Import Libraries

In [1]:

```
import seaborn as sns
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np


from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, OneHotEncoder
from sklearn.metrics import classification_report, confusion_matrix, roc_auc
from sklearn.linear_model import LogisticRegression
from sklearn.pipeline import Pipeline
from sklearn.compose import ColumnTransformer
import joblib


import warnings
warnings.filterwarnings('ignore')

plt.style.use('dark_background')

sns.set_style("darkgrid", {
    'axes.facecolor': '#111111',
    'figure.facecolor': '#111111',
    'axes.edgecolor': '#444444',
    'grid.color': '#333333',
    'text.color': 'white',
    'xtick.color': 'white',
    'ytick.color': 'white',
    'axes.labelcolor': 'white',
    'axes.grid': True,
})
```

Get Data

In [2]:

```
df = pd.read_csv("/kaggle/input/fraud-detection-dataset/AIML Dataset.csv")
```

In [3]:

```
df.sample(5)
```

Out[3]:

	step	type	amount	nameOrig	oldbalanceOrg	newbalanc
6356498	710	PAYMENT	13296.81	C2109162034	14575.0	12
965171	44	PAYMENT	311.29	C175453285	21403.0	210
2193004	185	PAYMENT	12815.93	C1383483588	151792.0	1389
735218	38	CASH_OUT	241549.25	C1559724887	77400.0	
3859312	283	TRANSFER	166418.85	C1607097218	0.0	

In [4]:

```
df.shape
```

Out[4]:

```
(6362620, 11)
```

In [5]:

```
df.describe().T
```

Out[5]:

	count	mean	std	min	25%	
step	6362620.0	2.433972e+02	1.423320e+02	1.0	156.00	23
amount	6362620.0	1.798619e+05	6.038582e+05	0.0	13389.57	7487
oldbalanceOrg	6362620.0	8.338831e+05	2.888243e+06	0.0	0.00	1420
newbalanceOrig	6362620.0	8.551137e+05	2.924049e+06	0.0	0.00	
oldbalanceDest	6362620.0	1.100702e+06	3.399180e+06	0.0	0.00	13270
newbalanceDest	6362620.0	1.224996e+06	3.674129e+06	0.0	0.00	21466
isFraud	6362620.0	1.290820e-03	3.590480e-02	0.0	0.00	
isFlaggedFraud	6362620.0	2.514687e-06	1.585775e-03	0.0	0.00	

In [6]:

```
df.describe(include='object').T
```

Out[6]:

	count	unique	top	freq
type	6362620	5	CASH_OUT	2237500
nameOrig	6362620	6353307	C1902386530	3
nameDest	6362620	2722362	C1286084959	113

In [7]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6362620 entries, 0 to 6362619
Data columns (total 11 columns):
 #   Column          Dtype  
---  ------          -----  
 0   step            int64  
 1   type            object 
 2   amount          float64
 3   nameOrig        object 
 4   oldbalanceOrg   float64
 5   newbalanceOrig  float64
 6   nameDest        object 
 7   oldbalanceDest  float64
 8   newbalanceDest  float64
 9   isFraud         int64  
 10  isFlaggedFraud  int64  
dtypes: float64(5), int64(3), object(3)
memory usage: 534.0+ MB
```

In [8]:

```
list(df.columns)
```

Out[8]:

```
['step',
 'type',
 'amount',
 'nameOrig',
 'oldbalanceOrg',
 'newbalanceOrig',
 'nameDest',
 'oldbalanceDest',
 'newbalanceDest',
 'isFraud',
 'isFlaggedFraud']
```

In [9]:

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

Out[9]:

```
isFraud
0    6354407
1       8213
Name: count, dtype: int64
```

In [10]:

```
df['isFlaggedFraud'].value_counts()
```

Out[10]:

```
isFlaggedFraud
0    6362604
1         16
Name: count, dtype: int64
```

In [11]:

```
df.isna().sum()
```

Out[11]:

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

EDA

In [12]:

```
round((df['isFraud'].value_counts()[1]/df.shape[0])*100, 3) # there is a cla
```

Out[12]:

```
0.129
```

In [13]:

```
df.columns
```

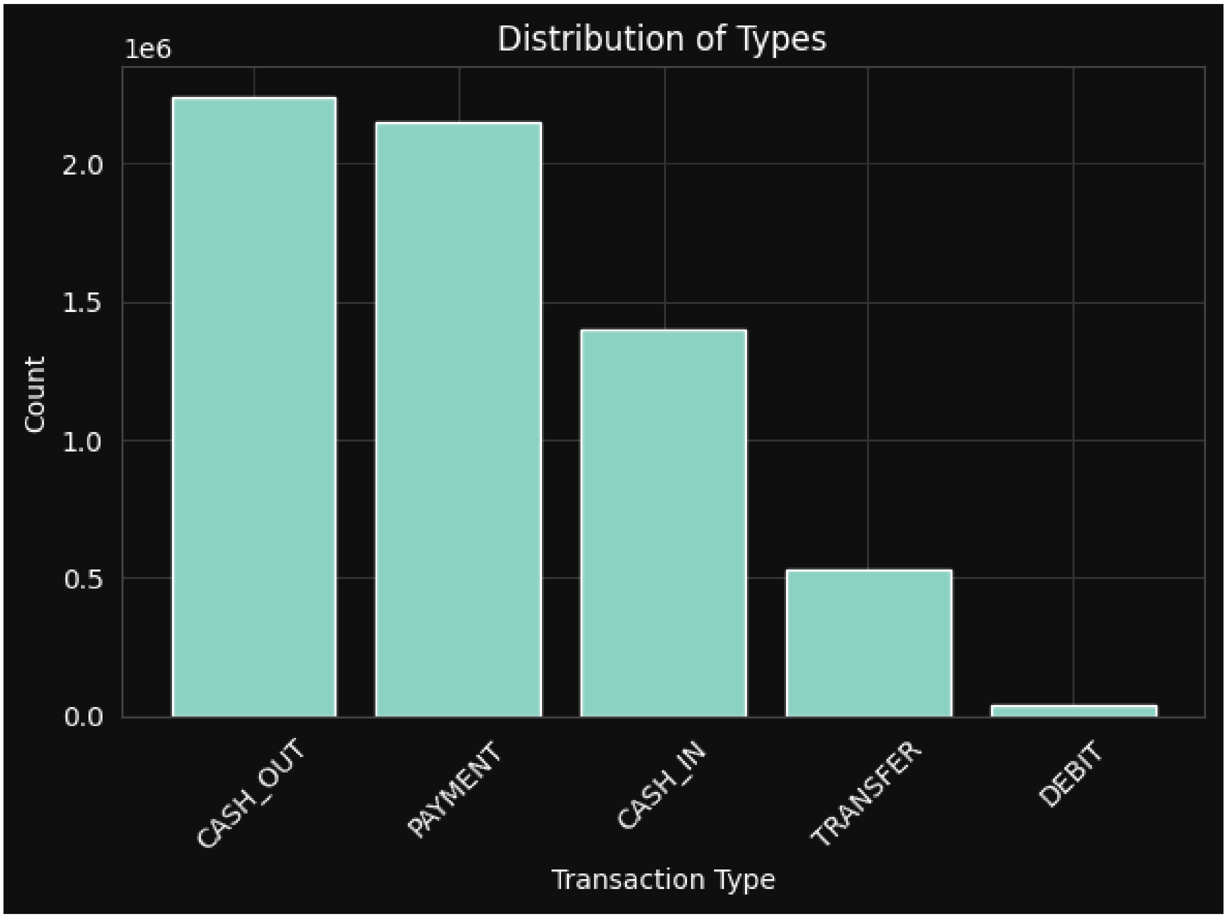
Out[13]:

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

In [14]:

```
type_counts = df['type'].value_counts()

plt.bar(type_counts.index, type_counts.values)
plt.xlabel('Transaction Type')
plt.ylabel('Count')
plt.title('Distribution of Types')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```



In [15]:

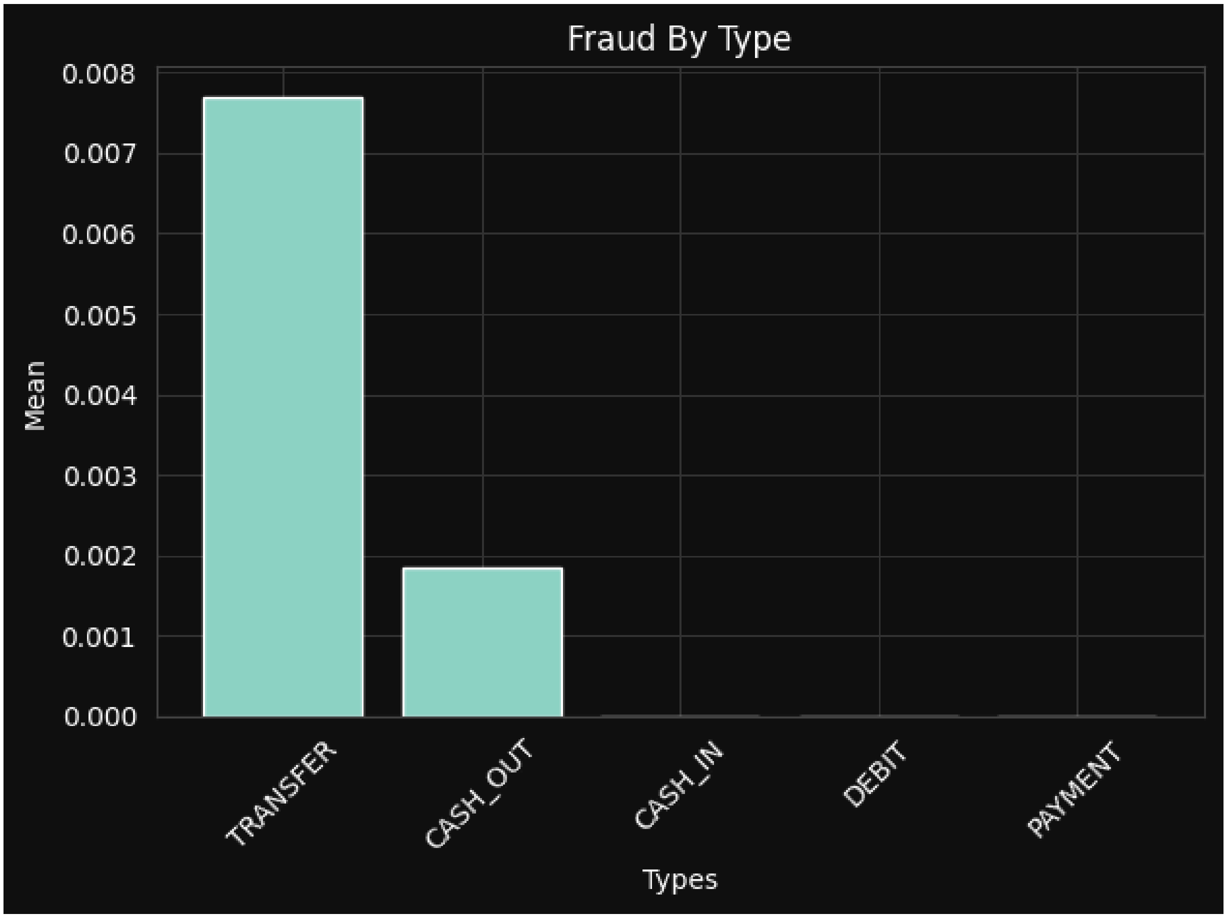
```
fraud_by_type = df.groupby('type')['isFraud'].mean().sort_values(ascending=F
fraud_by_type
```

Out[15]:

```
type
TRANSFER    0.007688
CASH_OUT    0.001840
CASH_IN     0.000000
DEBIT       0.000000
PAYMENT     0.000000
Name: isFraud, dtype: float64
```

In [16]:

```
plt.bar(fraud_by_type.index, fraud_by_type.values)
plt.xlabel('Types')
plt.ylabel('Mean')
plt.title('Fraud By Type')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```



In [17]:

```
df['amount'].describe().astype(int)
```

Out[17]:

```
count     6362620
mean       179861
std        603858
min             0
25%         13389
50%         74871
75%        208721
max      92445516
Name: amount, dtype: int64
```

In [18]:

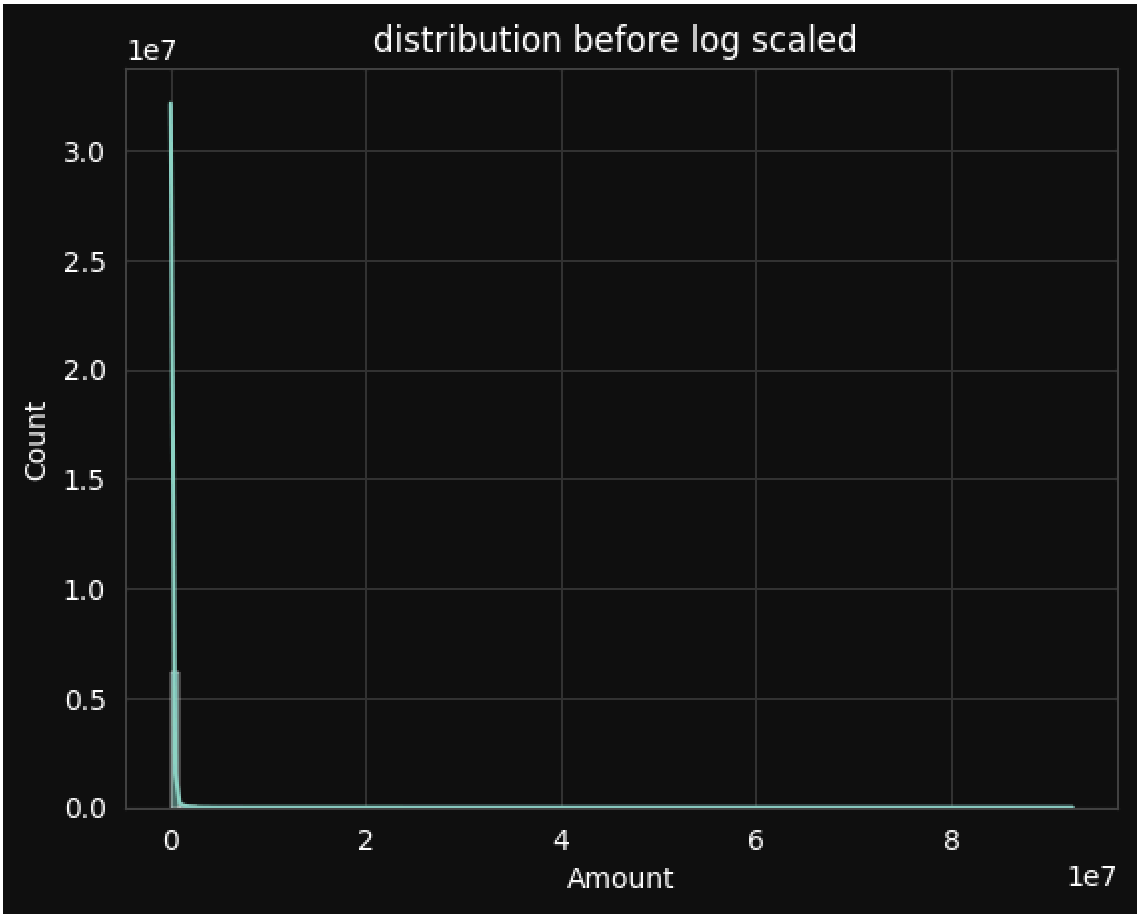
```
df['amount'].median()
```

Out[18]:

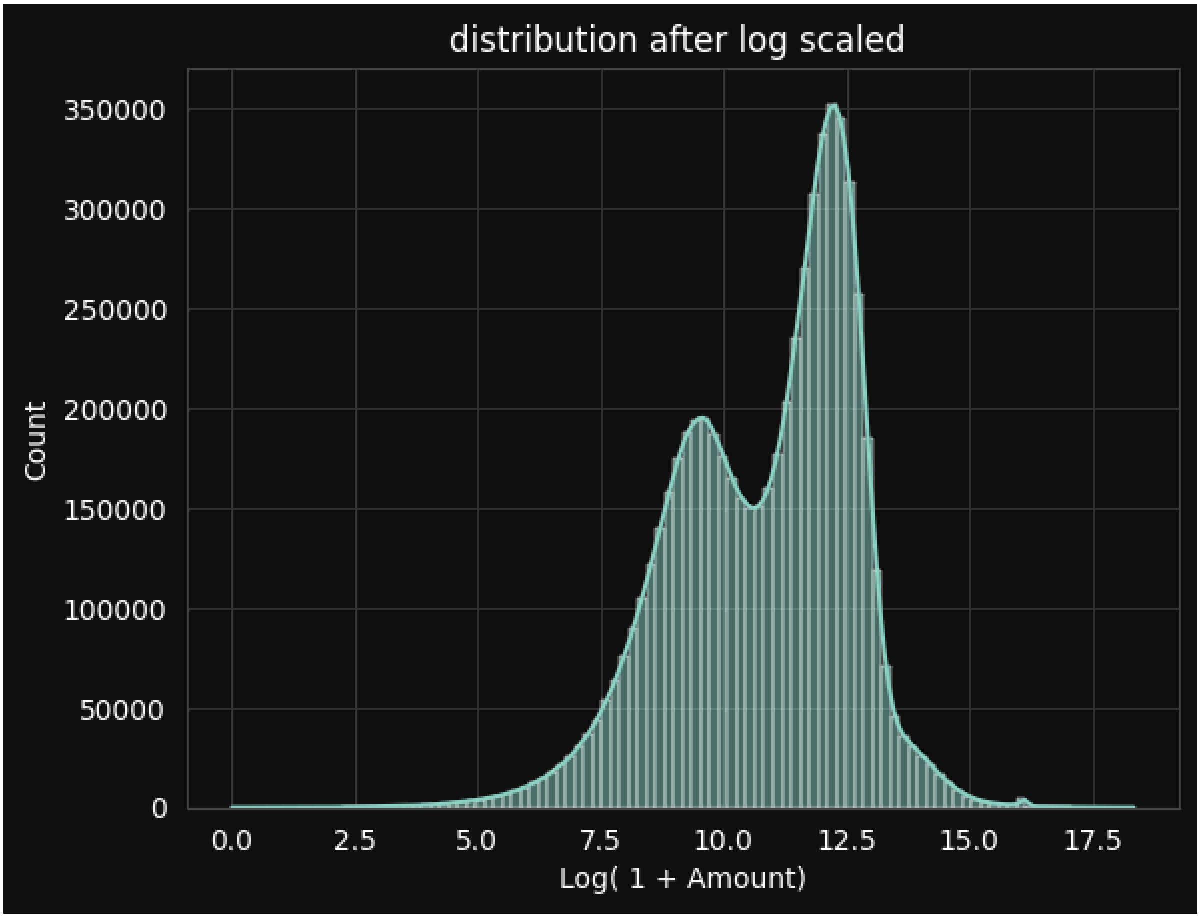
```
74871.94
```

In [19]:

```
sns.histplot(df['amount'], bins = 100, kde = True)
plt.title("distribution before log scaled ")
plt.xlabel("Amount")
plt.show()
```

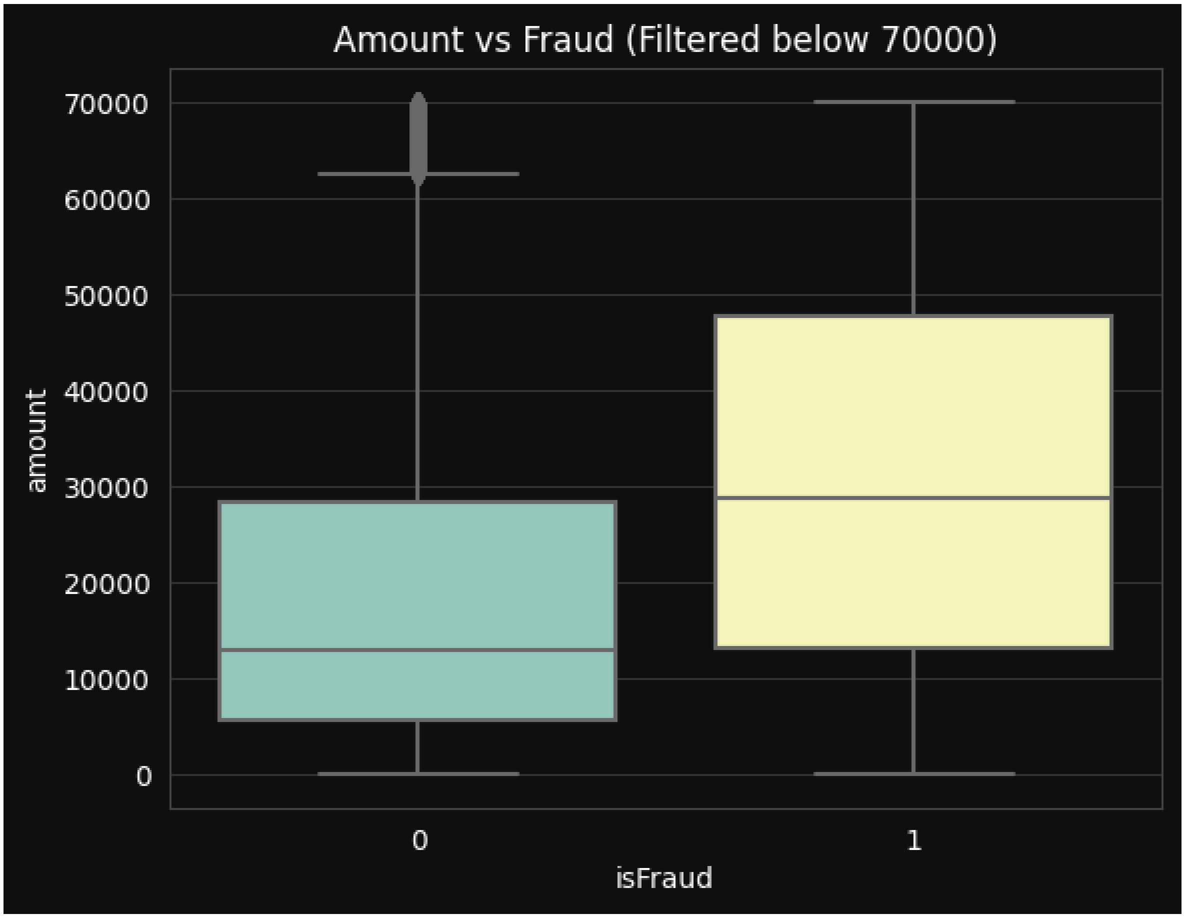


```
In [20]: sns.histplot(np.log1p(df['amount']), bins = 100, kde = True)
         plt.title("distribution after log scaled ")
         plt.xlabel("Log( 1 + Amount)")
         plt.show()
```



In [21]:

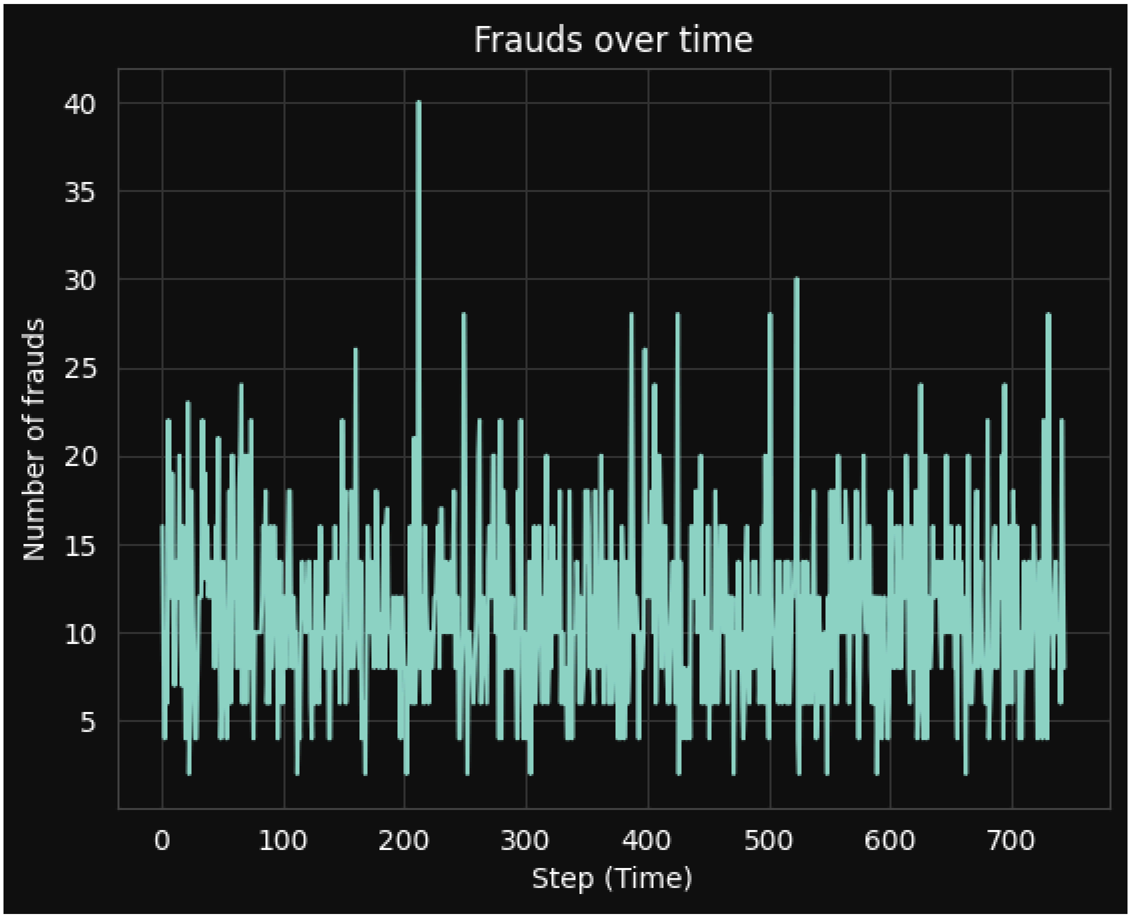
```
sns.boxplot(data = df[df['amount'] < 70000], x = 'isFraud', y = 'amount')
plt.title("Amount vs Fraud (Filtered below 70000)")
plt.show()
```



```
In [22]: df['balanceDiffOrig'] = df['oldbalanceOrg'] - df['newbalanceOrig']
         df['balanceDiffDest'] = df['oldbalanceDest'] - df['newbalanceDest']
```

```
In [23]: frauds_per_Step = df[df['isFraud'] == 1]['step'].value_counts().sort_index()

         plt.plot(frauds_per_Step.index, frauds_per_Step.values, label = "fraud per s
         plt.title("Frauds over time")
         plt.xlabel("Step (Time)")
         plt.ylabel("Number of frauds")
         plt.grid(True)
         plt.show()
         # we gain that it is time independent so let's drop it
```



```
In [24]: df.drop(columns = 'step', inplace=True)
```

```
In [25]: top_senders = df['nameOrig'].value_counts().head(10)
         top_senders
```

```
Out[25]: nameOrig
         C1902386530    3
         C363736674     3
         C545315117     3
         C724452879     3
         C1784010646    3
         C1677795071    3
         C1462946854    3
         C1999539787    3
         C2098525306    3
         C400299098     3
         Name: count, dtype: int64
```

```
In [26]: top_recievers = df['nameDest'].value_counts().head(10)
         top_recievers
```

Out[26]:

```
nameDest
C1286084959    113
C985934102     109
C665576141     105
C2083562754    102
C248609774     101
C1590550415    101
C451111351      99
C1789550256     99
C1360767589     98
C1023714065     97
Name: count, dtype: int64
```

In [27]:

```python
fraud_users = df[df['isFraud'] == 1]['nameDest'].value_counts().head(10)
fraud_users
```

Out[27]:

```
nameDest
C1193568854    2
C104038589     2
C200064275     2
C1497532505    2
C1601170327    2
C1655359478    2
C2020337583    2
C1653587362    2
C1013511446    2
C2129197098    2
Name: count, dtype: int64
```

In [28]:

```python
transfer_and_cash_out_df = df[df['type'].isin(['TRANSFER', 'CASH_OUT'])]
```

In [29]:

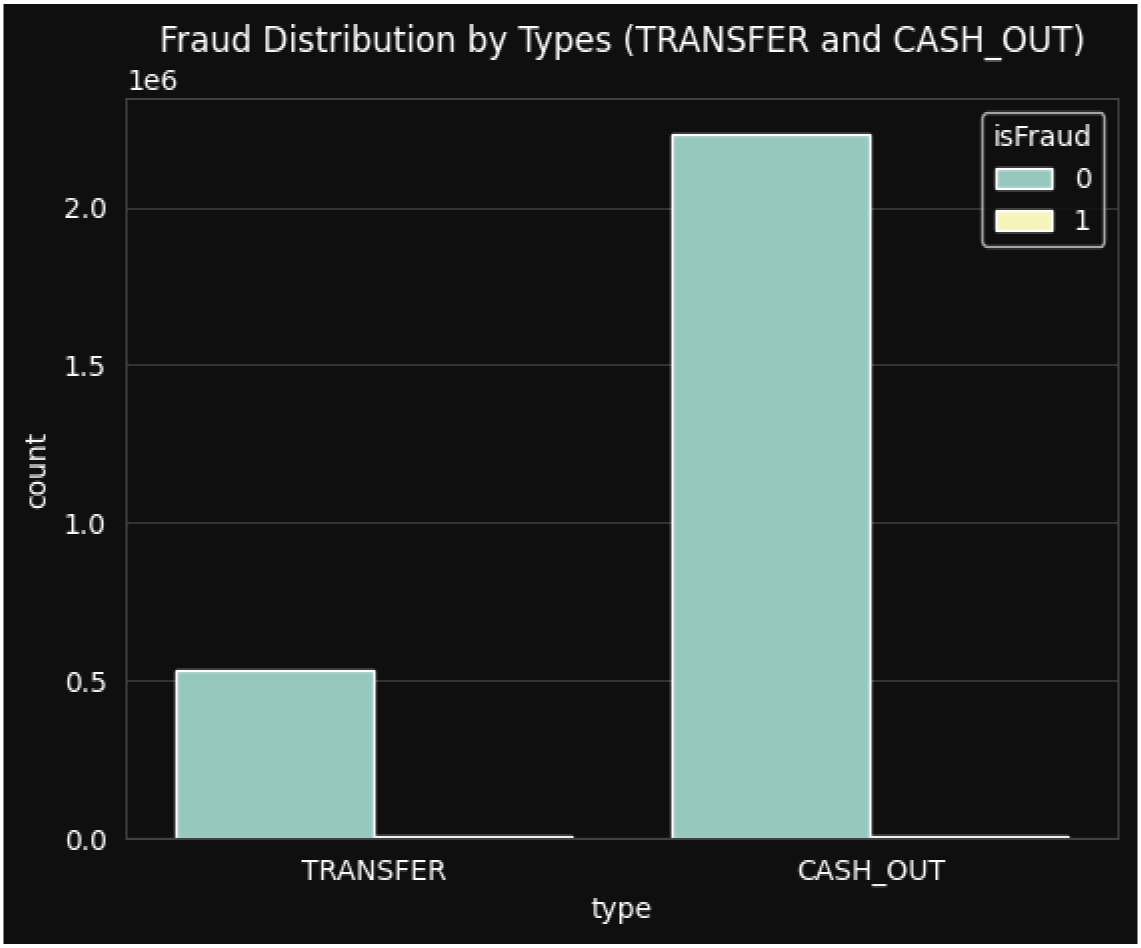
```python
transfer_and_cash_out_df['type'].value_counts()
```

Out[29]:

```
type
CASH_OUT    2237500
TRANSFER     532909
Name: count, dtype: int64
```

In [30]:

```python
sns.countplot(data = transfer_and_cash_out_df, x ='type', hue = 'isFraud')
plt.title("Fraud Distribution by Types (TRANSFER and CASH_OUT)")
plt.show()
```



In [31]:

```
corr = df[['amount', 'oldbalanceOrg', 'newbalanceOrig', 'oldbalanceDest', 'r
```

In [32]:

```
corr
```

Out[32]:

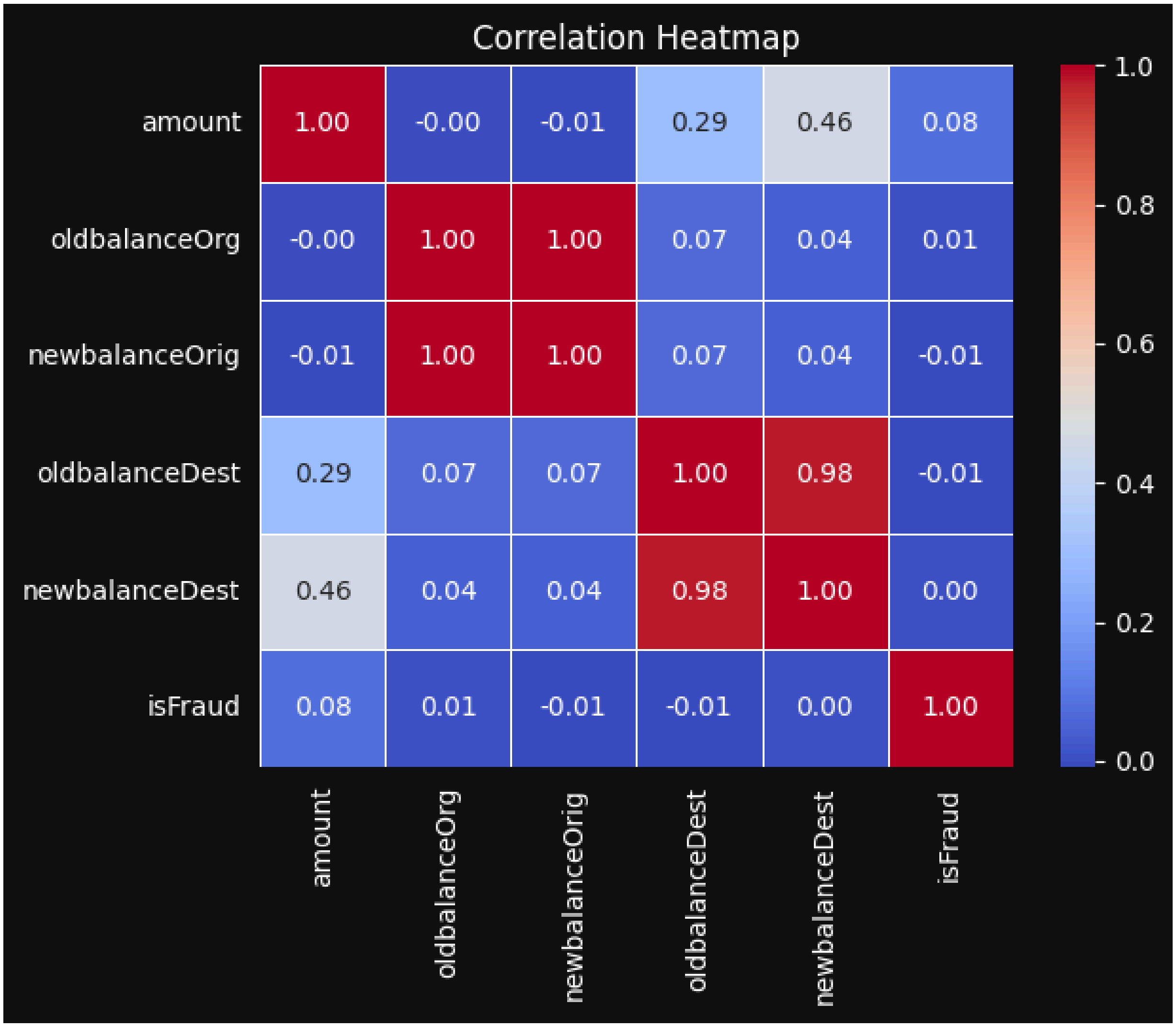
	amount	oldbalanceOrg	newbalanceOrig	oldbalanceDest
amount	1.000000	-0.002762	-0.007861	0.294137
oldbalanceOrg	-0.002762	1.000000	0.998803	0.066243
newbalanceOrig	-0.007861	0.998803	1.000000	0.067812
oldbalanceDest	0.294137	0.066243	0.067812	1.000000
newbalanceDest	0.459304	0.042029	0.041837	0.976569
isFraud	0.076688	0.010154	-0.008148	-0.005885

In [33]:

```
sns.heatmap(corr, annot=True, cmap='coolwarm', fmt='.2f', linewidths=0.5)
plt.title("Correlation Heatmap")
```

Out[33]:

```
Text(0.5, 1.0, 'Correlation Heatmap')
```



In [34]:
```
zero_after_transfer = df[
    (df['oldbalanceOrg'] > 0) &
    (df['newbalanceOrig'] == 0) &
    (df['type'].isin(['TRANSFER', 'CASH_OUT']))
]
```

In [35]:
```
len(zero_after_transfer)
```

Out[35]:
```
1188074
```

In [36]:
```
zero_after_transfer.sample(5)
```

Out[36]:

	type	amount	nameOrig	oldbalanceOrg	newbalanceOrig
1705176	CASH_OUT	51142.19	C1884221956	11617.0	0.0
134577	CASH_OUT	356393.44	C287904142	55794.3	0.0
2416276	CASH_OUT	188927.18	C1836064495	20519.0	0.0
5807692	CASH_OUT	438989.82	C946178941	1563.0	0.0
1438060	CASH_OUT	192569.64	C1247327890	3125.0	0.0

Modeling

In [37]:
```
df.head()
```

Out[37]:

	type	amount	nameOrig	oldbalanceOrg	newbalanceOrig	name
0	PAYMENT	9839.64	C1231006815	170136.0	160296.36	M197978
1	PAYMENT	1864.28	C1666544295	21249.0	19384.72	M204428
2	TRANSFER	181.00	C1305486145	181.0	0.00	C55326
3	CASH_OUT	181.00	C840083671	181.0	0.00	C3899
4	PAYMENT	11668.14	C2048537720	41554.0	29885.86	M123070

In [38]:
```
df_modeling = df.drop(columns=['nameOrig', 'nameDest', 'isFlaggedFraud'])
```

In [39]:
```
df_modeling.head()
```

Out[39]:

	type	amount	oldbalanceOrg	newbalanceOrig	oldbalanceDest	newb
0	PAYMENT	9839.64	170136.0	160296.36	0.0	
1	PAYMENT	1864.28	21249.0	19384.72	0.0	
2	TRANSFER	181.00	181.0	0.00	0.0	
3	CASH_OUT	181.00	181.0	0.00	21182.0	
4	PAYMENT	11668.14	41554.0	29885.86	0.0	

In [40]:
```
cat = ['type']
num = ['amount', 'oldbalanceOrg', 'newbalanceOrig', 'oldbalanceDest', 'newba
```

In [41]:
```
x = df_modeling.drop(columns='isFraud')
y = df_modeling['isFraud']
```

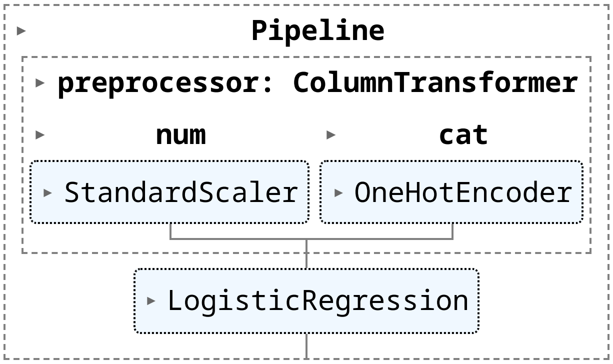
In [42]:
```
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.3, ran
```

In [43]:
```
preprocessor = ColumnTransformer(
    transformers=[
        ('num', StandardScaler(), num),
        ('cat', OneHotEncoder(), cat)
    ]
)
```

In [44]:
```
pipeline  = Pipeline([
    ('preprocessor', preprocessor),
    ('classifier', LogisticRegression(max_iter=1000, random_state=42, class_
])
```

In [45]:
```
pipeline.fit(x_train, y_train)
```

Out[45]:



In [46]:
```
ypred = pipeline.predict(x_test)
```

In [47]:
```
print(classification_report(y_test, ypred))
```

```
              precision    recall  f1-score   support

           0       1.00      0.95      0.97   1906322
           1       0.02      0.94      0.04      2464

    accuracy                           0.95   1908786
   macro avg       0.51      0.94      0.51   1908786
weighted avg       1.00      0.95      0.97   1908786
```

In [48]:
```
pipeline.score(x_test, y_test)
```

Out[48]:
```
0.9470490667890481
```

In [49]:
```
joblib.dump(pipeline, 'fraud_detection_model.pkl')
```

Out[49]:
```
['fraud_detection_model.pkl']
```

In [ ]:

This notebook was converted with convert.ploomber.io