031.17	5644143.01	
583835	136691.50	27000.0	0.00	19993.32	156684.82	
...	...	...	...	...	...	
303562	127191.61	20244.0	147435.61	0.00	0.00	
188511	335245.63	39670.0	374915.63	1738807.47	1403561.84	
772975	345132.39	5049.0	0.00	2483530.89	2828663.28	
558745	121787.74	0.0	0.00	1991066.22	2112853.96	
695017	237284.98	7096.0	244380.98	0.00	0.00	

842221 rows × 10 columns

In [11]:
```
training_df.to_csv("../data/frauds_training_data.csv", index=False)
```

In [13]:
```
out_of_sample_validation_df.to_csv("../data/frauds_out_of_sample_validation_data.cs
```

Down Sampling

In [2]:
```
os.chdir('.')

file_path = "../data/frauds_training_data.csv"
try:
    df = pd.read_csv(file_path, encoding='latin1')
except UnicodeDecodeError:
    df = pd.read_csv(file_path, encoding='iso-8859-1')
except UnicodeDecodeError:
    df = pd.read_csv(file_path, encoding='cp1252')

df.head()
```

Out[2]:

	amount	oldbalanceOrg	newbalanceOrig	oldbalanceDest	newbalanceDest	isFraud	tı
0	7758.19	0.0	0.00	354325.90	362084.10	0	
1	123387.17	12616828.0	12740215.17	183116.89	59729.72	0	
2	139038.19	0.0	0.00	5537247.72	5676285.91	0	
3	176770.24	0.0	0.00	974197.63	1150967.87	0	
4	133556.12	0.0	0.00	575465.10	709021.23	0	

In [3]:
```
df.shape
```

Out[3]:
```
(3368888, 10)
```

In [4]:
```
df['isFraud'].value_counts()
```

Out[4]:
```
isFraud
0    3362330
1       6558
Name: count, dtype: int64
```

Downsampling the 0s for increasing the target rate

In [5]:
```
not_frauds = df[df['isFraud']==0]
frauds = df[df['isFraud']==1]
```

In [6]:
```
not_frauds = not_frauds.sample(n=125000, random_state=14896)
not_frauds
```

Out[6]:

	amount	oldbalanceOrg	newbalanceOrig	oldbalanceDest	newbalanceDest	isF
1390555	426609.30	656677.43	1083286.74	2903172.86	2476563.56	
175607	195579.94	0.00	0.00	1461406.11	1656986.06	
1045907	281977.59	14584.00	0.00	528370.88	810348.47	
2284453	30811.92	21031.00	0.00	11929.95	42741.86	
47330	98965.90	0.00	0.00	8513266.99	8612232.89	
...	...	...	...	...	...	
87769	68149.29	10121.29	0.00	542612.51	610761.80	
377519	3118429.05	21290.00	0.00	225727.54	3344156.59	
3305228	8085.88	0.00	0.00	5256419.35	5264505.23	
1746318	12543.14	0.00	0.00	93728.20	106271.34	
859162	3568.51	549155.59	552724.10	0.00	0.00	

125000 rows × 10 columns

In [7]:

```python
downsampled_training_df = pd.concat([not_frauds, frauds], ignore_index=True)
downsampled_training_df = downsampled_training_df.sample(frac=1)

downsampled_training_df
```

Out[7]:

	amount	oldbalanceOrg	newbalanceOrig	oldbalanceDest	newbalanceDest	isFr
93128	135528.26	0.00	0.00	1545084.23	1680612.49	
54382	167677.85	17497.79	0.00	222658.12	390335.97	
110670	4968.27	20942.00	15973.73	1124530.63	1129498.90	
89349	98098.47	0.00	0.00	783723.06	881821.53	
32554	78222.46	0.00	0.00	8641589.45	8719811.91	
...	...	...	...	...	...	
91722	135643.68	0.00	0.00	2081989.64	2217633.32	
129123	2627070.50	2627070.50	0.00	463083.63	3090154.12	
48484	580620.02	0.00	0.00	1443644.27	2024264.29	
98803	125571.71	21436.00	0.00	108619.67	234191.38	
62777	111437.03	31434.09	0.00	152159.81	263596.83	

131558 rows × 10 columns

In [8]:
```
downsampled_training_df.to_csv("../data/downsampled_training_df.csv", index=False)
```

In [ ]:

In [2]:
```
os.chdir('.')

file_path = "../data/downsampled_training_df.csv"
try:
    df = pd.read_csv(file_path, encoding='latin1')
except UnicodeDecodeError:
    df = pd.read_csv(file_path, encoding='iso-8859-1')
except UnicodeDecodeError:
    df = pd.read_csv(file_path, encoding='cp1252')

df.head()
```

Out[2]:

	amount	oldbalanceOrg	newbalanceOrig	oldbalanceDest	newbalanceDest	isFraud	tı
0	135528.26	0.00	0.00	1545084.23	1680612.49	0	
1	167677.85	17497.79	0.00	222658.12	390335.97	0	
2	4968.27	20942.00	15973.73	1124530.63	1129498.90	0	
3	98098.47	0.00	0.00	783723.06	881821.53	0	
4	78222.46	0.00	0.00	8641589.45	8719811.91	0	

In [4]:
```
df.shape
```

Out[4]:
```
(131558, 10)
```

In [5]:
```
df['isFraud'].value_counts()
```

Out[5]:
```
isFraud
0    125000
1      6558
Name: count, dtype: int64
```

In [6]:
```
target_rate = df['isFraud'].mean() * 100
target_rate
```

Out[6]:
```
np.float64(4.984873591875827)
```

In [4]:
```
features = df.drop(columns=['isFraud'])
targets = df['isFraud']
```

In [12]:
```
features
```

Out[12]:

	amount	oldbalanceOrg	newbalanceOrig	oldbalanceDest	newbalanceDest	trar
0	135528.26	0.00	0.00	1545084.23	1680612.49	
1	167677.85	17497.79	0.00	222658.12	390335.97	
2	4968.27	20942.00	15973.73	1124530.63	1129498.90	
3	98098.47	0.00	0.00	783723.06	881821.53	
4	78222.46	0.00	0.00	8641589.45	8719811.91	
...	...	...	...	...	...	
131553	135643.68	0.00	0.00	2081989.64	2217633.32	
131554	2627070.50	2627070.50	0.00	463083.63	3090154.12	
131555	580620.02	0.00	0.00	1443644.27	2024264.29	
131556	125571.71	21436.00	0.00	108619.67	234191.38	
131557	111437.03	31434.09	0.00	152159.81	263596.83	

131558 rows × 9 columns

Low Variance Check

In [15]:

```python
data_variance = features.var()
low_variance_threshold = 0.25
low_variance_columns = data_variance[data_variance < low_variance_threshold].index.

print("Variance of each column:\n", data_variance)
print("\nColumns with low variance:", low_variance_columns)
```

```
Variance of each column:
 amount                              8.961479e+11
oldbalanceOrg                       1.196931e+13
newbalanceOrig                      1.187012e+13
oldbalanceDest                      1.501913e+13
newbalanceDest                      1.758966e+13
transaction_type                    4.677095e+00
diff_newbalanceDest_oldbalanceDest  1.060711e+12
diff_newbalanceDest_newbalanceOrig  2.959811e+13
diff_newbalanceOrig_oldbalanceOrg   3.990858e+11
dtype: float64

Columns with low variance: []
```

All columns have passed the low variance check

In [ ]:

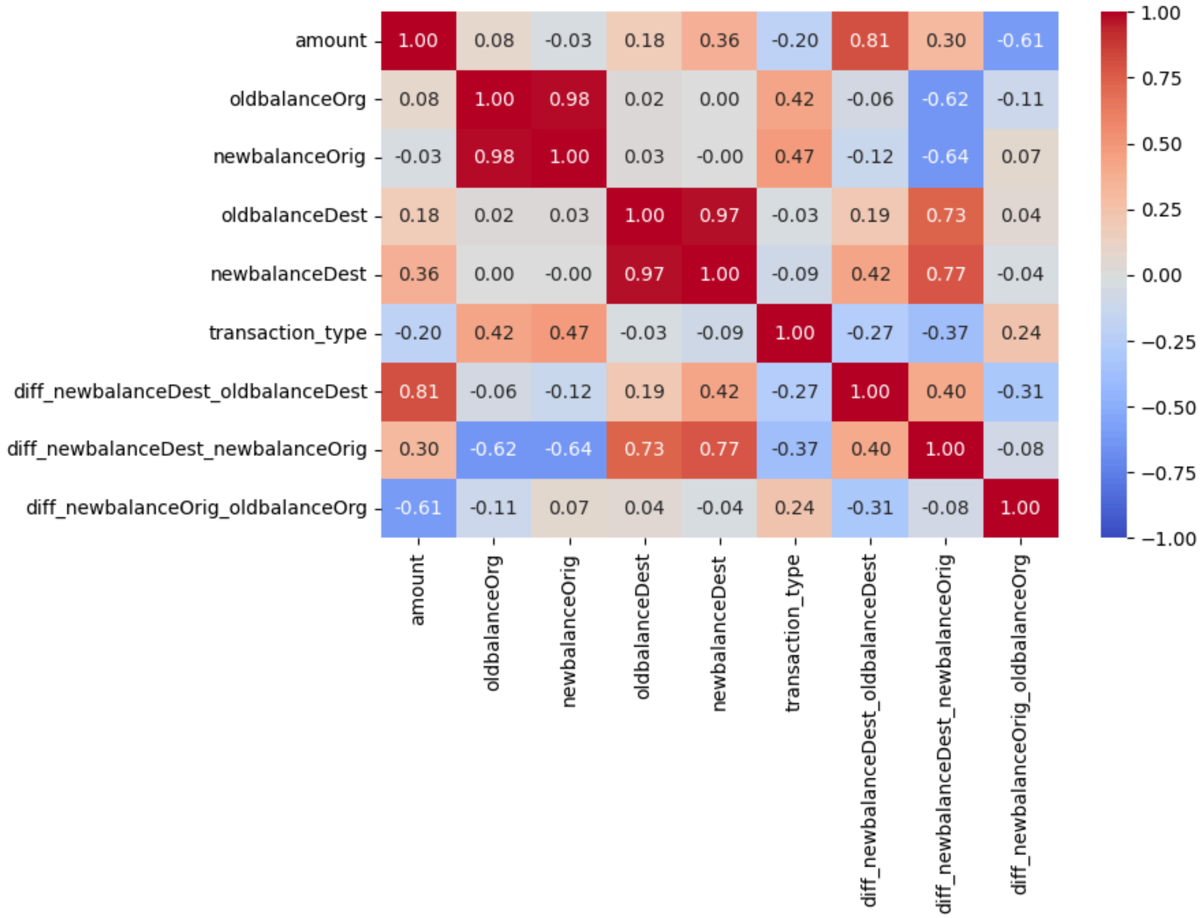
Correlation Check

In [5]:
```
correlation_matrix = features.corr()

correlation_matrix
```

Out[5]:

	amount	oldbalanceOrg	newbalanceOrig	oldbalance
amount	1.000000	0.079145	-0.032433	0.17
oldbalanceOrg	0.079145	1.000000	0.983268	0.02
newbalanceOrig	-0.032433	0.983268	1.000000	0.02
oldbalanceDest	0.177420	0.020046	0.026568	1.00
newbalanceDest	0.362308	0.003261	-0.004787	0.97
transaction_type	-0.195788	0.424545	0.470266	-0.02
diff_newbalanceDest_oldbalanceDest	0.807776	-0.062153	-0.119467	0.18
diff_newbalanceDest_newbalanceOrig	0.299841	-0.620170	-0.636970	0.73
diff_newbalanceOrig_oldbalanceOrg	-0.610318	-0.113992	0.068893	0.03

In [7]:
```
import seaborn as sns
import matplotlib.pyplot as plt

plt.figure(figsize=(8, 5))

# Generate a heatmap with annotated values
sns.heatmap(correlation_matrix, annot=True, fmt='.2f', cmap='coolwarm', vmin=-1, vm

# Show the plot
plt.show()
```



Removing one of the features from pairs with correlation more than 0.85

In [17]:
```
drop_feats = ['oldbalanceOrg', 'oldbalanceDest']
features = features.drop(columns=drop_feats)

features
```

Out[17]:

	amount	newbalanceOrig	newbalanceDest	transaction_type	diff_newbalanceDe
0	135528.26	0.00	1680612.49	2	
1	167677.85	0.00	390335.97	4	
2	4968.27	15973.73	1129498.90	6	
3	98098.47	0.00	881821.53	4	
4	78222.46	0.00	8719811.91	2	
...	...	...	...	...	
131553	135643.68	0.00	2217633.32	4	
131554	2627070.50	0.00	3090154.12	4	
131555	580620.02	0.00	2024264.29	4	
131556	125571.71	0.00	234191.38	4	
131557	111437.03	0.00	263596.83	4	

131558 rows × 7 columns

Boruta Feature Selection

In [22]:

```
from sklearn.ensemble import RandomForestClassifier
from boruta import BorutaPy

import numpy as np
np.int = int
np.float = float

# Initialize RandomForestClassifier
rf = RandomForestClassifier(n_estimators=100, random_state=2569)

# Initialize Boruta feature selector
boruta_selector = BorutaPy(rf, n_estimators='auto', random_state=742)

# Fit Boruta selector to the data
boruta_selector.fit(features.values, targets)

# Get the selected features
selected_features = features.columns[boruta_selector.support_]

print("Selected features:", selected_features)
```

```
C:\Users\ketan\PycharmProjects\CrimeDetectionProject\venv\Lib\site-packages\sklearn
\utils\validation.py:1339: DataConversionWarning: A column-vector y was passed when
a 1d array was expected. Please change the shape of y to (n_samples, ), for example
using ravel().
  y = column_or_1d(y, warn=True)
C:\Users\ketan\PycharmProjects\CrimeDetectionProject\venv\Lib\site-packages\sklearn
\base.py:1473: DataConversionWarning: A column-vector y was passed when a 1d array w
as expected. Please change the shape of y to (n_samples,), for example using ravel
().
  return fit_method(estimator, *args, **kwargs)
C:\Users\ketan\PycharmProjects\CrimeDetectionProject\venv\Lib\site-packages\sklearn
\base.py:1473: DataConversionWarning: A column-vector y was passed when a 1d array w
as expected. Please change the shape of y to (n_samples,), for example using ravel
().
  return fit_method(estimator, *args, **kwargs)
C:\Users\ketan\PycharmProjects\CrimeDetectionProject\venv\Lib\site-packages\sklearn
\base.py:1473: DataConversionWarning: A column-vector y was passed when a 1d array w
as expected. Please change the shape of y to (n_samples,), for example using ravel
().
  return fit_method(estimator, *args, **kwargs)
C:\Users\ketan\PycharmProjects\CrimeDetectionProject\venv\Lib\site-packages\sklearn
\base.py:1473: DataConversionWarning: A column-vector y was passed when a 1d array w
as expected. Please change the shape of y to (n_samples,), for example using ravel
().
  return fit_method(estimator, *args, **kwargs)
C:\Users\ketan\PycharmProjects\CrimeDetectionProject\venv\Lib\site-packages\sklearn
\base.py:1473: DataConversionWarning: A column-vector y was passed when a 1d array w
as expected. Please change the shape of y to (n_samples,), for example using ravel
().
  return fit_method(estimator, *args, **kwargs)
C:\Users\ketan\PycharmProjects\CrimeDetectionProject\venv\Lib\site-packages\sklearn
\base.py:1473: DataConversionWarning: A column-vector y was passed when a 1d array w
as expected. Please change the shape of y to (n_samples,), for example using ravel
().
  return fit_method(estimator, *args, **kwargs)
C:\Users\ketan\PycharmProjects\CrimeDetectionProject\venv\Lib\site-packages\sklearn
\base.py:1473: DataConversionWarning: A column-vector y was passed when a 1d array w
as expected. Please change the shape of y to (n_samples,), for example using ravel
().
  return fit_method(estimator, *args, **kwargs)
C:\Users\ketan\PycharmProjects\CrimeDetectionProject\venv\Lib\site-packages\sklearn
\base.py:1473: DataConversionWarning: A column-vector y was passed when a 1d array w
as expected. Please change the shape of y to (n_samples,), for example using ravel
().
  return fit_method(estimator, *args, **kwargs)
Selected features: Index(['amount', 'newbalanceOrig', 'newbalanceDest', 'transaction
_type',
       'diff_newbalanceDest_oldbalanceDest',
       'diff_newbalanceDest_newbalanceOrig',
       'diff_newbalanceOrig_oldbalanceOrg'],
      dtype='object')
```

In [23]:
```
selected_features_df = features[selected_features]
selected_features_df
```

Out[23]:

	amount	newbalanceOrig	newbalanceDest	transaction_type	diff_newbalanceDe
0	135528.26	0.00	1680612.49	2	
1	167677.85	0.00	390335.97	4	
2	4968.27	15973.73	1129498.90	6	
3	98098.47	0.00	881821.53	4	
4	78222.46	0.00	8719811.91	2	
...	...	...	...	...	
131553	135643.68	0.00	2217633.32	4	
131554	2627070.50	0.00	3090154.12	4	
131555	580620.02	0.00	2024264.29	4	
131556	125571.71	0.00	234191.38	4	
131557	111437.03	0.00	263596.83	4	

131558 rows × 7 columns

In [30]:

```
processed_training_df = selected_features_df.join(targets)
processed_training_df
```

Out[30]:

	amount	newbalanceOrig	newbalanceDest	transaction_type	diff_newbalanceDe
0	135528.26	0.00	1680612.49	2	
1	167677.85	0.00	390335.97	4	
2	4968.27	15973.73	1129498.90	6	
3	98098.47	0.00	881821.53	4	
4	78222.46	0.00	8719811.91	2	
...	...	...	...	...	
131553	135643.68	0.00	2217633.32	4	
131554	2627070.50	0.00	3090154.12	4	
131555	580620.02	0.00	2024264.29	4	
131556	125571.71	0.00	234191.38	4	
131557	111437.03	0.00	263596.83	4	

131558 rows × 8 columns

In [31]:

```
processed_training_df.to_csv("../data/processed_training_df.csv", index=False)
```

Modelling

In [84]:
```
os.chdir('.')

file_path = "../data/processed_training_df.csv"
try:
    df = pd.read_csv(file_path, encoding='latin1')
except UnicodeDecodeError:
    df = pd.read_csv(file_path, encoding='iso-8859-1')
except UnicodeDecodeError:
    df = pd.read_csv(file_path, encoding='cp1252')

df.head()
```

Out[84]:

	amount	newbalanceOrig	newbalanceDest	transaction_type	diff_newbalanceDest_oldba
0	135528.26	0.00	1680612.49	2	
1	167677.85	0.00	390335.97	4	
2	4968.27	15973.73	1129498.90	6	
3	98098.47	0.00	881821.53	4	
4	78222.46	0.00	8719811.91	2	

In [85]:
```
df.shape
```

Out[85]:
```
(131558, 8)
```

In [86]:
```
df['isFraud'].value_counts()
```

Out[86]:
```
isFraud
0    125000
1      6558
Name: count, dtype: int64
```

In [87]:
```
X_train = df.drop(columns=['isFraud'])
y_train = df['isFraud']
```

In [2]:
```
validation_df = pd.read_csv("../data/frauds_out_of_sample_validation_data.csv")
validation_df
```

Out[2]:

	amount	oldbalanceOrg	newbalanceOrig	oldbalanceDest	newbalanceDest	isFra
0	278065.16	0.0	0.00	944658.21	1222723.37	
1	141449.48	104.0	0.00	112551.16	254000.63	
2	236523.18	92191.0	0.00	878301.14	1114824.32	
3	234387.18	3477569.8	3711956.98	5725031.17	5644143.01	
4	136691.50	27000.0	0.00	19993.32	156684.82	
...	...	...	...	...	...	
842216	127191.61	20244.0	147435.61	0.00	0.00	
842217	335245.63	39670.0	374915.63	1738807.47	1403561.84	
842218	345132.39	5049.0	0.00	2483530.89	2828663.28	
842219	121787.74	0.0	0.00	1991066.22	2112853.96	
842220	237284.98	7096.0	244380.98	0.00	0.00	

842221 rows × 10 columns

In [3]:

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

Out[3]:

```
isFraud
0    840582
1      1639
Name: count, dtype: int64
```

In [4]:

```
validation_df['isFraud'].mean()
```

Out[4]:

```
0.0019460450404347554
```

In [89]:

```
validation_df = validation_df.drop(['oldbalanceOrg', 'oldbalanceDest'], axis=1)
```

In [90]:

```
X_test = validation_df.drop(columns=['isFraud'])
y_test = validation_df['isFraud']
```

Normalization

In [66]:

```
from sklearn.preprocessing import MinMaxScaler

scaler = MinMaxScaler()
X_train = pd.DataFrame(scaler.fit_transform(X_train), columns=X_train.columns)
X_test = pd.DataFrame(scaler.fit_transform(X_test), columns=X_test.columns)
```

Isolation Forest

In [48]:
```
from sklearn.ensemble import IsolationForest
from sklearn.metrics import accuracy_score, precision_score, recall_score, roc_auc_

# Train Isolation Forest model
model = IsolationForest(contamination=0.05)
model.fit(X_train)
```

Out[48]:
```
IsolationForest
IsolationForest(contamination=0.05)
```

In [50]:
```
# Predict anomalies
predictions = model.predict(X_test)

# Convert Isolation Forest predictions to match true labels
# For Isolation Forest, -1 indicates anomaly and 1 indicates normal. Convert to 0 a
predictions = (predictions == -1).astype(int)

# Compute metrics
accuracy = accuracy_score(y_test, predictions)
precision = precision_score(y_test, predictions)
recall = recall_score(y_test, predictions)
roc_auc = roc_auc_score(y_test, predictions)

# Print results
print(f"Accuracy: {accuracy:.4f}")
print(f"Precision: {precision:.4f}")
print(f"Recall: {recall:.4f}")
print(f"AUC: {roc_auc:.4f}")
```

```
Accuracy: 0.9577
Precision: 0.0124
Recall: 0.2636
AUC: 0.6113
```

One Class SVM

In [51]:
```
from sklearn.svm import OneClassSVM

model = OneClassSVM(nu=0.05)
model.fit(X_train)
```

Out[51]:
```
OneClassSVM
OneClassSVM(nu=0.05)
```

In [104…
```
# Predict anomalies
predictions = model.predict(X_test)

predictions = (predictions == -1).astype(int)

accuracy = accuracy_score(y_test, predictions)
```

```
precision = precision_score(y_test, predictions)
recall = recall_score(y_test, predictions)
roc_auc = roc_auc_score(y_test, predictions)

print(f"Accuracy: {accuracy:.4f}")
print(f"Precision: {precision:.4f}")
print(f"Recall: {recall:.4f}")
print(f"AUC: {roc_auc:.4f}")
```

```
C:\Users\ketan\PycharmProjects\CrimeDetectionProject\venv\Lib\site-packages\sklearn
\base.py:493: UserWarning: X does not have valid feature names, but OneClassSVM was
fitted with feature names
  warnings.warn(
Accuracy: 0.9538
Precision: 0.0077
Recall: 0.1788
AUC: 0.5670
```

Autoencoder Neural Network

In [20]:
```
import numpy as np
from tensorflow.keras.models import Model
from tensorflow.keras.layers import Input, Dense
from tensorflow.keras import regularizers
from scipy.sparse import issparse

# Convert to numpy arrays and ensure correct type
X_train = np.array(X_train, dtype=np.float32)
X_test = np.array(X_test, dtype=np.float32)

# Convert sparse matrices to dense arrays if necessary
if issparse(X_train):
    X_train = X_train.toarray()
if issparse(X_test):
    X_test = X_test.toarray()

# Check for NaNs and Infinities
X_train = np.nan_to_num(X_train)
X_test = np.nan_to_num(X_test)

# Define autoencoder model
input_dim = X_train.shape[1]
input_layer = Input(shape=(input_dim,))
encoded = Dense(64, activation='relu')(input_layer)
encoded = Dense(32, activation='relu')(encoded)
decoded = Dense(64, activation='relu')(encoded)
decoded = Dense(input_dim, activation='sigmoid')(decoded)

autoencoder = Model(input_layer, decoded)
autoencoder.compile(optimizer='adam', loss='binary_crossentropy')

# Fit the model
autoencoder.fit(X_train, X_train, epochs=50, batch_size=256, shuffle=True, validati
```

```
Epoch 1/50
514/514 ━━━━━━━━━━━━━━━━━━━━ 6s 9ms/step - loss: 0.2997 - val_loss: 0.2099
Epoch 2/50
514/514 ━━━━━━━━━━━━━━━━━━━━ 7s 13ms/step - loss: 0.2028 - val_loss: 0.2089
Epoch 3/50
514/514 ━━━━━━━━━━━━━━━━━━━━ 8s 9ms/step - loss: 0.2019 - val_loss: 0.2083
Epoch 4/50
514/514 ━━━━━━━━━━━━━━━━━━━━ 6s 11ms/step - loss: 0.2016 - val_loss: 0.2080
Epoch 5/50
514/514 ━━━━━━━━━━━━━━━━━━━━ 6s 12ms/step - loss: 0.2013 - val_loss: 0.2077
Epoch 6/50
514/514 ━━━━━━━━━━━━━━━━━━━━ 9s 9ms/step - loss: 0.2012 - val_loss: 0.2076
Epoch 7/50
514/514 ━━━━━━━━━━━━━━━━━━━━ 7s 12ms/step - loss: 0.2013 - val_loss: 0.2076
Epoch 8/50
514/514 ━━━━━━━━━━━━━━━━━━━━ 10s 12ms/step - loss: 0.2013 - val_loss: 0.2074
Epoch 9/50
514/514 ━━━━━━━━━━━━━━━━━━━━ 9s 9ms/step - loss: 0.2010 - val_loss: 0.2076
Epoch 10/50
514/514 ━━━━━━━━━━━━━━━━━━━━ 7s 12ms/step - loss: 0.2012 - val_loss: 0.2074
Epoch 11/50
514/514 ━━━━━━━━━━━━━━━━━━━━ 9s 9ms/step - loss: 0.2014 - val_loss: 0.2071
Epoch 12/50
514/514 ━━━━━━━━━━━━━━━━━━━━ 7s 12ms/step - loss: 0.2011 - val_loss: 0.2070
Epoch 13/50
514/514 ━━━━━━━━━━━━━━━━━━━━ 9s 9ms/step - loss: 0.2010 - val_loss: 0.2068
Epoch 14/50
514/514 ━━━━━━━━━━━━━━━━━━━━ 7s 14ms/step - loss: 0.2011 - val_loss: 0.2067
Epoch 15/50
514/514 ━━━━━━━━━━━━━━━━━━━━ 9s 12ms/step - loss: 0.2012 - val_loss: 0.2066
Epoch 16/50
514/514 ━━━━━━━━━━━━━━━━━━━━ 7s 13ms/step - loss: 0.2011 - val_loss: 0.2067
Epoch 17/50
514/514 ━━━━━━━━━━━━━━━━━━━━ 10s 12ms/step - loss: 0.2011 - val_loss: 0.2066
Epoch 18/50
514/514 ━━━━━━━━━━━━━━━━━━━━ 6s 12ms/step - loss: 0.2011 - val_loss: 0.2067
Epoch 19/50
514/514 ━━━━━━━━━━━━━━━━━━━━ 11s 13ms/step - loss: 0.2009 - val_loss: 0.2068
Epoch 20/50
514/514 ━━━━━━━━━━━━━━━━━━━━ 6s 12ms/step - loss: 0.2010 - val_loss: 0.2069
Epoch 21/50
514/514 ━━━━━━━━━━━━━━━━━━━━ 9s 9ms/step - loss: 0.2011 - val_loss: 0.2069
Epoch 22/50
514/514 ━━━━━━━━━━━━━━━━━━━━ 6s 12ms/step - loss: 0.2009 - val_loss: 0.2069
Epoch 23/50
514/514 ━━━━━━━━━━━━━━━━━━━━ 6s 12ms/step - loss: 0.2013 - val_loss: 0.2069
Epoch 24/50
514/514 ━━━━━━━━━━━━━━━━━━━━ 11s 13ms/step - loss: 0.2011 - val_loss: 0.2069
Epoch 25/50
514/514 ━━━━━━━━━━━━━━━━━━━━ 8s 9ms/step - loss: 0.2011 - val_loss: 0.2068
Epoch 26/50
514/514 ━━━━━━━━━━━━━━━━━━━━ 7s 13ms/step - loss: 0.2012 - val_loss: 0.2068
Epoch 27/50
514/514 ━━━━━━━━━━━━━━━━━━━━ 6s 12ms/step - loss: 0.2009 - val_loss: 0.2069
Epoch 28/50
514/514 ━━━━━━━━━━━━━━━━━━━━ 7s 13ms/step - loss: 0.2013 - val_loss: 0.2069
```

```
Epoch 29/50
514/514 ━━━━━━━━━━━━━━━━━━━━ 10s 12ms/step - loss: 0.2010 - val_loss: 0.2069
Epoch 30/50
514/514 ━━━━━━━━━━━━━━━━━━━━ 5s 9ms/step - loss: 0.2008 - val_loss: 0.2069
Epoch 31/50
514/514 ━━━━━━━━━━━━━━━━━━━━ 5s 9ms/step - loss: 0.2010 - val_loss: 0.2069
Epoch 32/50
514/514 ━━━━━━━━━━━━━━━━━━━━ 7s 13ms/step - loss: 0.2009 - val_loss: 0.2070
Epoch 33/50
514/514 ━━━━━━━━━━━━━━━━━━━━ 10s 12ms/step - loss: 0.2010 - val_loss: 0.2070
Epoch 34/50
514/514 ━━━━━━━━━━━━━━━━━━━━ 9s 9ms/step - loss: 0.2009 - val_loss: 0.2071
Epoch 35/50
514/514 ━━━━━━━━━━━━━━━━━━━━ 7s 13ms/step - loss: 0.2009 - val_loss: 0.2069
Epoch 36/50
514/514 ━━━━━━━━━━━━━━━━━━━━ 8s 9ms/step - loss: 0.2009 - val_loss: 0.2070
Epoch 37/50
514/514 ━━━━━━━━━━━━━━━━━━━━ 6s 12ms/step - loss: 0.2010 - val_loss: 0.2070
Epoch 38/50
514/514 ━━━━━━━━━━━━━━━━━━━━ 9s 9ms/step - loss: 0.2010 - val_loss: 0.2071
Epoch 39/50
514/514 ━━━━━━━━━━━━━━━━━━━━ 6s 12ms/step - loss: 0.2012 - val_loss: 0.2071
Epoch 40/50
514/514 ━━━━━━━━━━━━━━━━━━━━ 10s 12ms/step - loss: 0.2010 - val_loss: 0.2070
Epoch 41/50
514/514 ━━━━━━━━━━━━━━━━━━━━ 6s 12ms/step - loss: 0.2008 - val_loss: 0.2072
Epoch 42/50
514/514 ━━━━━━━━━━━━━━━━━━━━ 11s 13ms/step - loss: 0.2009 - val_loss: 0.2071
Epoch 43/50
514/514 ━━━━━━━━━━━━━━━━━━━━ 8s 9ms/step - loss: 0.2010 - val_loss: 0.2071
Epoch 44/50
514/514 ━━━━━━━━━━━━━━━━━━━━ 7s 13ms/step - loss: 0.2010 - val_loss: 0.2073
Epoch 45/50
514/514 ━━━━━━━━━━━━━━━━━━━━ 5s 9ms/step - loss: 0.2009 - val_loss: 0.2072
Epoch 46/50
514/514 ━━━━━━━━━━━━━━━━━━━━ 6s 12ms/step - loss: 0.2010 - val_loss: 0.2072
Epoch 47/50
514/514 ━━━━━━━━━━━━━━━━━━━━ 10s 12ms/step - loss: 0.2010 - val_loss: 0.2072
Epoch 48/50
514/514 ━━━━━━━━━━━━━━━━━━━━ 6s 11ms/step - loss: 0.2009 - val_loss: 0.2074
Epoch 49/50
514/514 ━━━━━━━━━━━━━━━━━━━━ 10s 11ms/step - loss: 0.2008 - val_loss: 0.2074
Epoch 50/50
514/514 ━━━━━━━━━━━━━━━━━━━━ 9s 9ms/step - loss: 0.2009 - val_loss: 0.2074
```

Out[20]:
```
<keras.src.callbacks.history.History at 0x7bc9f70e5270>
```

In [28]:
```
test_reconstructions = autoencoder.predict(X_test)
test_errors = np.mean(np.square(X_test - test_reconstructions), axis=1)

anomaly_predictions = (test_errors > error_threshold).astype(int)
```

```
26320/26320 ━━━━━━━━━━━━━━━━━━━━ 38s 1ms/step
```

In [29]:
```
anomaly_predictions
```

Out[29]:
```
array([1, 1, 1, ..., 1, 1, 1])
```

In [30]:
```
from sklearn.metrics import accuracy_score, precision_score, recall_score, roc_auc_

# Compute metrics
accuracy = accuracy_score(y_test, anomaly_predictions)
precision = precision_score(y_test, anomaly_predictions)
recall = recall_score(y_test, anomaly_predictions)
roc_auc = roc_auc_score(y_test, anomaly_predictions)

# Print results
print(f"Accuracy: {accuracy:.4f}")
print(f"Precision: {precision:.4f}")
print(f"Recall: {recall:.4f}")
print(f"AUC: {roc_auc:.4f}")
```

```
Accuracy: 0.0019
Precision: 0.0019
Recall: 1.0000
AUC: 0.5000
```

Elliptic Envelope

In [ ]:
```
from sklearn.covariance import EllipticEnvelope

# Fit the Elliptic Envelope model
envelope = EllipticEnvelope(contamination=0.05)
envelope.fit(X_train)
```

In [101…
```
# Predict anomalies
y_pred = envelope.predict(X_test)

# Convert predictions to 0 for normal and 1 for anomaly
y_pred = (y_pred == -1).astype(int)
```

In [5]:
```
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)
roc_auc = roc_auc_score(y_test, y_pred)

# Print results
print(f"Accuracy: {accuracy:.4f}")
print(f"Precision: {precision:.4f}")
print(f"Recall: {recall:.4f}")
print(f"AUC: {roc_auc:.4f}")
```

```
Accuracy: 0.9247
Precision: 0.5925
Recall: 0.8773
AUC: 0.8129
```

Final Model Selected - Elliptic Envelope

In [ ]:
```
import pickle

file = "final_submission_model_elliptic_envelope.pkl"
with open(filename, 'wb') as file:
    pickle.dump(envelope, file)
```