In [1]:

```
import pandas as pd
```

In [2]:

```
df = pd.read_csv("C:/Users/Hithesha/Downloads/Fraud Detection Project/PS_20174392719_1491204439457_log.csv/PS_20
```

In [3]:

```
df.head(10)
```

Out[3]:

	step	type	amount	nameOrig	oldbalanceOrg	newbalanceOrig	nameDest	oldbalanceDest	newbalanceDest	isFraud	isFla
0	1	PAYMENT	9839.64	C1231006815	170136.00	160296.36	M1979787155	0.0	0.00	0	
1	1	PAYMENT	1864.28	C1666544295	21249.00	19384.72	M2044282225	0.0	0.00	0	
2	1	TRANSFER	181.00	C1305486145	181.00	0.00	C553264065	0.0	0.00	1	
3	1	CASH_OUT	181.00	C840083671	181.00	0.00	C38997010	21182.0	0.00	1	
4	1	PAYMENT	11668.14	C2048537720	41554.00	29885.86	M1230701703	0.0	0.00	0	
5	1	PAYMENT	7817.71	C90045638	53860.00	46042.29	M573487274	0.0	0.00	0	
6	1	PAYMENT	7107.77	C154988899	183195.00	176087.23	M408069119	0.0	0.00	0	
7	1	PAYMENT	7861.64	C1912850431	176087.23	168225.59	M633326333	0.0	0.00	0	
8	1	PAYMENT	4024.36	C1265012928	2671.00	0.00	M1176932104	0.0	0.00	0	
9	1	DEBIT	5337.77	C712410124	41720.00	36382.23	C195600860	41898.0	40348.79	0	

In [4]:

```
df.columns
```

Out[4]:

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

In [5]:

```
df
```

Out[5]:

	step	type	amount	nameOrig	oldbalanceOrg	newbalanceOrig	nameDest	oldbalanceDest	newbalanceDest	isFrau
0	1	PAYMENT	9839.64	C1231006815	170136.00	160296.36	M1979787155	0.00	0.00	
1	1	PAYMENT	1864.28	C1666544295	21249.00	19384.72	M2044282225	0.00	0.00	
2	1	TRANSFER	181.00	C1305486145	181.00	0.00	C553264065	0.00	0.00	
3	1	CASH_OUT	181.00	C840083671	181.00	0.00	C38997010	21182.00	0.00	
4	1	PAYMENT	11668.14	C2048537720	41554.00	29885.86	M1230701703	0.00	0.00	
...	...	...	...	...	...	...	...	...	...	
1048570	95	CASH_OUT	132557.35	C1179511630	479803.00	347245.65	C435674507	484329.37	616886.72	
1048571	95	PAYMENT	9917.36	C1956161225	90545.00	80627.64	M668364942	0.00	0.00	
1048572	95	PAYMENT	14140.05	C2037964975	20545.00	6404.95	M1355182933	0.00	0.00	
1048573	95	PAYMENT	10020.05	C1633237354	90605.00	80584.95	M1964992463	0.00	0.00	
1048574	95	PAYMENT	11450.03	C1264356443	80584.95	69134.92	M677577406	0.00	0.00	

1048575 rows × 11 columns

In [6]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1048575 entries, 0 to 1048574
Data columns (total 11 columns):
 #   Column          Non-Null Count    Dtype  
---  ------          --------------    -----  
 0   step            1048575 non-null  int64  
 1   type            1048575 non-null  object 
 2   amount          1048575 non-null  float64
 3   nameOrig        1048575 non-null  object 
 4   oldbalanceOrg   1048575 non-null  float64
 5   newbalanceOrig  1048575 non-null  float64
 6   nameDest        1048575 non-null  object 
 7   oldbalanceDest  1048575 non-null  float64
 8   newbalanceDest  1048575 non-null  float64
 9   isFraud         1048575 non-null  int64  
 10  isFlaggedFraud  1048575 non-null  int64  
dtypes: float64(5), int64(3), object(3)
memory usage: 88.0+ MB
```

In [7]:
```
df.describe()
```

Out[7]:

	step	amount	oldbalanceOrg	newbalanceOrig	oldbalanceDest	newbalanceDest	isFraud	isFlaggedFraud
count	1.048575e+06	1.048575e+06	1.048575e+06	1.048575e+06	1.048575e+06	1.048575e+06	1.048575e+06	1048575.0
mean	2.696617e+01	1.586670e+05	8.740095e+05	8.938089e+05	9.781600e+05	1.114198e+06	1.089097e-03	0.0
std	1.562325e+01	2.649409e+05	2.971751e+06	3.008271e+06	2.296780e+06	2.416593e+06	3.298351e-02	0.0
min	1.000000e+00	1.000000e-01	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.0
25%	1.500000e+01	1.214907e+04	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.0
50%	2.000000e+01	7.634333e+04	1.600200e+04	0.000000e+00	1.263772e+05	2.182604e+05	0.000000e+00	0.0
75%	3.900000e+01	2.137619e+05	1.366420e+05	1.746000e+05	9.159235e+05	1.149808e+06	0.000000e+00	0.0
max	9.500000e+01	1.000000e+07	3.890000e+07	3.890000e+07	4.210000e+07	4.220000e+07	1.000000e+00	0.0

In [8]:
```
df['type'].unique()
```

Out[8]:
```
array(['PAYMENT', 'TRANSFER', 'CASH_OUT', 'DEBIT', 'CASH_IN'],
      dtype=object)
```

In [9]:
```
df['isFraud'].value_counts(normalize = True) * 100
```

Out[9]:
```
isFraud
0    99.89109
1     0.10891
Name: proportion, dtype: float64
```

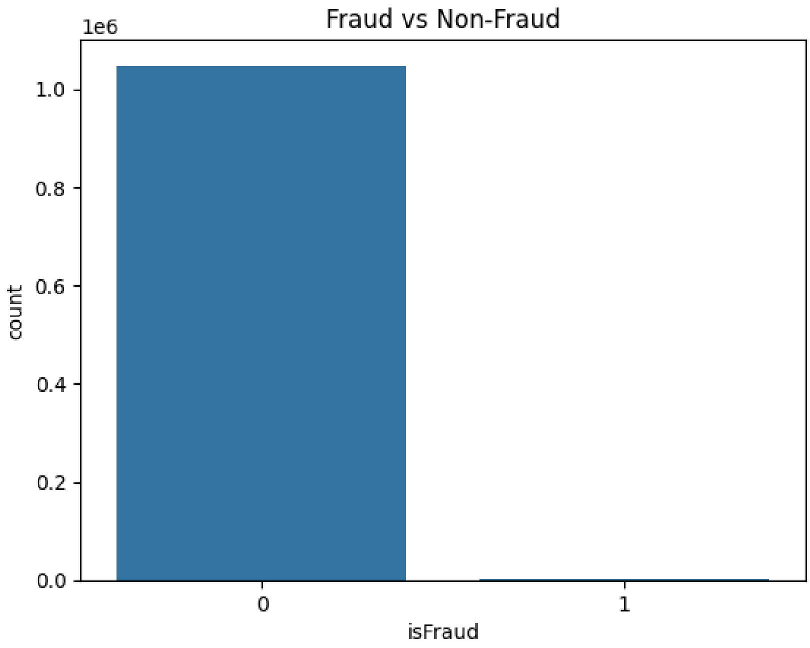
In [10]:
```
df['type'].unique()
```

Out[10]:
```
array(['PAYMENT', 'TRANSFER', 'CASH_OUT', 'DEBIT', 'CASH_IN'],
      dtype=object)
```

Exploratory Data Analysis

In [11]:
```
import matplotlib.pyplot as plt
import seaborn as sns
```

In [12]:
```
sns.countplot(data = df, x = "isFraud")
plt.title("Fraud vs Non-Fraud")
plt.show()
```



In [13]:
```
fraud_by_type = df.groupby("type")["isFraud"].mean() * 100
```

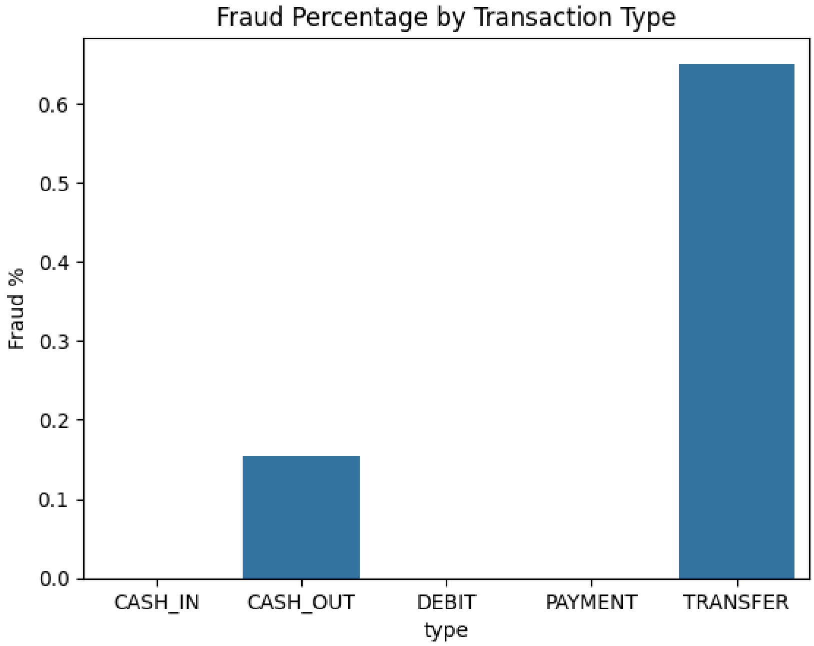
In [14]:
```
fraud_by_type.head()
```

Out[14]:
```
type
CASH_IN      0.000000
CASH_OUT     0.154694
DEBIT        0.000000
PAYMENT      0.000000
TRANSFER     0.650122
Name: isFraud, dtype: float64
```

In [15]:
```
fraud_by_type = df.groupby("type")["isFraud"].mean() * 100
print(fraud_by_type)
```

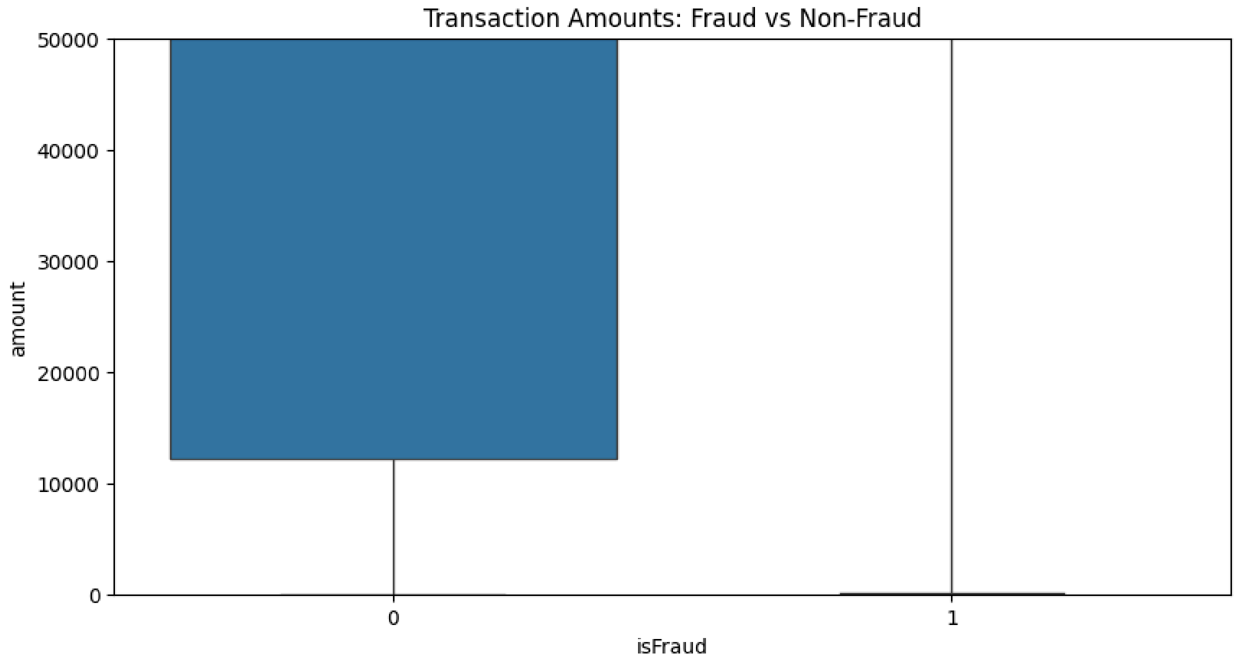
```
type
CASH_IN      0.000000
CASH_OUT     0.154694
DEBIT        0.000000
PAYMENT      0.000000
TRANSFER     0.650122
Name: isFraud, dtype: float64
```

In [16]:
```
sns.barplot(x = fraud_by_type.index, y = fraud_by_type.values)
plt.title("Fraud Percentage by Transaction Type")
plt.ylabel("Fraud %")
plt.show()
```



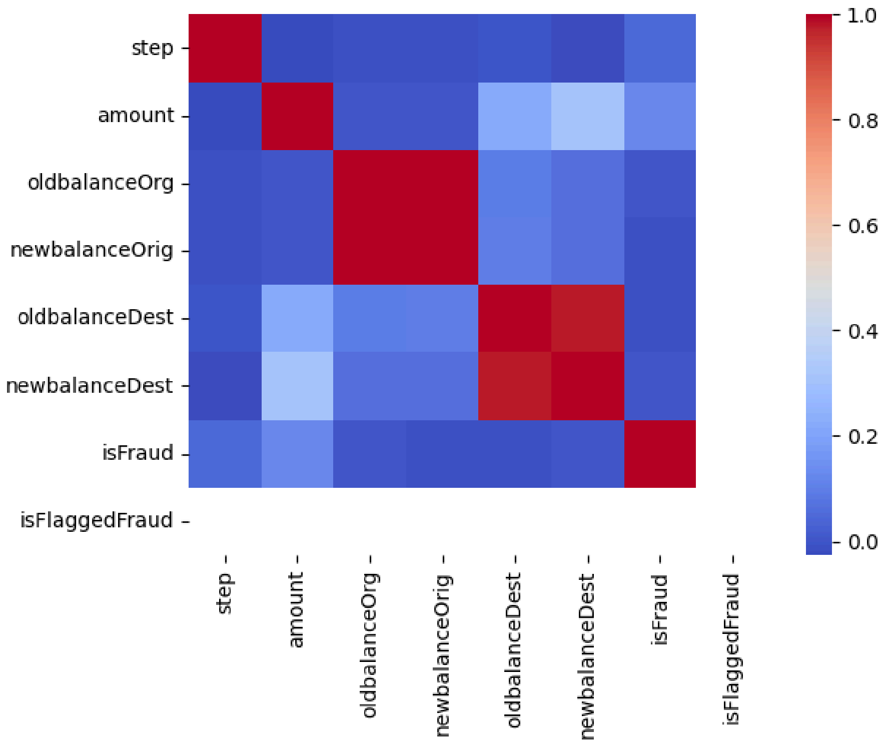
In [17]:

```
plt.figure(figsize=(10,5))
sns.boxplot(x="isFraud", y="amount", data=df)
plt.ylim(0, 50000)
plt.title("Transaction Amounts: Fraud vs Non-Fraud")
plt.show()
```



In [18]:

```
numeric_df = df.select_dtypes(include = 'number')
corr_matrix = numeric_df.corr()
sns.heatmap(corr_matrix, annot = False, cmap = "coolwarm")
plt.show()
```



In [19]:

```
df["error_balance_orig"] = df["oldbalanceOrg"] - df["amount"] - df["newbalanceOrig"]
df["error_balance_dest"] = df["oldbalanceDest"] + df["amount"] - df["newbalanceDest"]

print(df[df["isFraud"] == 1][["type", "amount", "error_balance_orig", "error_balance_dest"]].head())
```

```
         type   amount  error_balance_orig  error_balance_dest
2    TRANSFER    181.0                 0.0               181.0
3    CASH_OUT    181.0                 0.0             21363.0
251  TRANSFER   2806.0                 0.0              2806.0
252  CASH_OUT   2806.0                 0.0             29008.0
680  TRANSFER  20128.0                 0.0             20128.0
```

Fraud Detection

In [20]:

```
df_model = df.copy()
df_model = df[df["type"].isin(["TRANSFER","CASH_OUT"])]
```

In [21]:

```
df.head()
```

Out[21]:

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

In [22]:

```
df["type"].unique()
```

Out[22]:

```
array(['PAYMENT', 'TRANSFER', 'CASH_OUT', 'DEBIT', 'CASH_IN'],
      dtype=object)
```

In [23]:

```
df_model["type"].unique()
```

Out[23]:

```
array(['TRANSFER', 'CASH_OUT'], dtype=object)
```

In [24]:

```
df_model.head()
```

Out[24]:

	step	type	amount	nameOrig	oldbalanceOrg	newbalanceOrig	nameDest	oldbalanceDest	newbalanceDest	isFraud	isl
2	1	TRANSFER	181.00	C1305486145	181.0	0.0	C553264065	0.0	0.00	1	
3	1	CASH_OUT	181.00	C840083671	181.0	0.0	C38997010	21182.0	0.00	1	
15	1	CASH_OUT	229133.94	C905080434	15325.0	0.0	C476402209	5083.0	51513.44	0	
19	1	TRANSFER	215310.30	C1670993182	705.0	0.0	C1100439041	22425.0	0.00	0	
24	1	TRANSFER	311685.89	C1984094095	10835.0	0.0	C932583850	6267.0	2719172.89	0	

In [25]:

```
df_model.loc[:, "type"] = df_model["type"].map({"TRANSFER": 0, "CASH_OUT": 1})
```

In [26]:

```
df_model.head()
```

Out[26]:

	step	type	amount	nameOrig	oldbalanceOrg	newbalanceOrig	nameDest	oldbalanceDest	newbalanceDest	isFraud	isFlaggec
2	1	0	181.00	C1305486145	181.0	0.0	C553264065	0.0	0.00	1	
3	1	1	181.00	C840083671	181.0	0.0	C38997010	21182.0	0.00	1	
15	1	1	229133.94	C905080434	15325.0	0.0	C476402209	5083.0	51513.44	0	
19	1	0	215310.30	C1670993182	705.0	0.0	C1100439041	22425.0	0.00	0	
24	1	0	311685.89	C1984094095	10835.0	0.0	C932583850	6267.0	2719172.89	0	

In [27]:

```
from sklearn.preprocessing import LabelEncoder
```

Unsupervised Algorithm IsolationForest

In [28]:
```
from sklearn.ensemble import IsolationForest
```

In [29]:
```
features = ["amount", "oldbalanceOrg", "newbalanceOrig", "oldbalanceDest", "newbalanceDest", "type"]

X = df_model[features]
y = df_model["isFraud"]
```

In [30]:
```
X_copy = X.copy()
```

In [31]:
```
X_copy.head()
```

Out[31]:

	amount	oldbalanceOrg	newbalanceOrig	oldbalanceDest	newbalanceDest	type
2	181.00	181.0	0.0	0.0	0.00	0
3	181.00	181.0	0.0	21182.0	0.00	1
15	229133.94	15325.0	0.0	5083.0	51513.44	1
19	215310.30	705.0	0.0	22425.0	0.00	0
24	311685.89	10835.0	0.0	6267.0	2719172.89	0

In [32]:
```
le = LabelEncoder()
```

In [33]:
```
X_copy['type'] = le.fit_transform(X_copy['type'])
```

In [34]:
```
X_copy.head()
```

Out[34]:

	amount	oldbalanceOrg	newbalanceOrig	oldbalanceDest	newbalanceDest	type
2	181.00	181.0	0.0	0.0	0.00	0
3	181.00	181.0	0.0	21182.0	0.00	1
15	229133.94	15325.0	0.0	5083.0	51513.44	1
19	215310.30	705.0	0.0	22425.0	0.00	0
24	311685.89	10835.0	0.0	6267.0	2719172.89	0

In [35]:
```
X.head()
```

Out[35]:

	amount	oldbalanceOrg	newbalanceOrig	oldbalanceDest	newbalanceDest	type
2	181.00	181.0	0.0	0.0	0.00	0
3	181.00	181.0	0.0	21182.0	0.00	1
15	229133.94	15325.0	0.0	5083.0	51513.44	1
19	215310.30	705.0	0.0	22425.0	0.00	0
24	311685.89	10835.0	0.0	6267.0	2719172.89	0

In [36]:
```
y.head()
```

Out[36]:
```
2     1
3     1
15    0
19    0
24    0
Name: isFraud, dtype: int64
```

In [37]:
```
from sklearn.preprocessing import StandardScaler
```

In [38]:
```
scaler = StandardScaler()
```

In [39]:
```
X_scaled = scaler.fit_transform(X_copy)
```

In [40]:
```
iso_forest = IsolationForest(contamination = 0.0047, random_state = 42)
```

In [41]:
```
y_pred = iso_forest.fit_predict(X_scaled)
```

In [42]:

```
X_copy['anomaly'] = y_pred
```

In [43]:

```
X_copy.head(30)
```

Out[43]:

	amount	oldbalanceOrg	newbalanceOrig	oldbalanceDest	newbalanceDest	type	anomaly
2	181.00	181.00	0.00	0.00	0.00	0	1
3	181.00	181.00	0.00	21182.00	0.00	1	1
15	229133.94	15325.00	0.00	5083.00	51513.44	1	1
19	215310.30	705.00	0.00	22425.00	0.00	0	1
24	311685.89	10835.00	0.00	6267.00	2719172.89	0	1
42	110414.71	26845.41	0.00	288800.00	2415.16	1	1
47	56953.90	1942.02	0.00	70253.00	64106.18	1	1
48	5346.89	0.00	0.00	652637.00	6453430.91	1	1
51	23261.30	20411.53	0.00	25742.00	0.00	1	1
58	62610.80	79114.00	16503.20	517.00	8383.29	0	1
60	82940.31	3017.87	0.00	132372.00	49864.36	1	1
70	47458.86	209534.84	162075.98	52120.00	0.00	1	1
71	136872.92	162075.98	25203.05	217806.00	0.00	1	1
72	94253.33	25203.05	0.00	99773.00	965870.05	1	1
78	42712.39	10363.39	0.00	57901.66	24044.18	0	1
79	77957.68	0.00	0.00	94900.00	22233.65	0	1
80	17231.46	0.00	0.00	24672.00	0.00	0	1
81	78766.03	0.00	0.00	103772.00	277515.05	0	1
82	224606.64	0.00	0.00	354678.92	0.00	0	1
83	125872.53	0.00	0.00	348512.00	3420103.09	0	1
84	379856.23	0.00	0.00	900180.00	19200000.00	0	1
85	1505626.01	0.00	0.00	29031.00	5515763.34	0	1
86	554026.99	0.00	0.00	579285.56	0.00	0	1
87	147543.10	0.00	0.00	223220.00	16518.36	0	1
88	761507.39	0.00	0.00	1280036.23	19200000.00	0	1
89	1429051.47	0.00	0.00	2041543.62	19200000.00	0	1
90	358831.92	0.00	0.00	474384.53	3420103.09	0	1
91	367768.40	0.00	0.00	370763.10	16518.36	0	1
92	209711.11	0.00	0.00	399214.71	2415.16	0	1
93	583848.46	0.00	0.00	667778.00	2107778.11	0	1

In [44]:

```
X_copy['anomaly'] = X_copy['anomaly'].map({1:0, -1:1})
```

In [45]:
```
X_copy.head(30)
```

Out[45]:

	amount	oldbalanceOrg	newbalanceOrig	oldbalanceDest	newbalanceDest	type	anomaly
2	181.00	181.00	0.00	0.00	0.00	0	0
3	181.00	181.00	0.00	21182.00	0.00	1	0
15	229133.94	15325.00	0.00	5083.00	51513.44	1	0
19	215310.30	705.00	0.00	22425.00	0.00	0	0
24	311685.89	10835.00	0.00	6267.00	2719172.89	0	0
42	110414.71	26845.41	0.00	288800.00	2415.16	1	0
47	56953.90	1942.02	0.00	70253.00	64106.18	1	0
48	5346.89	0.00	0.00	652637.00	6453430.91	1	0
51	23261.30	20411.53	0.00	25742.00	0.00	1	0
58	62610.80	79114.00	16503.20	517.00	8383.29	0	0
60	82940.31	3017.87	0.00	132372.00	49864.36	1	0
70	47458.86	209534.84	162075.98	52120.00	0.00	1	0
71	136872.92	162075.98	25203.05	217806.00	0.00	1	0
72	94253.33	25203.05	0.00	99773.00	965870.05	1	0
78	42712.39	10363.39	0.00	57901.66	24044.18	0	0
79	77957.68	0.00	0.00	94900.00	22233.65	0	0
80	17231.46	0.00	0.00	24672.00	0.00	0	0
81	78766.03	0.00	0.00	103772.00	277515.05	0	0
82	224606.64	0.00	0.00	354678.92	0.00	0	0
83	125872.53	0.00	0.00	348512.00	3420103.09	0	0
84	379856.23	0.00	0.00	900180.00	19200000.00	0	0
85	1505626.01	0.00	0.00	29031.00	5515763.34	0	0
86	554026.99	0.00	0.00	579285.56	0.00	0	0
87	147543.10	0.00	0.00	223220.00	16518.36	0	0
88	761507.39	0.00	0.00	1280036.23	19200000.00	0	0
89	1429051.47	0.00	0.00	2041543.62	19200000.00	0	0
90	358831.92	0.00	0.00	474384.53	3420103.09	0	0
91	367768.40	0.00	0.00	370763.10	16518.36	0	0
92	209711.11	0.00	0.00	399214.71	2415.16	0	0
93	583848.46	0.00	0.00	667778.00	2107778.11	0	0

In [46]:
```
from sklearn.metrics import classification_report, confusion_matrix, roc_auc_score, roc_curve
```

In [47]:
```
y_true = df_model['isFraud']
```

In [48]:
```
y_pred = X_copy['anomaly']
```

In [49]:
```
cm = confusion_matrix(y_true, y_pred)

print("Confusion Matrix: \n", cm)
```

```
Confusion Matrix:
 [[457267   1985]
 [   972    170]]
```

In [50]:
```
print("Classsification Report: \n", classification_report(y_true, y_pred, digits = 4))
```

```
Classsification Report:
               precision    recall  f1-score   support

           0     0.9979    0.9957    0.9968    459252
           1     0.0789    0.1489    0.1031      1142

    accuracy                         0.9936    460394
   macro avg     0.5384    0.5723    0.5500    460394
weighted avg     0.9956    0.9936    0.9946    460394
```

In [51]:

```
roc_auc = roc_auc_score(y_true, y_pred)
```

In [52]:

```
print("ROC-AUC Score: ", roc_auc)
```

```
ROC-AUC Score:  0.5722697002479766
```

Simply Calculating Accuracy (Which is not useful)

In [53]:

```
from sklearn.metrics import accuracy_score
```

In [54]:

```
accuracy = accuracy_score(y_true, y_pred)
print("Accuracy :", accuracy * 100)
```

```
Accuracy : 99.35772403636885
```

Supervised Algorithm XGBoost

In [55]:

```
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.metrics import average_precision_score
from xgboost import XGBClassifier
```

In [56]:

```
drop_cols = ["nameOrig", "nameDest", "step"]
```

In [57]:

```
Xxg = df.drop(columns = drop_cols + ['isFraud'])
yxg = df['isFraud']
```

In [58]:

```
Xxg.head()
```

Out[58]:

	type	amount	oldbalanceOrg	newbalanceOrig	oldbalanceDest	newbalanceDest	isFlaggedFraud	error_balance_orig	error_balanc
0	PAYMENT	9839.64	170136.0	160296.36	0.0	0.0	0	0.0	9
1	PAYMENT	1864.28	21249.0	19384.72	0.0	0.0	0	0.0	1
2	TRANSFER	181.00	181.0	0.00	0.0	0.0	0	0.0	
3	CASH_OUT	181.00	181.0	0.00	21182.0	0.0	0	0.0	21
4	PAYMENT	11668.14	41554.0	29885.86	0.0	0.0	0	0.0	11

In [59]:

```
yxg.head()
```

Out[59]:

```
0    0
1    0
2    1
3    1
4    0
Name: isFraud, dtype: int64
```

In [60]:

```
for col in Xxg.select_dtypes(include = ['object']).columns:
    Xxg[col] = Xxg[col].astype('category')
```

In [61]:

```
X_train, X_test, y_train, y_test = train_test_split(Xxg,yxg, test_size = 0.2, random_state = 42, stratify = yxg)
```

In [62]:

```
scale_pos_weight = (y_train.value_counts()[0]/y_train.value_counts()[1])
```

In [63]:

```
print("Scale pos weight: ",scale_pos_weight)
```

```
Scale pos weight:  916.7899343544858
```

In [64]:

```
model = XGBClassifier(n_estimators = 300, max_depth = 6, learning_rate = 0.1, subsample = 0.8, colsample_bytree=
    scale_pos_weight=scale_pos_weight,
    random_state=42,
    eval_metric='logloss',enable_categorical = True,
    tree_method = "hist")
```

In [65]:
```
model.fit(X_train, y_train)
```

Out[65]:
```
▾                         XGBClassifier
XGBClassifier(base_score=None, booster=None, callbacks=None,
              colsample_bylevel=None, colsample_bynode=None,
              colsample_bytree=0.8, early_stopping_rounds=None,
              enable_categorical=True, eval_metric='logloss',
              feature_types=None, gamma=None, gpu_id=None, grow_policy=None,
              importance_type=None, interaction_constraints=None,
              learning_rate=0.1, max_bin=None, max_cat_threshold=None,
              max_cat_to_onehot=None, max_delta_step=None, max_depth=6,
              max_leaves=None, min_child_weight=None, missing=nan,
              monotone_constraints=None, n_estimators=300, n_jobs=None,
```

In [66]:
```
y_pred = model.predict(X_test)
```

In [67]:
```
print("Confusion Matrix: \n", confusion_matrix(y_test, y_pred))
```

```
Confusion Matrix:
 [[209473     14]
 [     7    221]]
```

In [68]:
```
print("Classification Report: ", classification_report(y_test, y_pred))
```

```
Classification Report:                precision    recall  f1-score   support

           0       1.00      1.00      1.00    209487
           1       0.94      0.97      0.95       228

    accuracy                           1.00    209715
   macro avg       0.97      0.98      0.98    209715
weighted avg       1.00      1.00      1.00    209715
```

In [69]:
```
pip install tensorflow
```

```
Requirement already satisfied: tensorflow in c:\users\hithesha\anaconda3\lib\site-packages (2.20.0)
Requirement already satisfied: absl-py>=1.0.0 in c:\users\hithesha\anaconda3\lib\site-packages (from tensorflo
w) (2.3.1)
Requirement already satisfied: astunparse>=1.6.0 in c:\users\hithesha\anaconda3\lib\site-packages (from tensorf
low) (1.6.3)
Requirement already satisfied: flatbuffers>=24.3.25 in c:\users\hithesha\anaconda3\lib\site-packages (from tens
orflow) (25.2.10)
Requirement already satisfied: gast!=0.5.0,!=0.5.1,!=0.5.2,>=0.2.1 in c:\users\hithesha\anaconda3\lib\site-pack
ages (from tensorflow) (0.6.0)
Requirement already satisfied: google_pasta>=0.1.1 in c:\users\hithesha\anaconda3\lib\site-packages (from tenso
rflow) (0.2.0)
Requirement already satisfied: libclang>=13.0.0 in c:\users\hithesha\anaconda3\lib\site-packages (from tensorfl
ow) (18.1.1)
Requirement already satisfied: opt_einsum>=2.3.2 in c:\users\hithesha\anaconda3\lib\site-packages (from tensorf
low) (3.4.0)
Requirement already satisfied: packaging in c:\users\hithesha\anaconda3\lib\site-packages (from tensorflow) (2
3.1)
Requirement already satisfied: protobuf>=5.28.0 in c:\users\hithesha\anaconda3\lib\site-packages (from tensorfl
ow) (6.32.0)
Requirement already satisfied: requests<3,>=2.21.0 in c:\users\hithesha\anaconda3\lib\site-packages (from tenso
rflow) (2.31.0)
Requirement already satisfied: setuptools in c:\users\hithesha\anaconda3\lib\site-packages (from tensorflow) (6
5.6.3)
Requirement already satisfied: six>=1.12.0 in c:\users\hithesha\anaconda3\lib\site-packages (from tensorflow)
(1.16.0)
Requirement already satisfied: termcolor>=1.1.0 in c:\users\hithesha\anaconda3\lib\site-packages (from tensorfl
ow) (3.1.0)
Requirement already satisfied: typing_extensions>=3.6.6 in c:\users\hithesha\anaconda3\lib\site-packages (from
tensorflow) (4.7.1)
Requirement already satisfied: wrapt>=1.11.0 in c:\users\hithesha\anaconda3\lib\site-packages (from tensorflow)
(1.17.3)
Requirement already satisfied: grpcio<2.0,>=1.24.3 in c:\users\hithesha\anaconda3\lib\site-packages (from tenso
rflow) (1.74.0)
Requirement already satisfied: tensorboard~=2.20.0 in c:\users\hithesha\anaconda3\lib\site-packages (from tenso
rflow) (2.20.0)
Requirement already satisfied: keras>=3.10.0 in c:\users\hithesha\anaconda3\lib\site-packages (from tensorflow)
(3.11.3)
Requirement already satisfied: numpy>=1.26.0 in c:\users\hithesha\anaconda3\lib\site-packages (from tensorflow)
(1.26.0)
Requirement already satisfied: h5py>=3.11.0 in c:\users\hithesha\anaconda3\lib\site-packages (from tensorflow)
(3.14.0)
Requirement already satisfied: ml_dtypes<1.0.0,>=0.5.1 in c:\users\hithesha\anaconda3\lib\site-packages (from t
ensorflow) (0.5.3)
Requirement already satisfied: charset-normalizer<4,>=2 in c:\users\hithesha\anaconda3\lib\site-packages (from
requests<3,>=2.21.0->tensorflow) (2.0.4)
Requirement already satisfied: idna<4,>=2.5 in c:\users\hithesha\anaconda3\lib\site-packages (from requests<3,>
=2.21.0->tensorflow) (3.4)
Requirement already satisfied: urllib3<3,>=1.21.1 in c:\users\hithesha\anaconda3\lib\site-packages (from reques
ts<3,>=2.21.0->tensorflow) (1.26.16)
Requirement already satisfied: certifi>=2017.4.17 in c:\users\hithesha\anaconda3\lib\site-packages (from reques
ts<3,>=2.21.0->tensorflow) (2025.1.31)
Requirement already satisfied: markdown>=2.6.8 in c:\users\hithesha\anaconda3\lib\site-packages (from tensorboa
rd~=2.20.0->tensorflow) (3.8.2)
Requirement already satisfied: pillow in c:\users\hithesha\anaconda3\lib\site-packages (from tensorboard~=2.20.
0->tensorflow) (9.4.0)
Requirement already satisfied: tensorboard-data-server<0.8.0,>=0.7.0 in c:\users\hithesha\anaconda3\lib\site-pa
ckages (from tensorboard~=2.20.0->tensorflow) (0.7.2)
Requirement already satisfied: werkzeug>=1.0.1 in c:\users\hithesha\anaconda3\lib\site-packages (from tensorboa
rd~=2.20.0->tensorflow) (3.1.3)
Requirement already satisfied: wheel<1.0,>=0.23.0 in c:\users\hithesha\anaconda3\lib\site-packages (from astunp
arse>=1.6.0->tensorflow) (0.38.4)
Requirement already satisfied: rich in c:\users\hithesha\anaconda3\lib\site-packages (from keras>=3.10.0->tenso
rflow) (14.1.0)
Requirement already satisfied: namex in c:\users\hithesha\anaconda3\lib\site-packages (from keras>=3.10.0->tens
orflow) (0.1.0)
Requirement already satisfied: optree in c:\users\hithesha\anaconda3\lib\site-packages (from keras>=3.10.0->ten
sorflow) (0.17.0)
Requirement already satisfied: MarkupSafe>=2.1.1 in c:\users\hithesha\anaconda3\lib\site-packages (from werkzeu
g>=1.0.1->tensorboard~=2.20.0->tensorflow) (2.1.1)
Requirement already satisfied: markdown-it-py>=2.2.0 in c:\users\hithesha\anaconda3\lib\site-packages (from ric
h->keras>=3.10.0->tensorflow) (4.0.0)
Requirement already satisfied: pygments<3.0.0,>=2.13.0 in c:\users\hithesha\anaconda3\lib\site-packages (from r
ich->keras>=3.10.0->tensorflow) (2.15.1)
Requirement already satisfied: mdurl~=0.1 in c:\users\hithesha\anaconda3\lib\site-packages (from markdown-it-py
>=2.2.0->rich->keras>=3.10.0->tensorflow) (0.1.2)
Note: you may need to restart the kernel to use updated packages.
```

In [70]:

```
from tensorflow.keras.models import Model
from tensorflow.keras.layers import Input, Dense
from tensorflow.keras.optimizers import Adam
```

In [71]:

```
from sklearn.preprocessing import LabelEncoder
```

In [72]:

```
X_copy = X_copy.copy()
```

In [73]:

```
encoder = LabelEncoder()
```

In [74]:

```
X_copy['type'] = encoder.fit_transform(X_copy['type'])
```

In [75]:

```
X_copy.head()
```

Out[75]:

	amount	oldbalanceOrg	newbalanceOrig	oldbalanceDest	newbalanceDest	type	anomaly
2	181.00	181.0	0.0	0.0	0.00	0	0
3	181.00	181.0	0.0	21182.0	0.00	1	0
15	229133.94	15325.0	0.0	5083.0	51513.44	1	0
19	215310.30	705.0	0.0	22425.0	0.00	0	0
24	311685.89	10835.0	0.0	6267.0	2719172.89	0	0

In [76]:

```
print("Classes mapped:", dict(zip(encoder.classes_, encoder.transform(encoder.classes_))))
```

```
Classes mapped: {0: 0, 1: 1}
```

Build Auto Encoder

In [77]:

```
df.columns
```

Out[77]:

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

In [78]:

```
X = X_copy.drop('anomaly', axis = 1)
y = X_copy["anomaly"]
```

In [79]:

```
print(X.head())
print()
y.head()
```

```
       amount  oldbalanceOrg  newbalanceOrig  oldbalanceDest  newbalanceDest  \
2      181.00          181.0             0.0             0.0            0.00
3      181.00          181.0             0.0         21182.0            0.00
15  229133.94        15325.0             0.0          5083.0        51513.44
19  215310.30          705.0             0.0         22425.0            0.00
24  311685.89        10835.0             0.0          6267.0      2719172.89

    type
2      0
3      1
15     1
19     0
24     0
```

Out[79]:

```
2     0
3     0
15    0
19    0
24    0
Name: anomaly, dtype: int64
```

In [80]:

```
from sklearn.preprocessing import OneHotEncoder
```

In [81]:

```
encoder = OneHotEncoder(drop = 'first', sparse = False)
```

In [82]:

```
type_encoded = encoder.fit_transform(X[['type']])
```

```
C:\Users\Hithesha\anaconda3\lib\site-packages\sklearn\preprocessing\_encoders.py:975: FutureWarning: `sparse` w
as renamed to `sparse_output` in version 1.2 and will be removed in 1.4. `sparse_output` is ignored unless you
leave `sparse` to its default value.
  warnings.warn(
```

In [83]:

```
import pandas as pd
type_encoded_df = pd.DataFrame(type_encoded, columns=encoder.get_feature_names_out(['type']))
X = X.drop("type", axis=1).reset_index(drop=True)
X = pd.concat([X, type_encoded_df], axis=1)
```

In [84]:

```
from sklearn.preprocessing import StandardScaler
```

In [85]:

```
scaler = StandardScaler()
```

In [86]:

```
X_scaled = scaler.fit_transform(X)
```

In [87]:

```
input_dim = X_scaled.shape[1]
```

Encoder

In [88]:

```
from tensorflow.keras import regularizers
```

In [89]:

```
input_layer = Input(shape=(input_dim,))
encoder = Dense(16, activation="relu", activity_regularizer=regularizers.l1(1e-5))(input_layer)
encoder = Dense(8, activation="relu")(encoder)
```

Decoder

In [90]:

```
decoder = Dense(16, activation='relu')(encoder)
decoder = Dense(input_dim, activation='linear')(decoder)
```

AutoEncoder Model

In [91]:

```
autoencoder = Model(inputs=input_layer, outputs=decoder)
autoencoder.compile(optimizer="adam", loss="mse")
```

Training autoencoder on Non-anomalies only

In [92]:
```
X_train = X_scaled[y == 0]
history = autoencoder.fit(X_train, X_train, epochs = 20, batch_size = 32, validation_split = 0.1, shuffle = True
```

```
Epoch 1/20
12888/12888 ━━━━━━━━━━━━━━━━━━━━ 34s 2ms/step - loss: 0.0167 - val_loss: 0.0013
Epoch 2/20
12888/12888 ━━━━━━━━━━━━━━━━━━━━ 42s 3ms/step - loss: 0.0010 - val_loss: 4.8601e-04
Epoch 3/20
12888/12888 ━━━━━━━━━━━━━━━━━━━━ 30s 2ms/step - loss: 5.9029e-04 - val_loss: 3.9772e-04
Epoch 4/20
12888/12888 ━━━━━━━━━━━━━━━━━━━━ 29s 2ms/step - loss: 6.7450e-04 - val_loss: 3.0343e-04
Epoch 5/20
12888/12888 ━━━━━━━━━━━━━━━━━━━━ 17s 1ms/step - loss: 4.7163e-04 - val_loss: 2.4230e-04
Epoch 6/20
12888/12888 ━━━━━━━━━━━━━━━━━━━━ 16s 1ms/step - loss: 5.7241e-04 - val_loss: 2.1772e-04
Epoch 7/20
12888/12888 ━━━━━━━━━━━━━━━━━━━━ 15s 1ms/step - loss: 4.2254e-04 - val_loss: 2.3987e-04
Epoch 8/20
12888/12888 ━━━━━━━━━━━━━━━━━━━━ 14s 1ms/step - loss: 4.0285e-04 - val_loss: 1.9216e-04
Epoch 9/20
12888/12888 ━━━━━━━━━━━━━━━━━━━━ 14s 1ms/step - loss: 3.2483e-04 - val_loss: 6.4104e-04
Epoch 10/20
12888/12888 ━━━━━━━━━━━━━━━━━━━━ 14s 1ms/step - loss: 3.4870e-04 - val_loss: 3.3483e-04
Epoch 11/20
12888/12888 ━━━━━━━━━━━━━━━━━━━━ 14s 1ms/step - loss: 3.1275e-04 - val_loss: 1.5302e-04
Epoch 12/20
12888/12888 ━━━━━━━━━━━━━━━━━━━━ 14s 1ms/step - loss: 3.3802e-04 - val_loss: 2.8083e-04
Epoch 13/20
12888/12888 ━━━━━━━━━━━━━━━━━━━━ 14s 1ms/step - loss: 3.4964e-04 - val_loss: 1.5864e-04
Epoch 14/20
12888/12888 ━━━━━━━━━━━━━━━━━━━━ 14s 1ms/step - loss: 3.0317e-04 - val_loss: 1.9848e-04
Epoch 15/20
12888/12888 ━━━━━━━━━━━━━━━━━━━━ 14s 1ms/step - loss: 5.0924e-04 - val_loss: 1.5965e-04
Epoch 16/20
12888/12888 ━━━━━━━━━━━━━━━━━━━━ 14s 1ms/step - loss: 2.4212e-04 - val_loss: 1.6425e-04
Epoch 17/20
12888/12888 ━━━━━━━━━━━━━━━━━━━━ 13s 1ms/step - loss: 3.0937e-04 - val_loss: 1.2382e-04
Epoch 18/20
12888/12888 ━━━━━━━━━━━━━━━━━━━━ 13s 1ms/step - loss: 3.1226e-04 - val_loss: 1.2886e-04
Epoch 19/20
12888/12888 ━━━━━━━━━━━━━━━━━━━━ 12s 954us/step - loss: 2.7975e-04 - val_loss: 1.2130e-04
Epoch 20/20
12888/12888 ━━━━━━━━━━━━━━━━━━━━ 12s 959us/step - loss: 2.5579e-04 - val_loss: 3.7062e-04
```

In [93]:
```
import numpy as np
```

Reconstruction Errors

In [94]:
```
reconstructions = autoencoder.predict(X_scaled)
mse = np.mean(np.power(X_scaled - reconstructions, 2), axis=1)
```

```
14388/14388 ━━━━━━━━━━━━━━━━━━━━ 8s 557us/step
```

In [95]:
```
threshold = np.percentile(mse, 95)
preds = (mse > threshold).astype(int)
```

In [96]:
```
from sklearn.metrics import classification_report
print(classification_report(y, preds))
```

```
              precision    recall  f1-score   support

           0       1.00      0.95      0.98    458239
           1       0.09      0.97      0.17      2155

    accuracy                           0.95    460394
   macro avg       0.55      0.96      0.57    460394
weighted avg       1.00      0.95      0.97    460394
```

In [97]:
```
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay
```

In [98]:
```
cm = confusion_matrix(y,preds)
```

In [99]:
```
print("Confusion Matrix: \n",cm)
```

```
Confusion Matrix: 
 [[437307  20932]
 [    67   2088]]
```

In [100]:
```
pip install pypandoc
```

```
Requirement already satisfied: pypandoc in c:\users\hithesha\anaconda3\lib\site-packages (1.15)Note: you may ne
ed to restart the kernel to use updated packages.
```

In [ ]:
```

```