Importing required Python modules

In [1]:
```
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 = "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	c
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 [3]:
```
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 [4]:
```
pd.DataFrame(df.dtypes, columns=["DataType"])
```

Out[4]:

	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 [5]:
```
print(f"The number of rows in the dataset having isFlaggedFraud=0 are {df['
print(f"The number of rows in the dataset having isFlaggedFraud=1 are {df['
```

```
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 [6]:
```
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 [7]:
```
print(f"The minimum amount in the dataset within the non flagged transactior
print(f"The minimum amount in the dataset within the flagged transactions su
```

```
The minimum amount in the dataset within the non flagged transactions subs
et is 0.0
The minimum amount in the dataset within the flagged transactions subset i
s 353874.22
```

In [8]:
```
print(f"The maximum amount in the dataset within the non flagged transactior
print(f"The maximum amount in the dataset within the flagged transactions su
```

```
The maximum amount in the dataset within the non flagged transactions subs
et is 92445516.64
The maximum amount in the dataset within the flagged transactions subset i
s 10000000.0
```

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

In [9]:
```
flagged_frauds_df
```

Out[9]:

	step	type	amount	nameOrig	oldbalanceOrg	newbalanceOrig	na
2736446	212	TRANSFER	4953893.08	C728984460	4953893.08	4953893.08	C63
3247297	250	TRANSFER	1343002.08	C1100582606	1343002.08	1343002.08	C114
3760288	279	TRANSFER	536624.41	C1035541766	536624.41	536624.41	C110
5563713	387	TRANSFER	4892193.09	C908544136	4892193.09	4892193.09	C89
5996407	425	TRANSFER	10000000.00	C689608084	19585040.37	19585040.37	C139
5996409	425	TRANSFER	9585040.37	C452586515	19585040.37	19585040.37	C110
6168499	554	TRANSFER	3576297.10	C193696150	3576297.10	3576297.10	C48
6205439	586	TRANSFER	353874.22	C1684585475	353874.22	353874.22	C177
6266413	617	TRANSFER	2542664.27	C786455622	2542664.27	2542664.27	C661
6281482	646	TRANSFER	10000000.00	C19004745	10399045.08	10399045.08	C180
6281484	646	TRANSFER	399045.08	C724693370	10399045.08	10399045.08	C190
6296014	671	TRANSFER	3441041.46	C917414431	3441041.46	3441041.46	C108
6351225	702	TRANSFER	3171085.59	C1892216157	3171085.59	3171085.59	C130
6362460	730	TRANSFER	10000000.00	C2140038573	17316255.05	17316255.05	C139
6362462	730	TRANSFER	7316255.05	C1869569059	17316255.05	17316255.05	C186
6362584	741	TRANSFER	5674547.89	C992223106	5674547.89	5674547.89	C136

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 [10]:
```
df = df[df['isFlaggedFraud'] !=1]
df
```

Out[10]:

	step	type	amount	nameOrig	oldbalanceOrg	newbalanceOrig	nan
0	1	PAYMENT	9839.64	C1231006815	170136.00	160296.36	M1979
1	1	PAYMENT	1864.28	C1666544295	21249.00	19384.72	M2044
2	1	TRANSFER	181.00	C1305486145	181.00	0.00	C553
3	1	CASH_OUT	181.00	C840083671	181.00	0.00	C38
4	1	PAYMENT	11668.14	C2048537720	41554.00	29885.86	M1230
...	...	...	...	...	...	...	
6362615	743	CASH_OUT	339682.13	C786484425	339682.13	0.00	C776
6362616	743	TRANSFER	6311409.28	C1529008245	6311409.28	0.00	C1881
6362617	743	CASH_OUT	6311409.28	C1162922333	6311409.28	0.00	C1365
6362618	743	TRANSFER	850002.52	C1685995037	850002.52	0.00	C2080
6362619	743	CASH_OUT	850002.52	C1280323807	850002.52	0.00	C873

6362604 rows × 11 columns

In [11]:
```
not_frauds_df = pd.DataFrame(df[df['isFraud']==0])
frauds_df = pd.DataFrame(df[df['isFraud']==1])
```

In [12]:
```
df['orgCustomerType'] = df['nameOrig'].str[0]
df['destCustomerType'] = df['nameDest'].str[0]
```

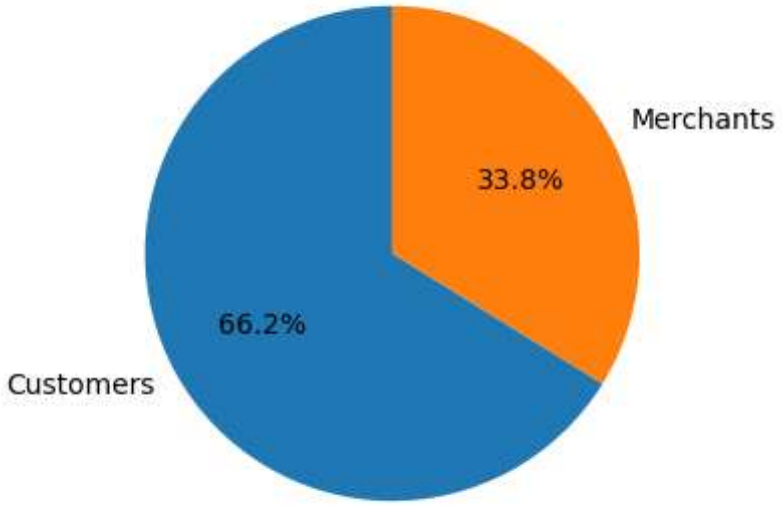
In [13]:
```
df['orgCustomerType'].value_counts()
```

Out[13]:
```
C    6362604
Name: orgCustomerType, dtype: int64
```

In [14]:
```
df['destCustomerType'].value_counts()
```

Out[14]:
```
C    4211109
M    2151495
Name: destCustomerType, dtype: int64
```

In [15]:
```
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()
```



In [16]:
```
frauds_df['orgCustomerType'] = frauds_df['nameOrig'].str[0]
frauds_df['destCustomerType'] = frauds_df['nameDest'].str[0]

frauds_df
```

Out[16]:

	step	type	amount	nameOrig	oldbalanceOrg	newbalanceOrig	nam
2	1	TRANSFER	181.00	C1305486145	181.00	0.0	C5532
3	1	CASH_OUT	181.00	C840083671	181.00	0.0	C389
251	1	TRANSFER	2806.00	C1420196421	2806.00	0.0	C9727
252	1	CASH_OUT	2806.00	C2101527076	2806.00	0.0	C10072
680	1	TRANSFER	20128.00	C137533655	20128.00	0.0	C18484
...	...	...	...	...	...	...	
6362615	743	CASH_OUT	339682.13	C786484425	339682.13	0.0	C7769
6362616	743	TRANSFER	6311409.28	C1529008245	6311409.28	0.0	C18818
6362617	743	CASH_OUT	6311409.28	C1162922333	6311409.28	0.0	C13651
6362618	743	TRANSFER	850002.52	C1685995037	850002.52	0.0	C20803
6362619	743	CASH_OUT	850002.52	C1280323807	850002.52	0.0	C8732

8197 rows × 13 columns

In [17]:
```
frauds_df['orgCustomerType'].value_counts()
```

Out[17]:
```
C    8197
Name: orgCustomerType, dtype: int64
```

In [18]:
```
frauds_df['destCustomerType'].value_counts()
```

Out[18]:
```
C    8197
Name: destCustomerType, dtype: int64
```

In [19]:
```
not_frauds_df['orgCustomerType'] = not_frauds_df['nameOrig'].str[0]
not_frauds_df['destCustomerType'] = not_frauds_df['nameDest'].str[0]

not_frauds_df
```

Out[19]:

	step	type	amount	nameOrig	oldbalanceOrg	newbalanceOrig	nam
0	1	PAYMENT	9839.64	C1231006815	170136.0	160296.36	M19797
1	1	PAYMENT	1864.28	C1666544295	21249.0	19384.72	M20442
4	1	PAYMENT	11668.14	C2048537720	41554.0	29885.86	M12307
5	1	PAYMENT	7817.71	C90045638	53860.0	46042.29	M5734
6	1	PAYMENT	7107.77	C154988899	183195.0	176087.23	M4080
...	...	...	...	...	...	...	
6362319	718	PAYMENT	8634.29	C642813806	518802.0	510167.71	M7477
6362320	718	CASH_OUT	159188.22	C691808084	3859.0	0.00	C18181
6362321	718	CASH_OUT	186273.84	C102120699	168046.0	0.00	C15156
6362322	718	TRANSFER	82096.45	C614459560	13492.0	0.00	C8553
6362323	718	DEBIT	1864.24	C49652609	20426.0	18561.76	C17990

6354407 rows × 13 columns

In [20]:
```
not_frauds_df['orgCustomerType'].value_counts()
```

Out[20]:
```
C    6354407
Name: orgCustomerType, dtype: int64
```

In [21]:
```
not_frauds_df['destCustomerType'].value_counts()
```

Out[21]:
```
C    4202912
M    2151495
Name: destCustomerType, dtype: int64
```

In [22]:
```
not_frauds_df = not_frauds_df[not_frauds_df['destCustomerType'] != "M"]
not_frauds_df
```

Out[22]:

	step	type	amount	nameOrig	oldbalanceOrg	newbalanceOrig	name
9	1	DEBIT	5337.77	C712410124	41720.0	36382.23	C1956(
10	1	DEBIT	9644.94	C1900366749	4465.0	0.00	C9976(
15	1	CASH_OUT	229133.94	C905080434	15325.0	0.00	C4764(
19	1	TRANSFER	215310.30	C1670993182	705.0	0.00	C11004:
21	1	DEBIT	9302.79	C1566511282	11299.0	1996.21	C19735:
...	...	...	...	...	...	...	
6362317	718	CASH_OUT	317177.48	C857156502	170.0	0.00	C7841(
6362320	718	CASH_OUT	159188.22	C691808084	3859.0	0.00	C18181{
6362321	718	CASH_OUT	186273.84	C102120699	168046.0	0.00	C15156:
6362322	718	TRANSFER	82096.45	C614459560	13492.0	0.00	C8553{
6362323	718	DEBIT	1864.24	C49652609	20426.0	18561.76	C17990(

4202912 rows × 13 columns

In [23]:
```
frauds_df['diff_new_bals'] = frauds_df['newbalanceDest'] - frauds_df['oldba
not_frauds_df['diff_new_bals'] = not_frauds_df['newbalanceDest'] - not_frau

print(f"The average value of diffs in Non-Fraudulent transactions are : {no
print(f"The average value of diffs in Fraudulent transactions are : {frauds_
```

```
The average value of diffs in Non-Fraudulent transactions are : 186727.637
68504516
The average value of diffs in Fraudulent transactions are : 736893.5632743
661

C:\Users\mcsan\AppData\Local\Temp\ipykernel_3496\1740282389.py:2: SettingW
ithCopyWarning:
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-doc
s/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (https://
pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-
view-versus-a-copy)
  not_frauds_df['diff_new_bals'] = not_frauds_df['newbalanceDest'] - not_f
rauds_df['oldbalanceDest']
```

In [24]:
```
frauds_df['diff_new_bals'] = frauds_df['newbalanceDest'] - frauds_df['newba
not_frauds_df['diff_new_bals'] = not_frauds_df['newbalanceDest'] - not_frau

print(f"The average value of diffs in Non-Fraudulent transactions are : {not
print(f"The average value of diffs in Fraudulent transactions are : {frauds_
```

```
The average value of diffs in Non-Fraudulent transactions are : 589481.066
5288476
The average value of diffs in Fraudulent transactions are : 1104697.313473
2218

C:\Users\mcsan\AppData\Local\Temp\ipykernel_3496\470572971.py:2: SettingWi
thCopyWarning:
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-doc
s/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (https://
pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-
view-versus-a-copy)
  not_frauds_df['diff_new_bals'] = not_frauds_df['newbalanceDest'] - not_f
rauds_df['newbalanceOrig']
```

In [25]:
```
frauds_df['diff_new_bals'] = frauds_df['newbalanceOrig'] - frauds_df['oldba
not_frauds_df['diff_new_bals'] = not_frauds_df['newbalanceOrig'] - not_frau

print(f"The average value of diffs in Non-Fraudulent transactions are : {not
print(f"The average value of diffs in Fraudulent transactions are : {frauds_
```

```
The average value of diffs in Non-Fraudulent transactions are : 38253.2073
0465675
The average value of diffs in Fraudulent transactions are : -1460119.47791
14309

C:\Users\mcsan\AppData\Local\Temp\ipykernel_3496\632296387.py:2: SettingWi
thCopyWarning:
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-doc
s/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (https://
pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-
view-versus-a-copy)
  not_frauds_df['diff_new_bals'] = not_frauds_df['newbalanceOrig'] - not_f
rauds_df['oldbalanceOrg']
```

In [ ]:
```

```

Now observing the shape of updated dataframe

In [26]:
```
print(f"This dataset has {df.shape[0]} rows after removing the FlaggedFraud
print(f"This dataset has {df.shape[1]} columns after removing the FlaggedFra
```

```
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 [27]:
```
print(f"The number of rows in the dataset having isFraud=0 are {df['isFraud
print(f"The number of rows in the dataset having isFraud=1 are {df['isFraud
```

```
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 [28]:
```
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 [29]:
```
print(f"The unique values of TYPE column in Non-Fradulent transactions are
print(f"The unique values of TYPE column in Fradulent transactions are : {fr
```

```
The unique values of TYPE column in Non-Fradulent transactions are : ['PAY
MENT' 'DEBIT' 'CASH_OUT' 'TRANSFER' 'CASH_IN']
The unique values of TYPE column in Fradulent transactions are : ['TRANSFE
R' 'CASH_OUT']
```

2D. Observing the amount stats in both cases

In [30]:
```
print(f"The average value of executed transactions in Non-Fraudulent transac
print(f"The average value of executed transactions in Fraudulent transactior
```

```
The average value of executed transactions in Non-Fraudulent transactions
are : 178197.04172739814
The average value of executed transactions in Fraudulent transactions are
: 1461343.1577589358
```

In [31]:
```
print(f"The maximum value of executed transactions in Non-Fraudulent transac
print(f"The maximum value of executed transactions in Fraudulent transactior
```

```
The maximum value of executed transactions in Non-Fraudulent transactions
are : 92445516.64
The maximum value of executed transactions in Fraudulent transactions are
: 10000000.0
```

In [32]:
```
print(f"The minimum value of executed transactions in Non-Fraudulent transac
print(f"The minimum value of executed transactions in Fraudulent transactior
```

```
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 [33]:
```
print(f"The standard deviation between executed transactions in Non-Fraudule
print(f"The standard deviation between executed transactions in Fraudulent t
```

```
The standard deviation between executed transactions in Non-Fraudulent tra
nsactions are : 596236.9813471739
The standard deviation between executed transactions in Fraudulent transac
tions are : 2397046.563628229
```

In [34]:
```
print(f"The coefficient of variance of executed transactions in Non-Fraudule
print(f"The coefficient of variance of executed transactions in Fraudulent t
```

```
The coefficient of variance of executed transactions in Non-Fraudulent tra
nsactions are : 3.3459420850503454
The coefficient of variance of executed transactions in Fraudulent transac
tions are : 1.6403036828832556
```

2D. Observing the oldbalanceOrg stats in both cases

In [35]:
```
print(f"The average value of oldbalanceOrg in Non-Fraudulent transactions ar
print(f"The average value of oldbalanceOrg in Fraudulent transactions are :
```

```
The average value of oldbalanceOrg in Non-Fraudulent transactions are : 83
2828.7117272562
The average value of oldbalanceOrg in Fraudulent transactions are : 163762
7.68592412
```

In [36]:
```
print(f"The maximum value of oldbalanceOrg in Non-Fraudulent transactions ar
print(f"The maximum value of oldbalanceOrg in Fraudulent transactions are :
```

```
The maximum value of oldbalanceOrg in Non-Fraudulent transactions are : 43
818855.3
The maximum value of oldbalanceOrg in Fraudulent transactions are : 595850
40.37
```

In [37]:
```
print(f"The minimum value of oldbalanceOrg in Non-Fraudulent transactions ar
print(f"The minimum value of oldbalanceOrg in Fraudulent transactions are :
```

```
The minimum value of oldbalanceOrg in Non-Fraudulent transactions are : 0.
0
The minimum value of oldbalanceOrg in Fraudulent transactions are : 0.0
```

In [38]:
```
print(f"The standard deviation between oldbalanceOrg in Non-Fraudulent trans
print(f"The standard deviation between oldbalanceOrg in Fraudulent transacti
```

```
The standard deviation between oldbalanceOrg in Non-Fraudulent transaction
s are : 2887144.03036968
The standard deviation between oldbalanceOrg in Fraudulent transactions ar
e : 3528099.5182469995
```

In [39]:
```
print(f"The coefficient of variance of oldbalanceOrg in Non-Fraudulent trans
print(f"The coefficient of variance of oldbalanceOrg in Fraudulent transacti
```

```
The coefficient of variance of oldbalanceOrg in Non-Fraudulent transaction
s are : 3.4666720655941954
The coefficient of variance of oldbalanceOrg in Fraudulent transactions ar
e : 2.1543965997717476
```

2E. Observing the newbalanceOrig stats in both cases

In [40]:
```
print(f"The average value of newbalanceOrig in Non-Fraudulent transactions a
print(f"The average value of newbalanceOrig in Fraudulent transactions are
```

```
The average value of newbalanceOrig in Non-Fraudulent transactions are : 8
55970.2281087907
The average value of newbalanceOrig in Fraudulent transactions are : 17750
8.20801268757
```

In [41]:
```
print(f"The maximum value of newbalanceOrig in Non-Fraudulent transactions a
print(f"The maximum value of newbalanceOrig in Fraudulent transactions are
```

```
The maximum value of newbalanceOrig in Non-Fraudulent transactions are : 4
3686616.33
The maximum value of newbalanceOrig in Fraudulent transactions are : 49585
040.37
```

In [42]:
```
print(f"The minimum value of newbalanceOrig in Non-Fraudulent transactions a
print(f"The minimum value of newbalanceOrig in Fraudulent transactions are
```

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

In [43]:
```
print(f"The standard deviation between newbalanceOrig in Non-Fraudulent trar
print(f"The standard deviation between newbalanceOrig in Fraudulent transact
```

```
The standard deviation between newbalanceOrig in Non-Fraudulent transactio
ns are : 2924986.9646518226
The standard deviation between newbalanceOrig in Fraudulent transactions a
re : 1915377.846506979
```

In [44]:
```
print(f"The coefficient of variance of newbalanceOrig in Non-Fraudulent tran
print(f"The coefficient of variance of newbalanceOrig in Fraudulent transact
```

```
The coefficient of variance of newbalanceOrig in Non-Fraudulent transactio
ns are : 3.4171596962132513
The coefficient of variance of newbalanceOrig in Fraudulent transactions a
re : 10.790362135648824
```

2F. Observing the oldbalanceDest stats in both cases

In [45]:
```
print(f"The average value of oldbalanceDest in Non-Fraudulent transactions a
print(f"The average value of oldbalanceDest in Fraudulent transactions are
```

```
The average value of oldbalanceDest in Non-Fraudulent transactions are : 1
101420.8745694289
The average value of oldbalanceDest in Fraudulent transactions are : 54531
1.9582115407
```

In [46]:
```
print(f"The maximum value of oldbalanceDest in Non-Fraudulent transactions a
print(f"The maximum value of oldbalanceDest in Fraudulent transactions are
```

```
The maximum value of oldbalanceDest in Non-Fraudulent transactions are : 3
56015889.35
The maximum value of oldbalanceDest in Fraudulent transactions are : 23623
0516.82
```

In [47]:
```
print(f"The minimum value of oldbalanceDest in Non-Fraudulent transactions a
print(f"The minimum value of oldbalanceDest in Fraudulent transactions are
```

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

In [48]:
```
print(f"The standard deviation between oldbalanceDest in Non-Fraudulent tran
print(f"The standard deviation between oldbalanceDest in Fraudulent transact
```

```
The standard deviation between oldbalanceDest in Non-Fraudulent transactio
ns are : 3399201.7933451207
The standard deviation between oldbalanceDest in Fraudulent transactions a
re : 3339589.2539171097
```

In [49]:
```
print(f"The coefficient of variance of oldbalanceDest in Non-Fraudulent tran
print(f"The coefficient of variance of oldbalanceDest in Fraudulent transact
```

```
The coefficient of variance of oldbalanceDest in Non-Fraudulent transactio
ns are : 3.0861969950169574
The coefficient of variance of oldbalanceDest in Fraudulent transactions a
re : 6.12418122072723
```

2G. Observing the newbalanceDest stats in both cases

In [50]:
```
print(f"The average value of newbalanceDest in Non-Fraudulent transactions a
print(f"The average value of newbalanceDest in Fraudulent transactions are
```

```
The average value of newbalanceDest in Non-Fraudulent transactions are : 1
224925.6845633048
The average value of newbalanceDest in Fraudulent transactions are : 12822
05.521485908
```

In [51]:
```
print(f"The maximum value of newbalanceDest in Non-Fraudulent transactions a
print(f"The maximum value of newbalanceDest in Fraudulent transactions are
```

```
The maximum value of newbalanceDest in Non-Fraudulent transactions are : 3
56179278.92
The maximum value of newbalanceDest in Fraudulent transactions are : 23672
6494.66
```

In [52]:
```
print(f"The minimum value of newbalanceDest in Non-Fraudulent transactions a
print(f"The minimum value of newbalanceDest in Fraudulent transactions are
```

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

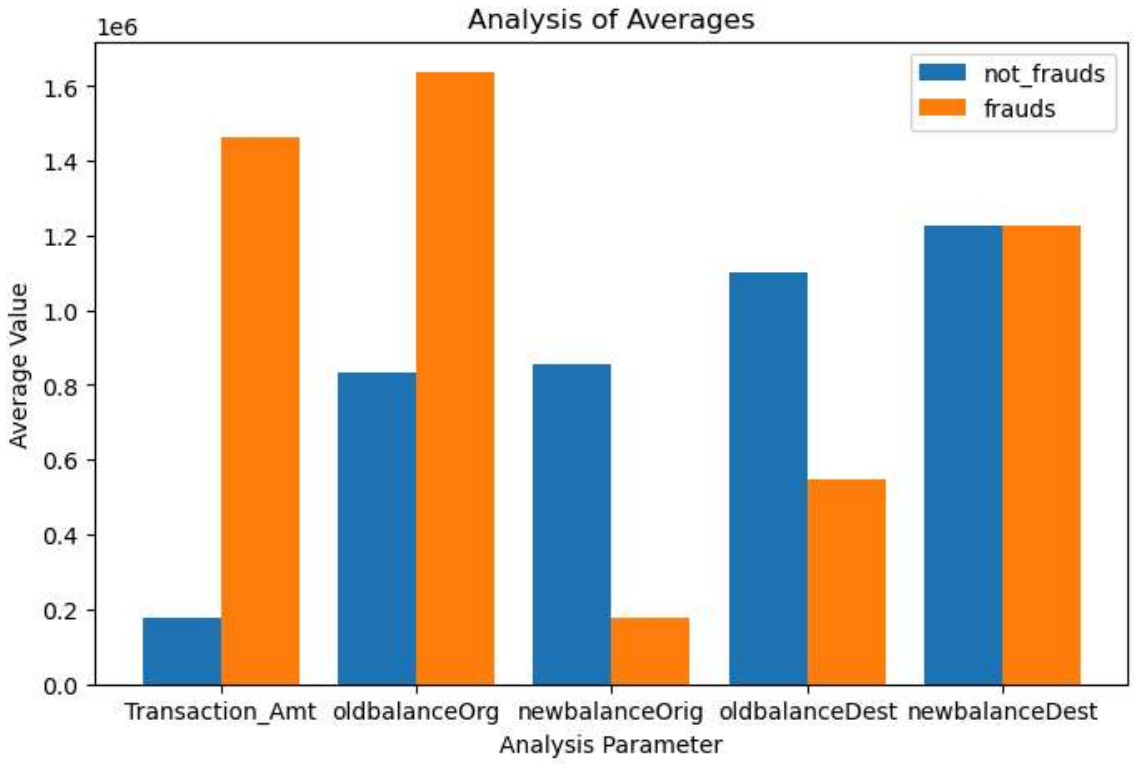
In [53]:
```
print(f"The standard deviation between newbalanceDest in Non-Fraudulent trar
print(f"The standard deviation between newbalanceDest in Fraudulent transact
```

```
The standard deviation between newbalanceDest in Non-Fraudulent transactio
ns are : 3673815.709962235
The standard deviation between newbalanceDest in Fraudulent transactions a
re : 3912220.6495843115
```

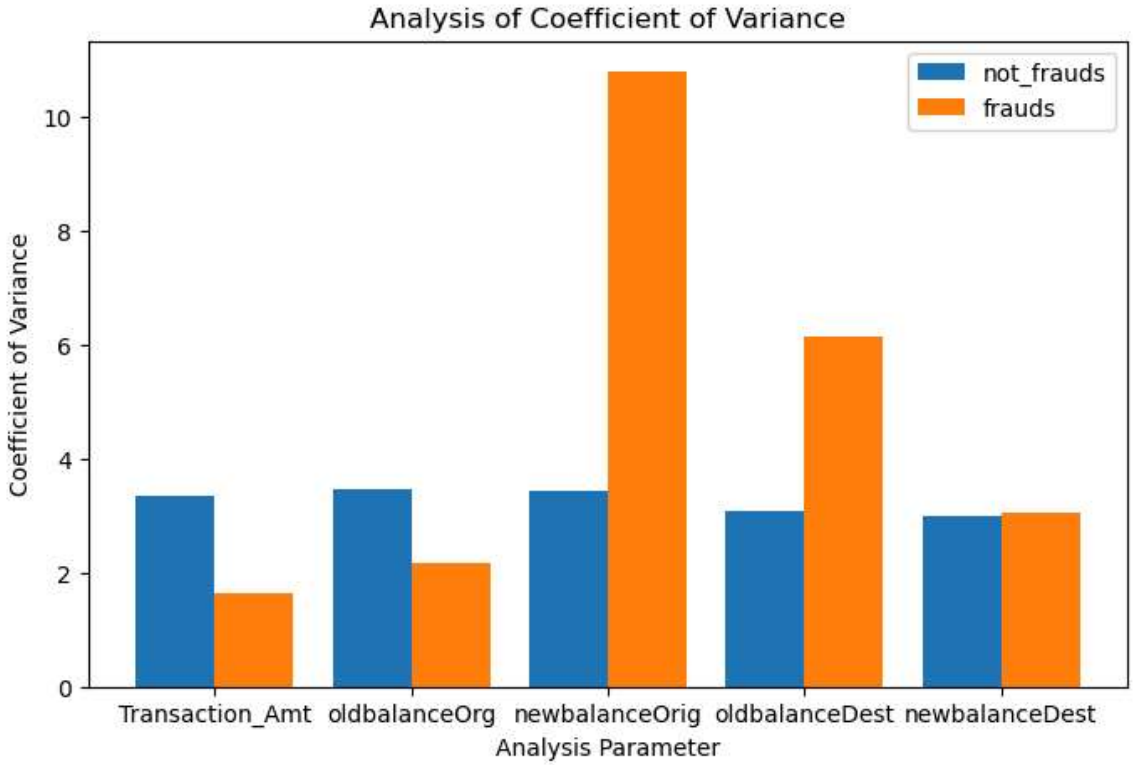
In [54]:
```
print(f"The coefficient of variance of newbalanceDest in Non-Fraudulent trar
print(f"The coefficient of variance of newbalanceDest in Fraudulent transact
```

```
The coefficient of variance of newbalanceDest in Non-Fraudulent transactio
ns are : 2.9992151819985535
The coefficient of variance of newbalanceDest in Fraudulent transactions a
re : 3.0511650308996963
```

In [55]:
```
X = ['Transaction_Amt','oldbalanceOrg','newbalanceOrig','oldbalanceDest', 'n
not_frauds = [not_frauds_df['amount'].mean(), not_frauds_df['oldbalanceOrg']
frauds = [frauds_df['amount'].mean(),frauds_df['oldbalanceOrg'].mean(),frauc

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 [56]:
```
X = ['Transaction_Amt','oldbalanceOrg','newbalanceOrig','oldbalanceDest', 'n
not_frauds = [not_frauds_df['amount'].std()/not_frauds_df['amount'].mean(),
              not_frauds_df['oldbalanceOrg'].std()/not_frauds_df['oldbalance
              not_frauds_df['newbalanceOrig'].std()/not_frauds_df['newbalan
              not_frauds_df['oldbalanceDest'].std()/not_frauds_df['oldbalan
              not_frauds_df['newbalanceDest'].std()/not_frauds_df['newbalan
             ]
frauds = [frauds_df['amount'].std()/frauds_df['amount'].mean(),
              frauds_df['oldbalanceOrg'].std()/frauds_df['oldbalanceOrg'].me
              frauds_df['newbalanceOrig'].std()/frauds_df['newbalanceOrig']
              frauds_df['oldbalanceDest'].std()/frauds_df['oldbalanceDest']
              frauds_df['newbalanceDest'].std()/frauds_df['newbalanceDest']
             ]

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 [57]:
```
frauds_df
```

Out[57]:

	step	type	amount	nameOrig	oldbalanceOrg	newbalanceOrig	nam
2	1	TRANSFER	181.00	C1305486145	181.00	0.0	C5532
3	1	CASH_OUT	181.00	C840083671	181.00	0.0	C389
251	1	TRANSFER	2806.00	C1420196421	2806.00	0.0	C9727
252	1	CASH_OUT	2806.00	C2101527076	2806.00	0.0	C10072
680	1	TRANSFER	20128.00	C137533655	20128.00	0.0	C18484
...	...	...	...	...	...	...	
6362615	743	CASH_OUT	339682.13	C786484425	339682.13	0.0	C7769
6362616	743	TRANSFER	6311409.28	C1529008245	6311409.28	0.0	C18818
6362617	743	CASH_OUT	6311409.28	C1162922333	6311409.28	0.0	C13651
6362618	743	TRANSFER	850002.52	C1685995037	850002.52	0.0	C20803
6362619	743	CASH_OUT	850002.52	C1280323807	850002.52	0.0	C8732

8197 rows × 13 columns

In [58]:
```
frauds_df.shape
```

Out[58]:
```
(8197, 13)
```

In [59]:
```
frauds_df.groupby("type").count()
```

Out[59]:

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

Data Cleaning

Removing flagged rows

In [60]:
```
df = df[df['isFlaggedFraud'] !=1]
df
```

Out[60]:

	step	type	amount	nameOrig	oldbalanceOrg	newbalanceOrig	nan
0	1	PAYMENT	9839.64	C1231006815	170136.00	160296.36	M19797
1	1	PAYMENT	1864.28	C1666544295	21249.00	19384.72	M20442
2	1	TRANSFER	181.00	C1305486145	181.00	0.00	C553
3	1	CASH_OUT	181.00	C840083671	181.00	0.00	C38
4	1	PAYMENT	11668.14	C2048537720	41554.00	29885.86	M12307
...	...	...	...	...	...	...	
6362615	743	CASH_OUT	339682.13	C786484425	339682.13	0.00	C776
6362616	743	TRANSFER	6311409.28	C1529008245	6311409.28	0.00	C1881
6362617	743	CASH_OUT	6311409.28	C1162922333	6311409.28	0.00	C1365
6362618	743	TRANSFER	850002.52	C1685995037	850002.52	0.00	C2080
6362619	743	CASH_OUT	850002.52	C1280323807	850002.52	0.00	C873

6362604 rows × 13 columns

Removing Merchants

In [61]:
```
df['destCustomerType'] = df['nameDest'].str[0]
df = df[df['destCustomerType'] != "M"]

df
```

Out[61]:

	step	type	amount	nameOrig	oldbalanceOrg	newbalanceOrig	nam
2	1	TRANSFER	181.00	C1305486145	181.00	0.00	C5532
3	1	CASH_OUT	181.00	C840083671	181.00	0.00	C389
9	1	DEBIT	5337.77	C712410124	41720.00	36382.23	C1956
10	1	DEBIT	9644.94	C1900366749	4465.00	0.00	C9976
15	1	CASH_OUT	229133.94	C905080434	15325.00	0.00	C4764
...	...	...	...	...	...	...	...
6362615	743	CASH_OUT	339682.13	C786484425	339682.13	0.00	C7769
6362616	743	TRANSFER	6311409.28	C1529008245	6311409.28	0.00	C18818
6362617	743	CASH_OUT	6311409.28	C1162922333	6311409.28	0.00	C13651
6362618	743	TRANSFER	850002.52	C1685995037	850002.52	0.00	C20803
6362619	743	CASH_OUT	850002.52	C1280323807	850002.52	0.00	C8732

4211109 rows × 13 columns

In [62]:
```
df['isFraud'].value_counts()
```

Out[62]:
```
0    4202912
1       8197
Name: isFraud, dtype: int64
```

In [63]:
```
target_rate = df['isFraud'].mean() * 100
target_rate
```

Out[63]:
```
0.19465181262228073
```

Target rate is 0.19%

Removing the aux columns

In [64]:
```
df = df.drop(['step', 'nameOrig', 'destCustomerType', 'nameDest', 'isFlagge
df
```

Out[64]:

	type	amount	oldbalanceOrg	newbalanceOrig	oldbalanceDest	newbalanc
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	403
10	DEBIT	9644.94	4465.00	0.00	10845.00	1579
15	CASH_OUT	229133.94	15325.00	0.00	5083.00	515
...	...	...	...	...	...	
6362615	CASH_OUT	339682.13	339682.13	0.00	0.00	3396
6362616	TRANSFER	6311409.28	6311409.28	0.00	0.00	
6362617	CASH_OUT	6311409.28	6311409.28	0.00	68488.84	63798
6362618	TRANSFER	850002.52	850002.52	0.00	0.00	
6362619	CASH_OUT	850002.52	850002.52	0.00	6510099.11	73601

4211109 rows × 8 columns

In [65]:
```
# 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
```

Out[65]:

	amount	oldbalanceOrg	newbalanceOrig	oldbalanceDest	newbalanceDest	isFrau
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 × 8 columns

Feature Generation

In [66]:
```
df['diff_newbalanceDest_oldbalanceDest'] = df['newbalanceDest'] - df['oldba
df['diff_newbalanceDest_newbalanceOrig'] = df['newbalanceDest'] - df['newba
df['diff_newbalanceOrig_oldbalanceOrg'] = df['newbalanceOrig'] - df['oldbal
```

In [67]:
```
df
```

Out[67]:

	amount	oldbalanceOrg	newbalanceOrig	oldbalanceDest	newbalanceDest	isFrau
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 × 11 columns

In [68]:
```
df.to_csv("frauds_feats.csv", index=False)
```

Data Splitting

In [69]:
```
df['isFraud'].value_counts()
```

Out[69]:
```
0    4202912
1       8197
Name: isFraud, dtype: int64
```

In [70]:
```
not_frauds = df[df['isFraud']==0]
frauds = df[df['isFraud']==1]
```

In [71]:
```
not_frauds_test = not_frauds.sample(frac=0.2, random_state=1576023)
not_frauds_train = not_frauds.drop(not_frauds_test.index)
```

In [72]:
```
frauds_test = frauds.sample(frac=0.2, random_state=4014)
frauds_train = frauds.drop(frauds_test.index)
```

In [73]:
```
training_df = pd.concat([not_frauds_train, frauds_train], ignore_index=True)
training_df = training_df.sample(frac=1)

training_df
```

Out[73]:

	amount	oldbalanceOrg	newbalanceOrig	oldbalanceDest	newbalanceDest	isFraud
2718711	62499.47	0.00	0.00	8879615.73	8942115.21	0
2912760	120488.82	4695253.15	4815741.97	832935.30	712446.48	0
2627675	338131.57	0.00	0.00	810907.87	1149039.44	0
2032616	55673.92	61093.00	5419.08	730701.39	786375.31	0
284121	119629.15	3010.00	0.00	0.00	119629.15	0
...	...	...	...	...	...	...
1157103	94204.77	28123.48	0.00	1198981.50	1293186.27	0
399672	403968.61	0.00	0.00	1109964.27	1513932.89	0
1552551	315944.37	3016.00	0.00	670190.49	986134.86	0
647033	62442.20	5112276.51	5174718.71	425698.75	363256.55	0
501132	28025.49	21286623.43	21314648.93	749519.08	721493.59	0

3368888 rows × 11 columns

In [74]:
```
out_of_sample_validation_df = pd.concat([not_frauds_test, frauds_test], ignc
out_of_sample_validation_df = out_of_sample_validation_df.sample(frac=1)

out_of_sample_validation_df
```

Out[74]:

	amount	oldbalanceOrg	newbalanceOrig	oldbalanceDest	newbalanceDest	isFraud
286501	154984.40	144543.45	299527.86	1244827.23	1089842.83	0
465741	102179.95	25795.00	127974.95	297922.81	195742.86	0
113041	193003.77	22712.80	0.00	265335.38	458339.15	0
202936	3076.59	958752.73	961829.32	1183682.89	1180606.30	0
277944	245568.54	0.00	0.00	1466715.44	1712283.97	0
...	...	...	...	...	...	...
151576	116319.29	17212831.21	17329150.51	8861581.51	8745262.22	0
750185	226883.90	10384.00	0.00	340540.79	567424.68	0
252233	193325.47	1442644.47	1635969.94	812593.27	619267.80	0
735351	44153.01	16217575.54	16261728.54	55431.63	11278.63	0
312566	166936.06	172335.00	5398.94	366931.90	533867.95	0

842221 rows × 11 columns

In [75]:
```
training_df.to_csv("frauds_training_data.csv", index=False)
```

In [76]:
```
out_of_sample_validation_df.to_csv("frauds_out_of_sample_validation_data.csv
```

Down Sampling

In [77]:
```
os.chdir('.')

file_path = "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[77]:

	amount	oldbalanceOrg	newbalanceOrig	oldbalanceDest	newbalanceDest	isFraud	orgC
0	62499.47	0.00	0.00	8879615.73	8942115.21	0	
1	120488.82	4695253.15	4815741.97	832935.30	712446.48	0	
2	338131.57	0.00	0.00	810907.87	1149039.44	0	
3	55673.92	61093.00	5419.08	730701.39	786375.31	0	
4	119629.15	3010.00	0.00	0.00	119629.15	0	

In [78]:
```
df.shape
```

Out[78]:
```
(3368888, 11)
```

In [79]:
```
df['isFraud'].value_counts()
```

Out[79]:
```
0    3362330
1       6558
Name: isFraud, dtype: int64
```

Downsampling the 0s for increasing the target rate

In [80]:
```
not_frauds = df[df['isFraud']==0]
frauds = df[df['isFraud']==1]
```

In [81]:
```
not_frauds = not_frauds.sample(n=125000, random_state=14896)
not_frauds
```

Out[81]:

	amount	oldbalanceOrg	newbalanceOrig	oldbalanceDest	newbalanceDest	isFrau
1390566	323939.69	10858.00	0.00	0.00	323939.69	
175577	192611.32	0.00	0.00	722270.35	914881.67	
1045878	565285.79	0.00	0.00	981717.66	1547003.45	
2284498	104756.17	0.00	0.00	4384302.49	4314677.99	
47311	388047.37	0.00	0.00	447387.03	835434.40	
...	...	...	...	...	...	
87754	62471.44	40998.00	0.00	0.00	62471.44	
377487	41084.61	10999.00	0.00	47292.43	88377.05	
3305228	85250.85	196243.99	110993.14	564357.43	649608.28	
1746307	2004963.30	0.00	0.00	2840389.44	4845352.74	
859101	494045.09	0.00	0.00	3250955.84	3266406.06	

125000 rows × 11 columns

In [82]:
```
downsampled_training_df = pd.concat([not_frauds, frauds], ignore_index=True)
downsampled_training_df = downsampled_training_df.sample(frac=1)

downsampled_training_df
```

Out[82]:

	amount	oldbalanceOrg	newbalanceOrig	oldbalanceDest	newbalanceDest	isFraud
68240	93780.30	15111903.18	15205683.49	2681083.07	2587302.77	0
9313	160256.69	0.00	0.00	367154.71	527411.40	0
85974	77417.72	90755.66	13337.95	11636.00	0.00	0
83761	270031.56	150274.49	0.00	920127.61	1190159.18	0
105789	88908.40	127553.00	216461.40	348962.04	260053.64	0
...	...	...	...	...	...	...
107306	78212.39	11403.00	89615.39	1596540.50	1518328.11	0
30166	60793.95	42710.00	0.00	270439.52	333346.03	0
54226	170843.32	0.00	0.00	200303.94	371147.26	0
121386	35681.14	9412532.10	9448213.24	682786.15	647105.02	0
83068	40489.14	16324.00	56813.14	3192642.65	3152153.50	0

131558 rows × 11 columns

In [83]:
```
downsampled_training_df.to_csv("downsampled_training_df.csv", index=False)
```

In [ ]:

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

file_path = "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[84]:

	amount	oldbalanceOrg	newbalanceOrig	oldbalanceDest	newbalanceDest	isFraud	orgC
0	93780.30	15111903.18	15205683.49	2681083.07	2587302.77	0	
1	160256.69	0.00	0.00	367154.71	527411.40	0	
2	77417.72	90755.66	13337.95	11636.00	0.00	0	
3	270031.56	150274.49	0.00	920127.61	1190159.18	0	
4	88908.40	127553.00	216461.40	348962.04	260053.64	0	

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

Out[85]:
```
(131558, 11)
```

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

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

In [87]:
```
target_rate = df['isFraud'].mean() * 100
target_rate
```

Out[87]:
```
4.984873591875827
```

In [88]:
```
features = df.drop(columns=['isFraud'])
targets = df['isFraud']
```

In [89]:
```
features
```

Out[89]:

	amount	oldbalanceOrg	newbalanceOrig	oldbalanceDest	newbalanceDest	orgCust
0	93780.30	15111903.18	15205683.49	2681083.07	2587302.77	
1	160256.69	0.00	0.00	367154.71	527411.40	
2	77417.72	90755.66	13337.95	11636.00	0.00	
3	270031.56	150274.49	0.00	920127.61	1190159.18	
4	88908.40	127553.00	216461.40	348962.04	260053.64	
...	...	...	...	...	...	
131553	78212.39	11403.00	89615.39	1596540.50	1518328.11	
131554	60793.95	42710.00	0.00	270439.52	333346.03	
131555	170843.32	0.00	0.00	200303.94	371147.26	
131556	35681.14	9412532.10	9448213.24	682786.15	647105.02	
131557	40489.14	16324.00	56813.14	3192642.65	3152153.50	

131558 rows × 10 columns

In [90]:
```
targets
```

Out[90]:
```
0         0
1         0
2         0
3         0
4         0
         ..
131553    0
131554    0
131555    0
131556    0
131557    0
Name: isFraud, Length: 131558, dtype: int64
```

Low Variance Check

In [91]:
```
data_variance = features.var()
low_variance_threshold = 0.25
low_variance_columns = data_variance[data_variance < low_variance_threshold]

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

```
Variance of each column:
 amount                                 8.264916e+11
oldbalanceOrg                          1.217013e+13
newbalanceOrig                         1.207664e+13
oldbalanceDest                         1.603929e+13
newbalanceDest                         1.838088e+13
transaction_type                       4.682594e+00
diff_newbalanceDest_oldbalanceDest     9.961237e+11
diff_newbalanceDest_newbalanceOrig     3.059086e+13
diff_newbalanceOrig_oldbalanceOrg      3.992779e+11
dtype: float64

Columns with low variance: []

C:\Users\mcsan\AppData\Local\Temp\ipykernel_3496\425552714.py:1: FutureWar
ning: Dropping of nuisance columns in DataFrame reductions (with 'numeric_
only=None') is deprecated; in a future version this will raise TypeError.
Select only valid columns before calling the reduction.
  data_variance = features.var()
```

All columns have passed the low variance check

In [ ]:
```
```

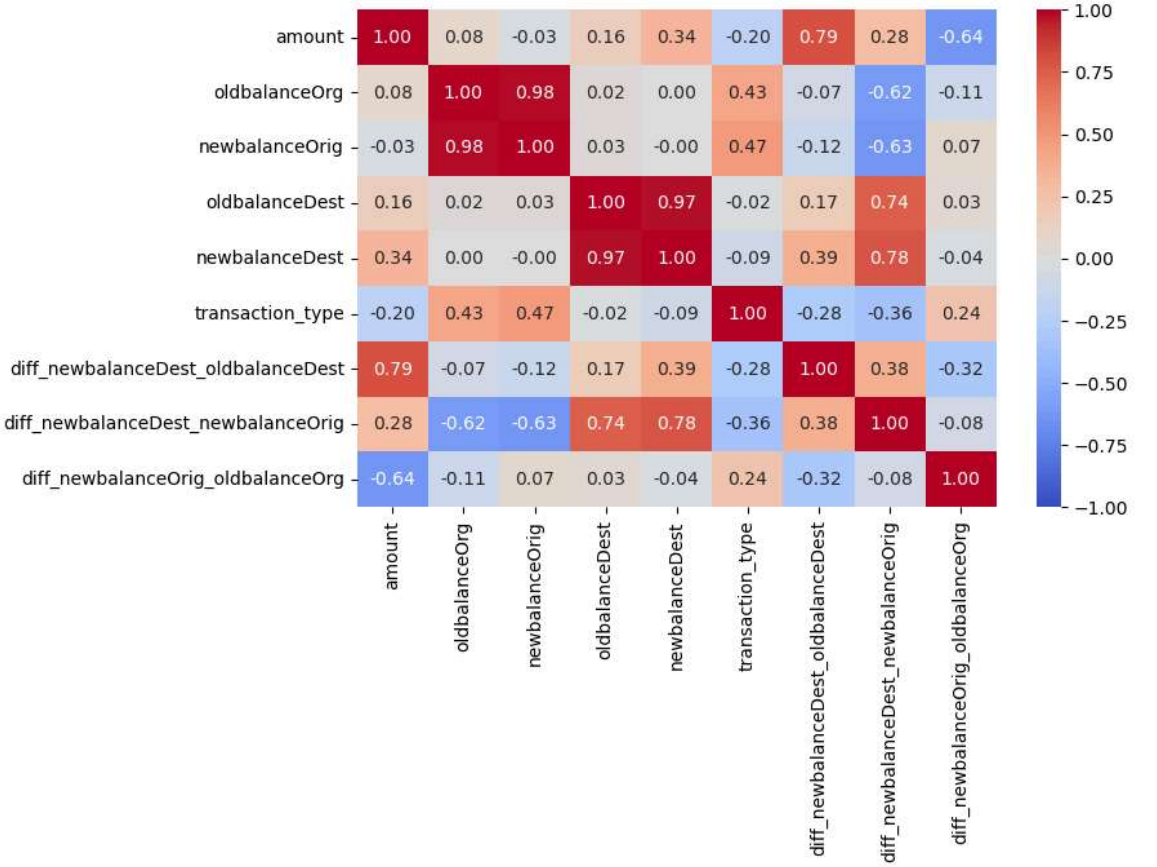
Correlation Check

In [92]:
```
correlation_matrix = features.corr()

correlation_matrix
```

Out[92]:

	amount	oldbalanceOrg	newbalanceOrig	oldbalanceDe
amount	1.000000	0.081005	-0.034227	0.1625
oldbalanceOrg	0.081005	1.000000	0.983540	0.0197
newbalanceOrig	-0.034227	0.983540	1.000000	0.0261
oldbalanceDest	0.162546	0.019727	0.026113	1.0000
newbalanceDest	0.335780	0.003230	-0.004475	0.9733
transaction_type	-0.201636	0.427816	0.473177	-0.0230
diff_newbalanceDest_oldbalanceDest	0.790139	-0.065283	-0.124005	0.1683
diff_newbalanceDest_newbalanceOrig	0.281787	-0.615469	-0.631784	0.7380
diff_newbalanceOrig_oldbalanceOrg	-0.635458	-0.111770	0.069628	0.0347

In [93]:
```
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', vmi

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



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

In [94]:
```
drop_feats = ['orgCustomerType']
features = features.drop(columns=drop_feats)

features
```

Out[94]:

	amount	oldbalanceOrg	newbalanceOrig	oldbalanceDest	newbalanceDest	transact
0	93780.30	15111903.18	15205683.49	2681083.07	2587302.77	
1	160256.69	0.00	0.00	367154.71	527411.40	
2	77417.72	90755.66	13337.95	11636.00	0.00	
3	270031.56	150274.49	0.00	920127.61	1190159.18	
4	88908.40	127553.00	216461.40	348962.04	260053.64	
...	...	...	...	...	...	
131553	78212.39	11403.00	89615.39	1596540.50	1518328.11	
131554	60793.95	42710.00	0.00	270439.52	333346.03	
131555	170843.32	0.00	0.00	200303.94	371147.26	
131556	35681.14	9412532.10	9448213.24	682786.15	647105.02	
131557	40489.14	16324.00	56813.14	3192642.65	3152153.50	

131558 rows × 9 columns

Boruta Feature Selection

In [140]:
```
from sklearn.ensemble import RandomForestClassifier
from boruta import BorutaPy

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

# 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)
```

```
Selected features: Index(['amount', 'oldbalanceOrg', 'newbalanceOrig', 'ol
dbalanceDest',
       'newbalanceDest', 'transaction_type',
       'diff_newbalanceDest_oldbalanceDest',
       'diff_newbalanceDest_newbalanceOrig',
       'diff_newbalanceOrig_oldbalanceOrg'],
      dtype='object')
```

In [141]:
```
selected_features_df = features[selected_features]
selected_features_df
```

Out[141]:

	amount	oldbalanceOrg	newbalanceOrig	oldbalanceDest	newbalanceDest	transact
0	93780.30	15111903.18	15205683.49	2681083.07	2587302.77	
1	160256.69	0.00	0.00	367154.71	527411.40	
2	77417.72	90755.66	13337.95	11636.00	0.00	
3	270031.56	150274.49	0.00	920127.61	1190159.18	
4	88908.40	127553.00	216461.40	348962.04	260053.64	
...	...	...	...	...	...	
131553	78212.39	11403.00	89615.39	1596540.50	1518328.11	
131554	60793.95	42710.00	0.00	270439.52	333346.03	
131555	170843.32	0.00	0.00	200303.94	371147.26	
131556	35681.14	9412532.10	9448213.24	682786.15	647105.02	
131557	40489.14	16324.00	56813.14	3192642.65	3152153.50	

131558 rows × 9 columns

In [142]:
```
processed_training_df = selected_features_df.join(targets)
processed_training_df
```

Out[142]:

	amount	oldbalanceOrg	newbalanceOrig	oldbalanceDest	newbalanceDest	transact
0	93780.30	15111903.18	15205683.49	2681083.07	2587302.77	
1	160256.69	0.00	0.00	367154.71	527411.40	
2	77417.72	90755.66	13337.95	11636.00	0.00	
3	270031.56	150274.49	0.00	920127.61	1190159.18	
4	88908.40	127553.00	216461.40	348962.04	260053.64	
...	...	...	...	...	...	
131553	78212.39	11403.00	89615.39	1596540.50	1518328.11	
131554	60793.95	42710.00	0.00	270439.52	333346.03	
131555	170843.32	0.00	0.00	200303.94	371147.26	
131556	35681.14	9412532.10	9448213.24	682786.15	647105.02	
131557	40489.14	16324.00	56813.14	3192642.65	3152153.50	

131558 rows × 10 columns

In [143]:
```
processed_training_df.to_csv("processed_training_df.csv", index=False)
```

Modelling

In [144]:
```
os.chdir('.')

file_path = "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[144]:

	amount	oldbalanceOrg	newbalanceOrig	oldbalanceDest	newbalanceDest	transaction_ty
0	93780.30	15111903.18	15205683.49	2681083.07	2587302.77	
1	160256.69	0.00	0.00	367154.71	527411.40	
2	77417.72	90755.66	13337.95	11636.00	0.00	
3	270031.56	150274.49	0.00	920127.61	1190159.18	
4	88908.40	127553.00	216461.40	348962.04	260053.64	

In [131]:
```
df.shape
```

Out[131]:
```
(131558, 10)
```

In [132]:
```
df['isFraud'].value_counts()
```

Out[132]:
```
0    125000
1      6558
Name: isFraud, dtype: int64
```

In [170]:
```
X_train = df.drop(columns=['isFraud'])
y_train = df['isFraud']
```

In [171]:
```
validation_df = pd.read_csv("frauds_out_of_sample_validation_data.csv")
validation_df
```

Out[171]:

	amount	oldbalanceOrg	newbalanceOrig	oldbalanceDest	newbalanceDest	isFraud
0	154984.40	144543.45	299527.86	1244827.23	1089842.83	0
1	102179.95	25795.00	127974.95	297922.81	195742.86	0
2	193003.77	22712.80	0.00	265335.38	458339.15	0
3	3076.59	958752.73	961829.32	1183682.89	1180606.30	0
4	245568.54	0.00	0.00	1466715.44	1712283.97	0
...	...	...	...	...	...	...
842216	116319.29	17212831.21	17329150.51	8861581.51	8745262.22	0
842217	226883.90	10384.00	0.00	340540.79	567424.68	0
842218	193325.47	1442644.47	1635969.94	812593.27	619267.80	0
842219	44153.01	16217575.54	16261728.54	55431.63	11278.63	0
842220	166936.06	172335.00	5398.94	366931.90	533867.95	0

842221 rows × 11 columns

In [172]:
```
validation_df['isFraud'].value_counts()
```

Out[172]:
```
0    840582
1      1639
Name: isFraud, dtype: int64
```

In [173]:
```
validation_df['isFraud'].mean()
```

Out[173]:
```
0.0019460450404347554
```

In [174]:
```
validation_df = validation_df.drop(columns=['orgCustomerType'])
validation_df
```

Out[174]:

	amount	oldbalanceOrg	newbalanceOrig	oldbalanceDest	newbalanceDest	isFraud
0	154984.40	144543.45	299527.86	1244827.23	1089842.83	0
1	102179.95	25795.00	127974.95	297922.81	195742.86	0
2	193003.77	22712.80	0.00	265335.38	458339.15	0
3	3076.59	958752.73	961829.32	1183682.89	1180606.30	0
4	245568.54	0.00	0.00	1466715.44	1712283.97	0
...	...	...	...	...	...	...
842216	116319.29	17212831.21	17329150.51	8861581.51	8745262.22	0
842217	226883.90	10384.00	0.00	340540.79	567424.68	0
842218	193325.47	1442644.47	1635969.94	812593.27	619267.80	0
842219	44153.01	16217575.54	16261728.54	55431.63	11278.63	0
842220	166936.06	172335.00	5398.94	366931.90	533867.95	0

842221 rows × 10 columns

In [175]:
```
X_test = validation_df.drop(columns=['isFraud'])
y_test = validation_df['isFraud']
```

In [176]:
```
X_test
```

Out[176]:

	amount	oldbalanceOrg	newbalanceOrig	oldbalanceDest	newbalanceDest	transact
0	154984.40	144543.45	299527.86	1244827.23	1089842.83	
1	102179.95	25795.00	127974.95	297922.81	195742.86	
2	193003.77	22712.80	0.00	265335.38	458339.15	
3	3076.59	958752.73	961829.32	1183682.89	1180606.30	
4	245568.54	0.00	0.00	1466715.44	1712283.97	
...	...	...	...	...	...	
842216	116319.29	17212831.21	17329150.51	8861581.51	8745262.22	
842217	226883.90	10384.00	0.00	340540.79	567424.68	
842218	193325.47	1442644.47	1635969.94	812593.27	619267.80	
842219	44153.01	16217575.54	16261728.54	55431.63	11278.63	
842220	166936.06	172335.00	5398.94	366931.90	533867.95	

842221 rows × 9 columns

Normalization

In [177]:
```
from sklearn.preprocessing import MinMaxScaler

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

Isolation Forest

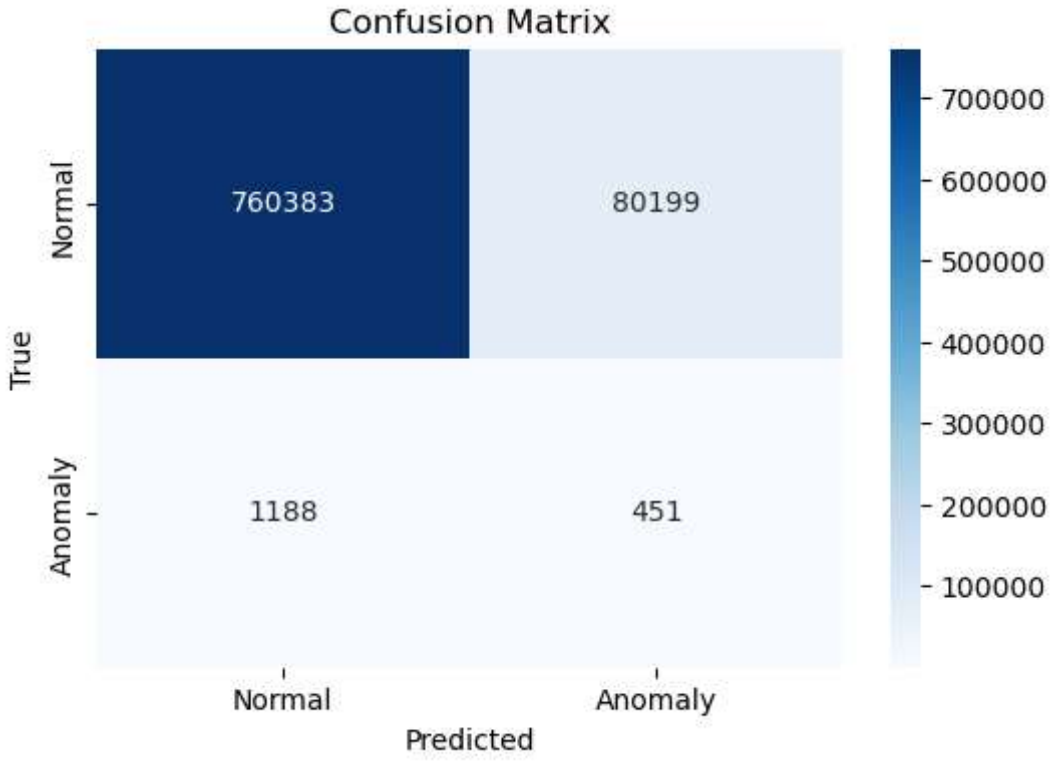
In [195]:
```
from sklearn.ensemble import IsolationForest
from sklearn.metrics import accuracy_score, precision_score, recall_score, r

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

Out[195]:
```
IsolationForest(contamination=0.05)
```

In [196]:
```
# 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
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: 0.9034
Precision: 0.0056
Recall: 0.2752
AUC: 0.5899
```

In [197]:
```
from sklearn.metrics import confusion_matrix
import seaborn as sns
import matplotlib.pyplot as plt

# Compute confusion matrix
cm = confusion_matrix(y_test, predictions)

# Plot confusion matrix
plt.figure(figsize=(6, 4))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', xticklabels=['Normal', '
plt.xlabel('Predicted')
plt.ylabel('True')
plt.title('Confusion Matrix')
plt.show()
```



One Class SVM

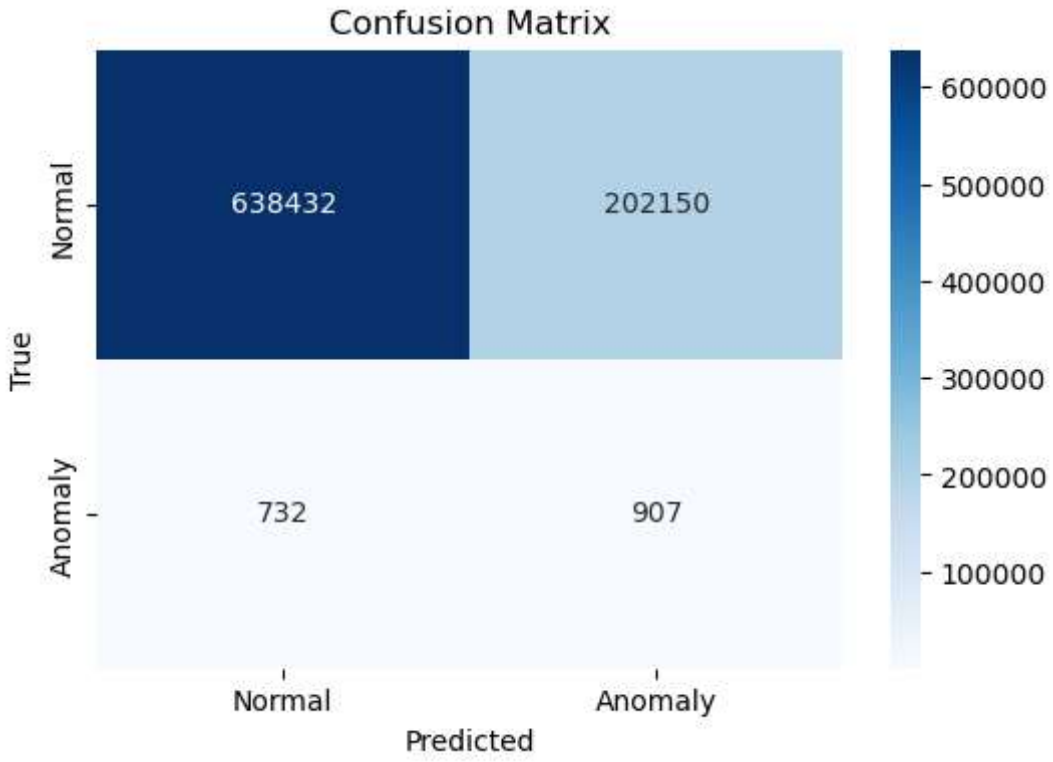
In [181]:
```
from sklearn.svm import OneClassSVM

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

Out[181]:
```
OneClassSVM(nu=0.05)
```

In [182]:
```
# 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}")
```

```
Accuracy: 0.7591
Precision: 0.0045
Recall: 0.5534
AUC: 0.6564
```

In [183]:
```
# Compute confusion matrix
cm = confusion_matrix(y_test, predictions)

# Plot confusion matrix
plt.figure(figsize=(6, 4))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', xticklabels=['Normal', '
plt.xlabel('Predicted')
plt.ylabel('True')
plt.title('Confusion Matrix')
plt.show()
```



Autoencoder Neural Network

In [198]:
```
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, v
```

```
Epoch 1/50
514/514 ━━━━━━━━━━━━━━━━━━━━ 10s 16ms/step - loss: 0.3077 - val_loss: 0.17
38
Epoch 2/50
514/514 ━━━━━━━━━━━━━━━━━━━━ 7s 14ms/step - loss: 0.1762 - val_loss: 0.172
0
Epoch 3/50
514/514 ━━━━━━━━━━━━━━━━━━━━ 11s 15ms/step - loss: 0.1754 - val_loss: 0.17
22
Epoch 4/50
514/514 ━━━━━━━━━━━━━━━━━━━━ 8s 16ms/step - loss: 0.1748 - val_loss: 0.172
4
Epoch 5/50
514/514 ━━━━━━━━━━━━━━━━━━━━ 10s 15ms/step - loss: 0.1748 - val_loss: 0.17
23
Epoch 6/50
514/514 ━━━━━━━━━━━━━━━━━━━━ 8s 16ms/step - loss: 0.1749 - val_loss: 0.172
2
Epoch 7/50
514/514 ━━━━━━━━━━━━━━━━━━━━ 7s 14ms/step - loss: 0.1749 - val_loss: 0.172
3
Epoch 8/50
514/514 ━━━━━━━━━━━━━━━━━━━━ 8s 16ms/step - loss: 0.1748 - val_loss: 0.172
2
Epoch 9/50
514/514 ━━━━━━━━━━━━━━━━━━━━ 7s 14ms/step - loss: 0.1749 - val_loss: 0.172
2
Epoch 10/50
514/514 ━━━━━━━━━━━━━━━━━━━━ 8s 16ms/step - loss: 0.1748 - val_loss: 0.172
3
Epoch 11/50
514/514 ━━━━━━━━━━━━━━━━━━━━ 7s 14ms/step - loss: 0.1748 - val_loss: 0.172
2
Epoch 12/50
514/514 ━━━━━━━━━━━━━━━━━━━━ 9s 17ms/step - loss: 0.1747 - val_loss: 0.172
4
Epoch 13/50
514/514 ━━━━━━━━━━━━━━━━━━━━ 7s 14ms/step - loss: 0.1748 - val_loss: 0.172
4
Epoch 14/50
514/514 ━━━━━━━━━━━━━━━━━━━━ 8s 16ms/step - loss: 0.1749 - val_loss: 0.172
4
Epoch 15/50
514/514 ━━━━━━━━━━━━━━━━━━━━ 7s 14ms/step - loss: 0.1748 - val_loss: 0.172
4
Epoch 16/50
514/514 ━━━━━━━━━━━━━━━━━━━━ 8s 16ms/step - loss: 0.1748 - val_loss: 0.172
4
Epoch 17/50
514/514 ━━━━━━━━━━━━━━━━━━━━ 7s 14ms/step - loss: 0.1747 - val_loss: 0.172
4
Epoch 18/50
514/514 ━━━━━━━━━━━━━━━━━━━━ 8s 16ms/step - loss: 0.1747 - val_loss: 0.172
5
Epoch 19/50
514/514 ━━━━━━━━━━━━━━━━━━━━ 7s 14ms/step - loss: 0.1748 - val_loss: 0.172
6
Epoch 20/50
514/514 ━━━━━━━━━━━━━━━━━━━━ 8s 16ms/step - loss: 0.1746 - val_loss: 0.172
6
Epoch 21/50
```

```
514/514 ━━━━━━━━━━━━━━━━━━━━ 8s 15ms/step - loss: 0.1747 - val_loss: 0.172
8
Epoch 22/50
514/514 ━━━━━━━━━━━━━━━━━━━━ 8s 15ms/step - loss: 0.1746 - val_loss: 0.172
7
Epoch 23/50
514/514 ━━━━━━━━━━━━━━━━━━━━ 8s 15ms/step - loss: 0.1748 - val_loss: 0.172
7
Epoch 24/50
514/514 ━━━━━━━━━━━━━━━━━━━━ 8s 15ms/step - loss: 0.1747 - val_loss: 0.172
8
Epoch 25/50
514/514 ━━━━━━━━━━━━━━━━━━━━ 8s 16ms/step - loss: 0.1746 - val_loss: 0.172
9
Epoch 26/50
514/514 ━━━━━━━━━━━━━━━━━━━━ 8s 15ms/step - loss: 0.1747 - val_loss: 0.173
0
Epoch 27/50
514/514 ━━━━━━━━━━━━━━━━━━━━ 8s 15ms/step - loss: 0.1747 - val_loss: 0.172
9
Epoch 28/50
514/514 ━━━━━━━━━━━━━━━━━━━━ 8s 15ms/step - loss: 0.1745 - val_loss: 0.173
0
Epoch 29/50
514/514 ━━━━━━━━━━━━━━━━━━━━ 8s 16ms/step - loss: 0.1747 - val_loss: 0.173
0
Epoch 30/50
514/514 ━━━━━━━━━━━━━━━━━━━━ 8s 15ms/step - loss: 0.1746 - val_loss: 0.173
1
Epoch 31/50
514/514 ━━━━━━━━━━━━━━━━━━━━ 8s 15ms/step - loss: 0.1748 - val_loss: 0.173
2
Epoch 32/50
514/514 ━━━━━━━━━━━━━━━━━━━━ 11s 17ms/step - loss: 0.1745 - val_loss: 0.17
31
Epoch 33/50
514/514 ━━━━━━━━━━━━━━━━━━━━ 7s 14ms/step - loss: 0.1745 - val_loss: 0.173
2
Epoch 34/50
514/514 ━━━━━━━━━━━━━━━━━━━━ 8s 16ms/step - loss: 0.1747 - val_loss: 0.173
3
Epoch 35/50
514/514 ━━━━━━━━━━━━━━━━━━━━ 10s 14ms/step - loss: 0.1746 - val_loss: 0.17
31
Epoch 36/50
514/514 ━━━━━━━━━━━━━━━━━━━━ 8s 16ms/step - loss: 0.1747 - val_loss: 0.173
2
Epoch 37/50
514/514 ━━━━━━━━━━━━━━━━━━━━ 8s 15ms/step - loss: 0.1746 - val_loss: 0.173
2
Epoch 38/50
514/514 ━━━━━━━━━━━━━━━━━━━━ 8s 16ms/step - loss: 0.1746 - val_loss: 0.173
2
Epoch 39/50
514/514 ━━━━━━━━━━━━━━━━━━━━ 7s 14ms/step - loss: 0.1747 - val_loss: 0.173
2
Epoch 40/50
514/514 ━━━━━━━━━━━━━━━━━━━━ 8s 16ms/step - loss: 0.1745 - val_loss: 0.173
3
Epoch 41/50
514/514 ━━━━━━━━━━━━━━━━━━━━ 7s 14ms/step - loss: 0.1745 - val_loss: 0.173
```

```
2
Epoch 42/50
514/514 ━━━━━━━━━━━━━━━━━━━━ 8s 16ms/step - loss: 0.1747 - val_loss: 0.173
3
Epoch 43/50
514/514 ━━━━━━━━━━━━━━━━━━━━ 7s 14ms/step - loss: 0.1747 - val_loss: 0.173
2
Epoch 44/50
514/514 ━━━━━━━━━━━━━━━━━━━━ 8s 16ms/step - loss: 0.1745 - val_loss: 0.173
4
Epoch 45/50
514/514 ━━━━━━━━━━━━━━━━━━━━ 8s 15ms/step - loss: 0.1748 - val_loss: 0.173
3
Epoch 46/50
514/514 ━━━━━━━━━━━━━━━━━━━━ 11s 16ms/step - loss: 0.1746 - val_loss: 0.17
33
Epoch 47/50
514/514 ━━━━━━━━━━━━━━━━━━━━ 7s 14ms/step - loss: 0.1746 - val_loss: 0.173
4
Epoch 48/50
514/514 ━━━━━━━━━━━━━━━━━━━━ 11s 16ms/step - loss: 0.1746 - val_loss: 0.17
35
Epoch 49/50
514/514 ━━━━━━━━━━━━━━━━━━━━ 11s 16ms/step - loss: 0.1744 - val_loss: 0.17
33
Epoch 50/50
514/514 ━━━━━━━━━━━━━━━━━━━━ 7s 14ms/step - loss: 0.1747 - val_loss: 0.173
3
```

Out[198]:
```
<keras.src.callbacks.history.History at 0x258b4ae0c10>
```

In [199]:
```
test_reconstructions = autoencoder.predict(X_test)
test_errors = np.mean(np.square(X_test - test_reconstructions), axis=1)
error_threshold = 0.01

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

```
26320/26320 ━━━━━━━━━━━━━━━━━━━━ 51s 2ms/step
```

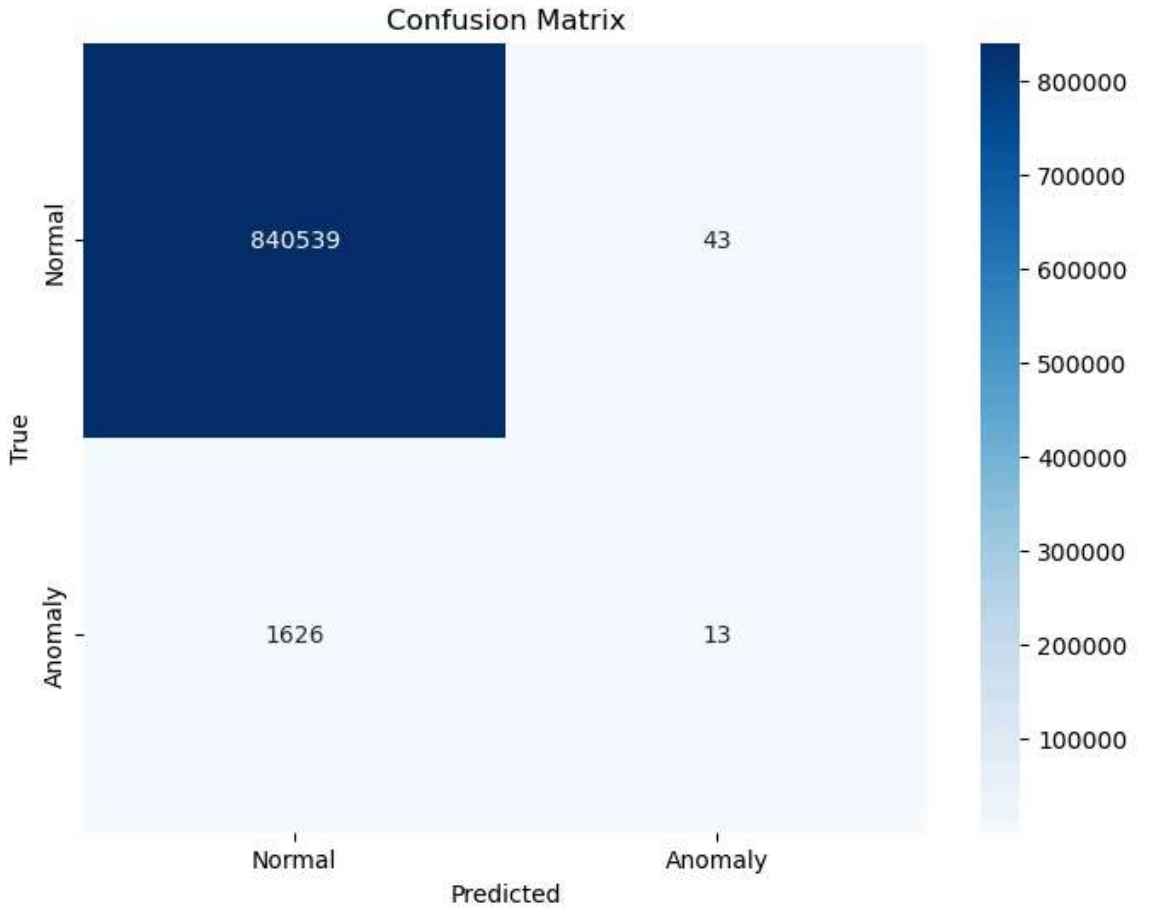
In [200]:
```
anomaly_predictions
```

Out[200]:
```
array([0, 0, 0, ..., 0, 0, 0])
```

In [201]:
```
from sklearn.metrics import accuracy_score, precision_score, recall_score, r

# 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: 0.9980
Precision: 0.2321
Recall: 0.0079
AUC: 0.5039
```

In [202]:
```
# Compute confusion matrix
cm = confusion_matrix(y_test, anomaly_predictions)

# Plot confusion matrix
plt.figure(figsize=(8, 6))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', xticklabels=['Normal', 'A
plt.xlabel('Predicted')
plt.ylabel('True')
plt.title('Confusion Matrix')
plt.show()
```



Elliptic Envelope

In [203]:
```
from sklearn.covariance import EllipticEnvelope

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

```
C:\Users\mcsan\anaconda3\lib\site-packages\sklearn\covariance\_robust_co
variance.py:183: RuntimeWarning: Determinant has increased; this should
not happen: log(det) > log(previous_det) (-186.450747992064066 > -187.90
5558166980995). You may want to try with a higher value of support_fract
ion (current value: 0.503).
  warnings.warn(
C:\Users\mcsan\anaconda3\lib\site-packages\sklearn\covariance\_robust_co
variance.py:183: RuntimeWarning: Determinant has increased; this should
not happen: log(det) > log(previous_det) (-186.172276740733338 > -188.18
8653704665228). You may want to try with a higher value of support_fract
ion (current value: 0.503).
  warnings.warn(
C:\Users\mcsan\anaconda3\lib\site-packages\sklearn\covariance\_robust_co
variance.py:183: RuntimeWarning: Determinant has increased; this should
not happen: log(det) > log(previous_det) (-186.106691919258026 > -188.68
8268485129413). You may want to try with a higher value of support_fract
ion (current value: 0.503).
  warnings.warn(
C:\Users\mcsan\anaconda3\lib\site-packages\sklearn\covariance\_robust_co
```

In [204]:
```
# 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 [205]:
```
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: 0.9324
Precision: 0.0003
Recall: 0.0092
AUC: 0.4717
```

In [ ]:
```
```