Accredian File

September 17, 2023

[1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier, plot_tree
from sklearn.ensemble import RandomForestClassifier
import xgboost as xgb
from sklearn.tree import plot_tree
from sklearn.metrics import accuracy_score, precision_score, recall_score,␣
↪f1_score, confusion_matrix, classification_report
```

1 Data ingestion

[2]:
```
df=pd.read_csv('Fraud.csv')
```

2 1 Data Overviewing

[3]:
```
df.head()
```

[3]:
```
   step      type    amount     nameOrig  oldbalanceOrg  newbalanceOrig  \
0     1   PAYMENT   9839.64  C1231006815       170136.0       160296.36
1     1   PAYMENT   1864.28  C1666544295        21249.0        19384.72
2     1  TRANSFER    181.00  C1305486145          181.0            0.00
3     1  CASH_OUT    181.00   C840083671          181.0            0.00
4     1   PAYMENT  11668.14  C2048537720        41554.0        29885.86

      nameDest  oldbalanceDest  newbalanceDest  isFraud  isFlaggedFraud
0  M1979787155             0.0             0.0        0               0
1  M2044282225             0.0             0.0        0               0
2   C553264065             0.0             0.0        1               0
3    C38997010         21182.0             0.0        1               0
4  M1230701703             0.0             0.0        0               0
```

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

[4]: (6362620, 11)

[5]:
```
df.describe()
```

[5]:
```
               step        amount  oldbalanceOrg  newbalanceOrig  \
count  6.362620e+06  6.362620e+06   6.362620e+06    6.362620e+06
mean   2.433972e+02  1.798619e+05   8.338831e+05    8.551137e+05
std    1.423320e+02  6.038582e+05   2.888243e+06    2.924049e+06
min    1.000000e+00  0.000000e+00   0.000000e+00    0.000000e+00
25%    1.560000e+02  1.338957e+04   0.000000e+00    0.000000e+00
50%    2.390000e+02  7.487194e+04   1.420800e+04    0.000000e+00
75%    3.350000e+02  2.087215e+05   1.073152e+05    1.442584e+05
max    7.430000e+02  9.244552e+07   5.958504e+07    4.958504e+07

       oldbalanceDest  newbalanceDest       isFraud  isFlaggedFraud
count    6.362620e+06    6.362620e+06  6.362620e+06    6.362620e+06
mean     1.100702e+06    1.224996e+06  1.290820e-03    2.514687e-06
std      3.399180e+06    3.674129e+06  3.590480e-02    1.585775e-03
min      0.000000e+00    0.000000e+00  0.000000e+00    0.000000e+00
25%      0.000000e+00    0.000000e+00  0.000000e+00    0.000000e+00
50%      1.327057e+05    2.146614e+05  0.000000e+00    0.000000e+00
75%      9.430367e+05    1.111909e+06  0.000000e+00    0.000000e+00
max      3.560159e+08    3.561793e+08  1.000000e+00    1.000000e+00
```

[6]:
```
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
```

3 Task-1.1 Data Cleaning

[7]:
```
### Handle missing/null values
df.nunique()
```

[7]:
```
step                  743
type                    5
amount            5316900
nameOrig          6353307
oldbalanceOrg     1845844
newbalanceOrig    2682586
nameDest          2722362
oldbalanceDest    3614697
newbalanceDest    3555499
isFraud                 2
isFlaggedFraud          2
dtype: int64
```

[8]:
```
df.isna().sum().sum()
```

[8]:
```
0
```

[9]:
```
df.duplicated().sum()
```

[9]:
```
0
```

4 Task-1.2 Data Preprocessing

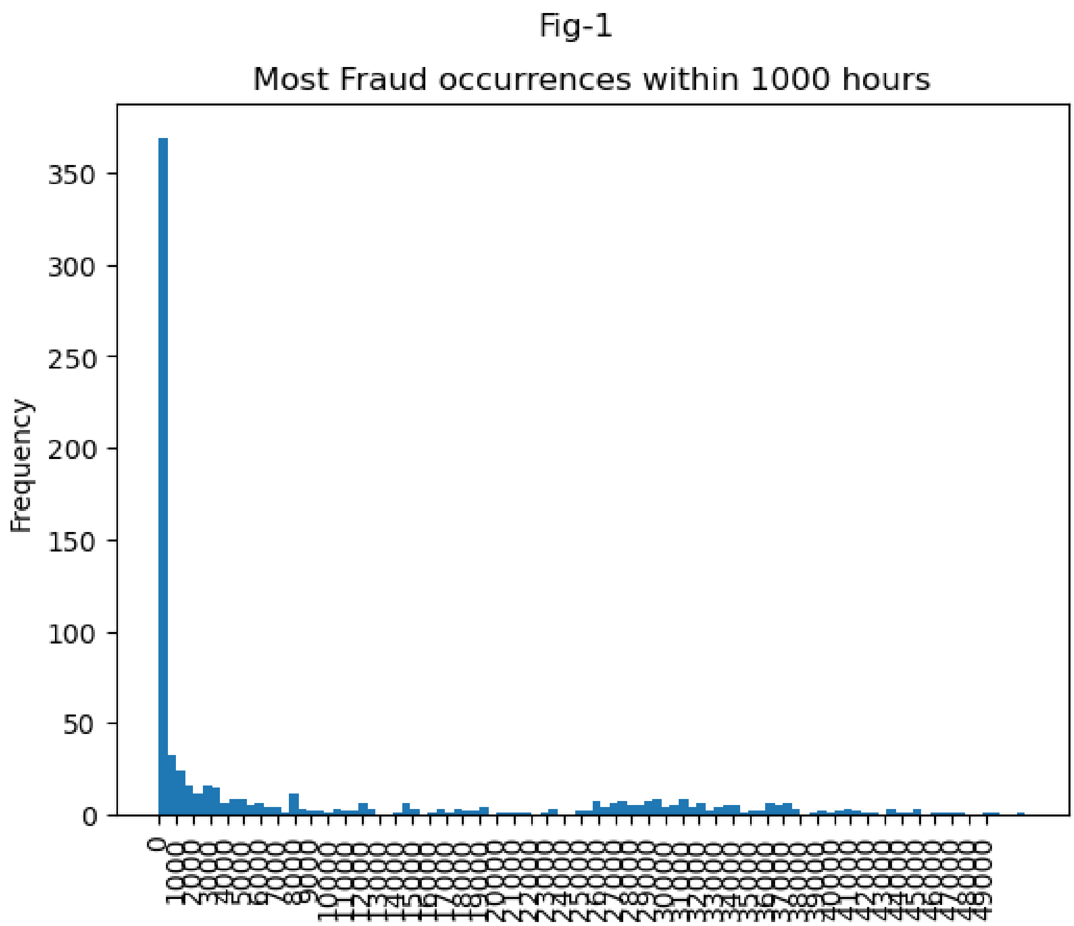
[10]:
```
cc = []      # Categorical Columns
nc = []      # numerical column

for i in df.columns:
    if df[i].dtype == 'object':
        cc.append(i)
    else:
        nc.append(i)
```

[11]:
```
cc.extend(['isFraud', 'isFlaggedFraud'])
nc.remove('isFraud')
nc.remove('isFlaggedFraud')
```

5 EDA

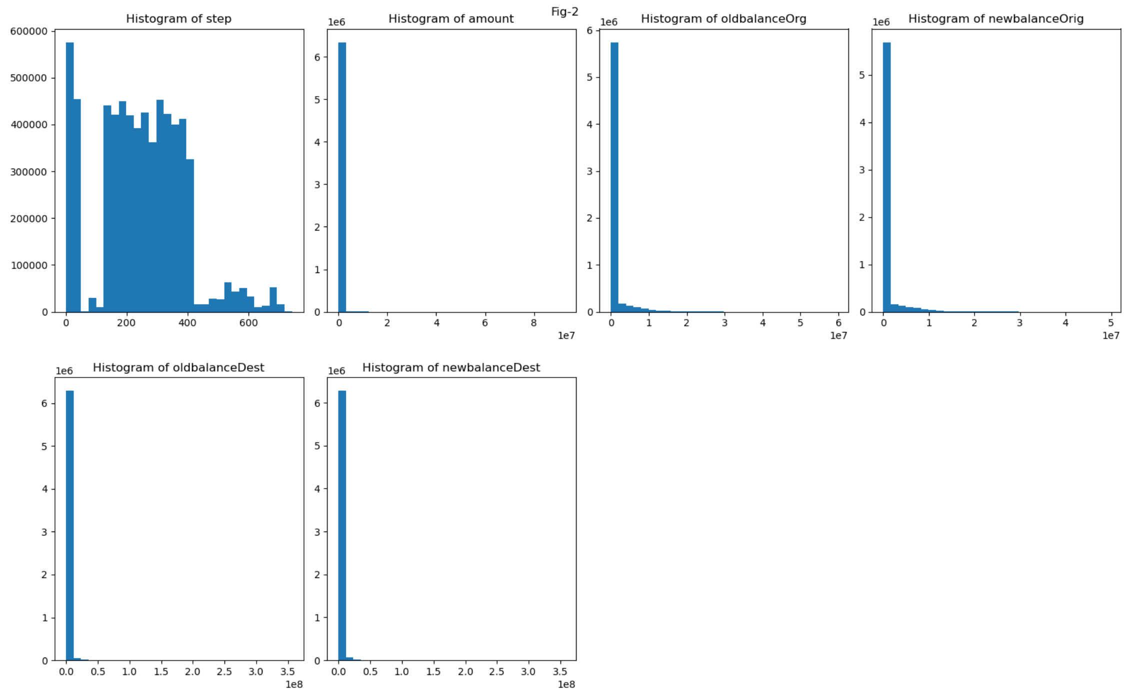
[12]:
```
df['step'].value_counts().plot(kind='hist', bins=100)
plt.xticks(range(0, 50000, 1000),rotation=90)
plt.suptitle("Distribution of hourly Fraud in Electronic Transactions")
```

```
plt.title("Most Fraud occurrences within 1000 hours")
plt.suptitle('Fig-1')
plt.show()
```



6 Historgam(Distribution of numerical columns)

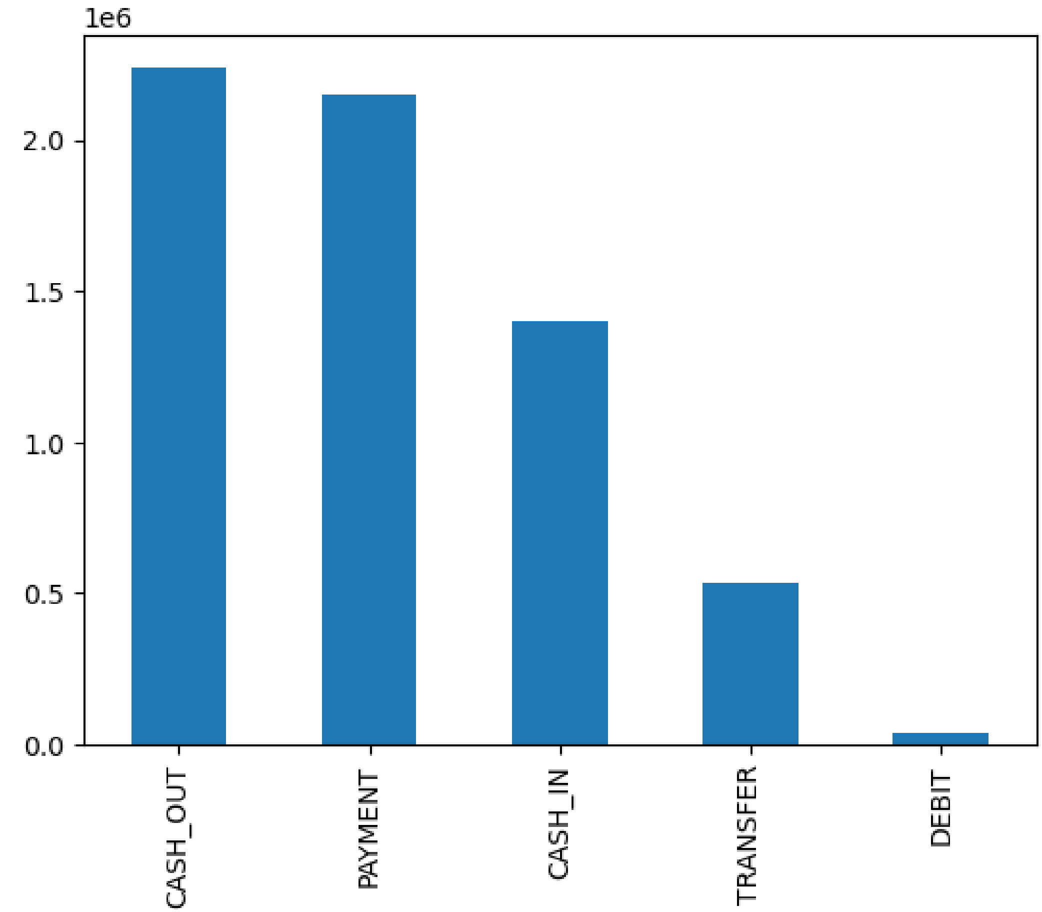
[13]:
```
plt.figure(figsize=(16, 10))
for i, col in enumerate(nc, start=1):
    plt.subplot(2, 4, i)
    plt.hist(df[col], bins=30)
    plt.title(f'Histogram of {col}')
plt.tight_layout()
plt.suptitle('Fig-2')
plt.show()
```



7 Share of Online Transaction type in fraud

[14]:
```
df['type'].value_counts().plot(kind='bar')
plt.suptitle('Fig-3')
plt.show()
```

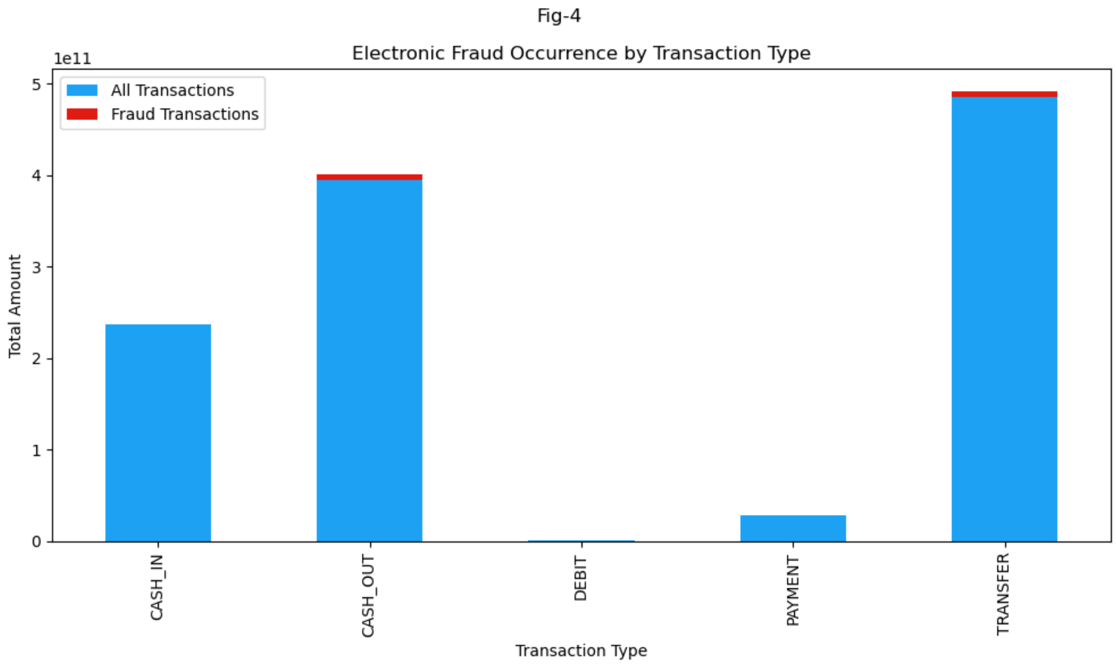
Fig-3



This graph reveals that electronic fraud incidents are concentrated exclusively in 'Cash Out' and 'Transfer' transaction types, highlighting potential areas of vulnerability

[15]:
```
df1 = df[df['isFraud'] == 1]
result_all = df.groupby('type')['amount'].sum()
result_fraud = df1.groupby('type')['amount'].sum()
# Combine both results into a single DataFrame
combined_result = pd.DataFrame({'All Transactions': result_all, 'Fraud␣
↪Transactions': result_fraud})
ax = combined_result.plot(kind='bar', stacked=True, figsize=(10, 6),␣
↪color=['#1DA1F2', '#DF1B12'])
plt.xlabel('Transaction Type')
plt.ylabel('Total Amount')
plt.title('Electronic Fraud Occurrence by Transaction Type')
plt.suptitle('Fig-4')
plt.tight_layout()
```

```
plt.show()
```



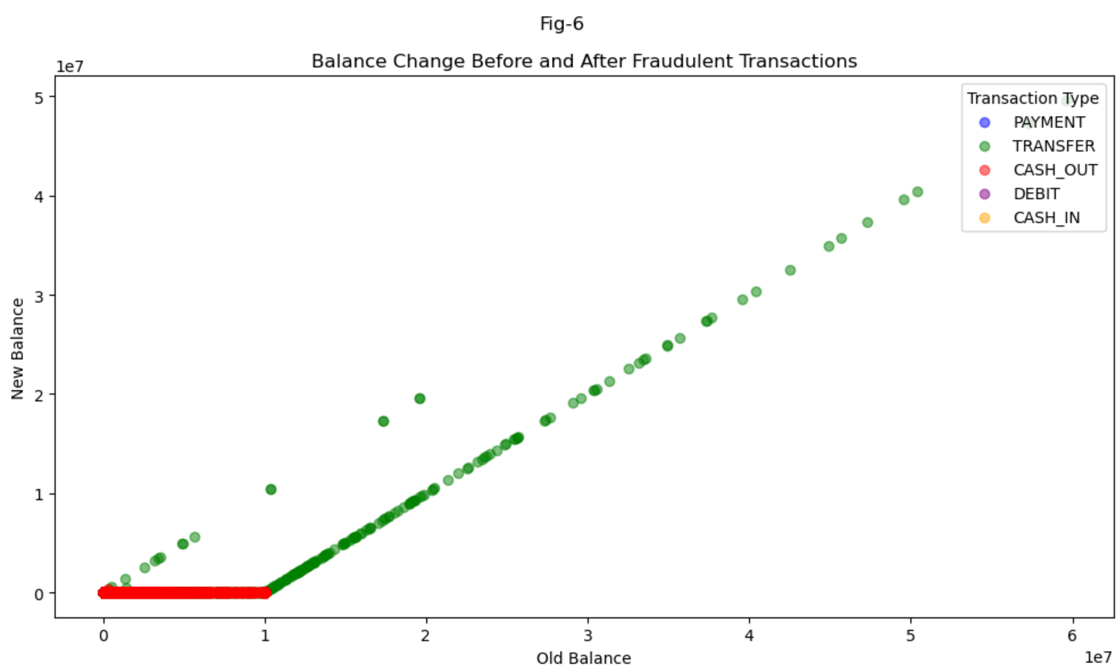
This graph illustrates the relationship between account balances before and after suspicious transactions

[16]:
```
# Assuming 'tf' is your DataFrame with columns 'oldbalanceDest',␣
↪'newbalanceDest', and 'isFraud'

# Create a scatter plot
tf=df
plt.figure(figsize=(10, 6))
plt.scatter(tf['oldbalanceOrg'], tf['newbalanceOrig'], c=tf['isFraud'],␣
↪cmap='coolwarm', alpha=0.5)
plt.xlabel('Old Balance')
plt.ylabel('New Balance')
plt.title('Balance Change Before and After Fraudulent Transactions')
plt.colorbar(label='Is Fraud')
plt.tight_layout()
plt.suptitle('Fig-5')
plt.show()
```

This plot illustrates that for fraudulent 'Cash Out' transactions, the new balance becomes zero, while for 'Transfer' transactions, the new balance decreases significantly but remains above zero. Fraudulent activities are concentrated in these two transaction types

[17]:
```
df_fraud = df1[df1['isFraud'] == 1]

# Create a unique color for each transaction type
type_colors = {
    'PAYMENT': 'blue',
    'TRANSFER': 'green',
    'CASH_OUT': 'red',
    'DEBIT': 'purple',
    'CASH_IN': 'orange'
    # Add more types and colors as needed
}

# Map transaction types to colors for the fraud DataFrame
df_fraud['type_color'] = df_fraud['type'].map(type_colors)

# Create a scatter plot using the mapped colors
plt.figure(figsize=(10, 6))
for transaction_type, color in type_colors.items():
    # Plot only the data corresponding to the current transaction type
    plt.scatter(
```

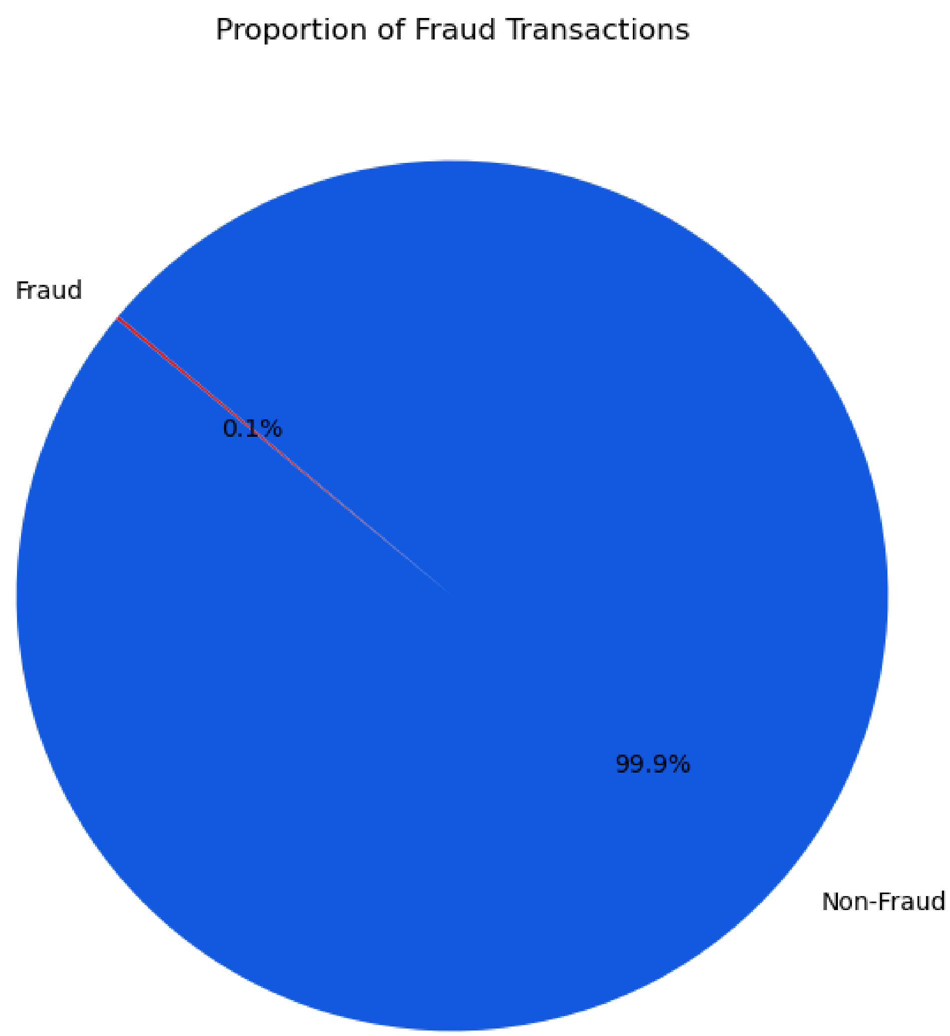
```
        df_fraud[df_fraud['type'] == transaction_type]['oldbalanceOrg'],
        df_fraud[df_fraud['type'] == transaction_type]['newbalanceOrig'],
        label=transaction_type,
        c=color,
        alpha=0.5
    )

# Add a legend
plt.legend(title='Transaction Type', loc='upper right')

plt.xlabel('Old Balance')
plt.ylabel('New Balance')
plt.title('Balance Change Before and After Fraudulent Transactions')
plt.suptitle('Fig-6')
#plt.suptitle("This plot illustrates that for fraudulent 'Cash Out'␣
 ↪transactions,the new balance becomes zero, while for 'Transfer'␣
 ↪transactions, the new balance decreases significantly but remains above zero.
 ↪ Fraudulent activities are concentrated in these two transaction types")
plt.tight_layout()

plt.show()
```



8 Percentage of fraud and not fraud

[18]:
```
fraud_count = df[df['isFraud'] == 1].shape[0]
total_count = df.shape[0]
fraud_labels = ['Fraud', 'Non-Fraud']
fraud_sizes = [fraud_count, total_count - fraud_count]
#colors = ['#1DA1F2', '#F5F3FA']

plt.figure(figsize=(8, 8))
plt.pie(fraud_sizes, labels=fraud_labels, colors=['#DF1B12', '#1259DF'],
 ↪autopct='%1.1f%%', startangle=140)
plt.title('Proportion of Fraud Transactions')
plt.suptitle('Fig-7')
plt.show()
```

Fig-7



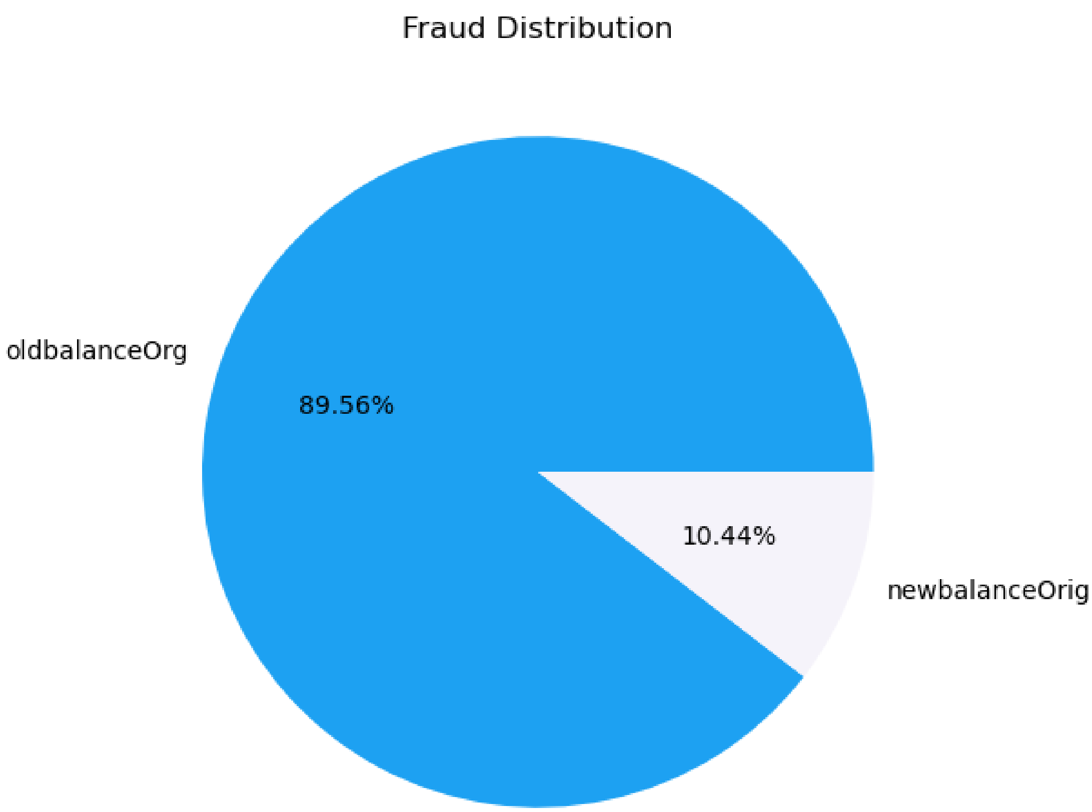
8.0.1 Distribution of Wealth Change Due to Fraudulent Transactions

Ther is drop of 88.34% of welth with respect to old balance

[19]:
```
tf=df[df['isFraud']==1]
sumamount=tf[['oldbalanceOrg','newbalanceOrig']].sum()
percentage_drop = ((tf['oldbalanceOrg'].sum() - tf['newbalanceOrig'].sum()) /␣
 ↪tf['oldbalanceOrg'].sum()) * 100
plt.figure(figsize=(6, 6))
```

```
plt.pie(sumamount, labels=sumamount.index, autopct='%1.2f%%',colors=['#1DA1F2',␣
↪'#F5F3FA'])
plt.title('Fraud Distribution')
plt.suptitle('Fig-8')
plt.show()
print(f"The percentage drop in wealth with respect to old balance is␣
↪approximately {percentage_drop:.2f}%.")
```

Fig-8

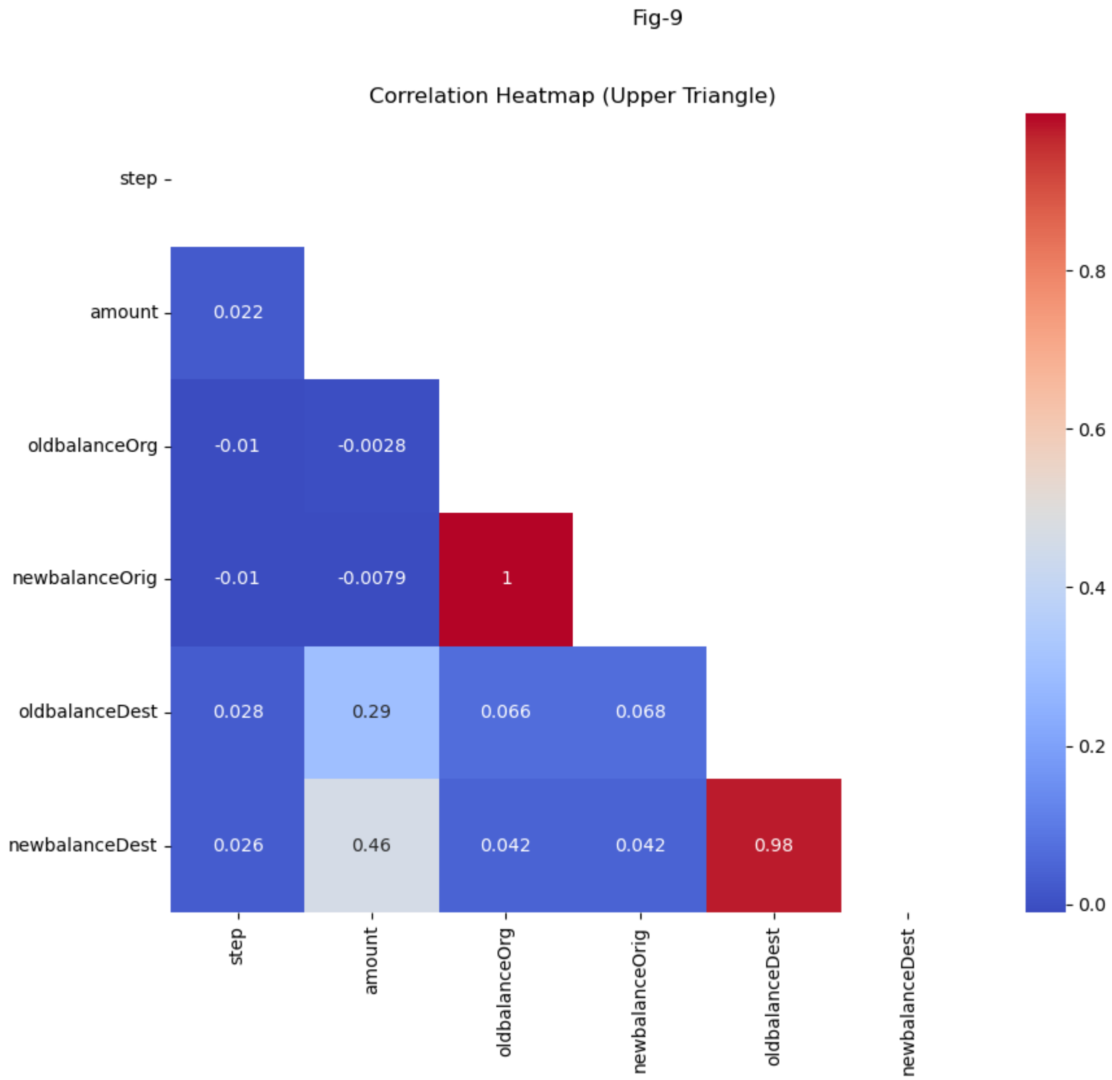



```
The percentage drop in wealth with respect to old balance is approximately
88.34%.
```

9 Correlation Matrix

[20]:
```
correlation_matrix = df[nc].corr()
mask = np.triu(np.ones_like(correlation_matrix, dtype=bool))
plt.figure(figsize=(10, 8))
```

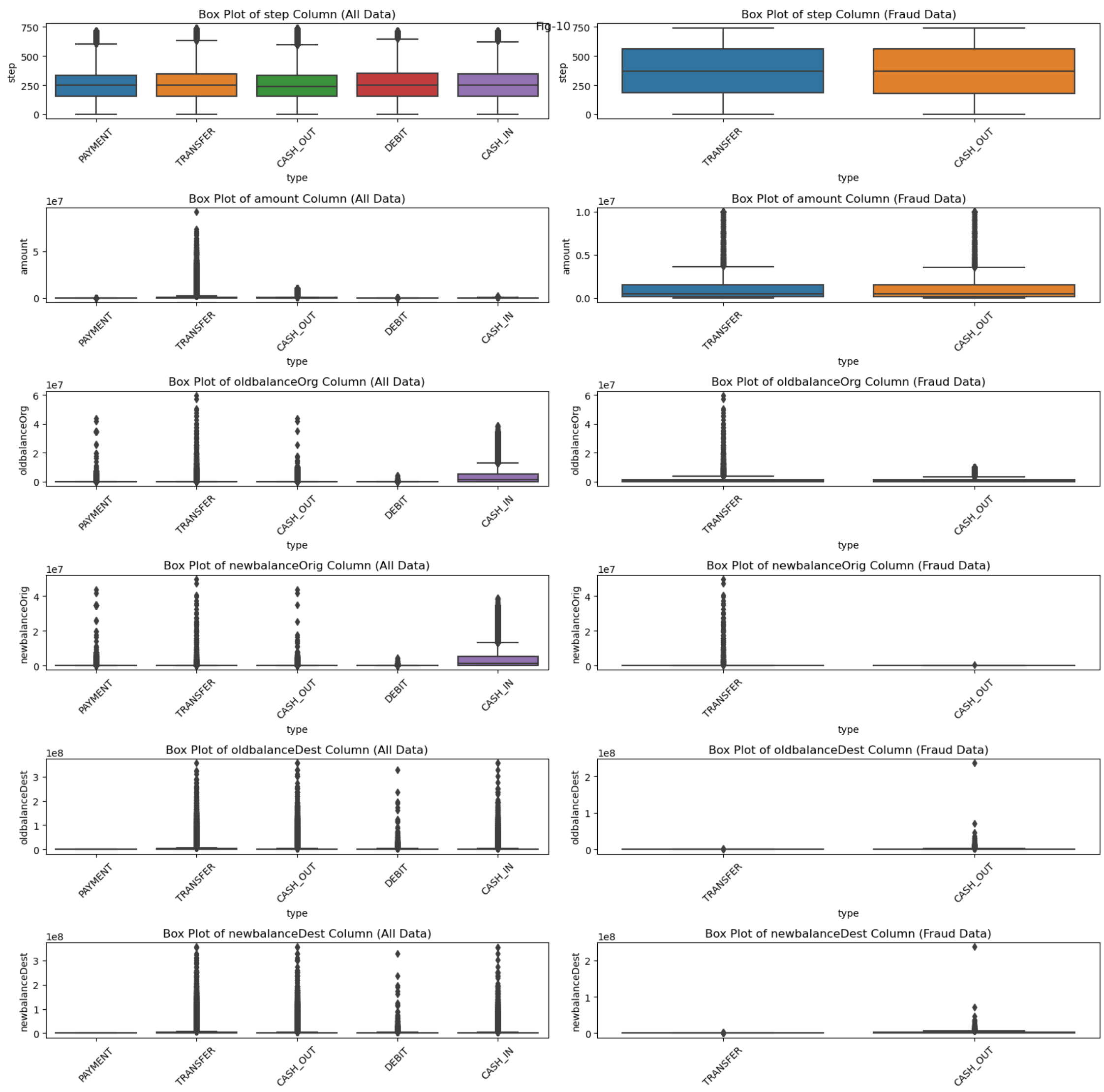
```
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', mask=mask)
plt.title('Correlation Heatmap (Upper Triangle)')
plt.suptitle('Fig-9')
plt.show()
```



9.1 Distribution of Numerical Features by Transaction Type (All Data vs. Fraud Data)

[21]:
```
fig, axs = plt.subplots(len(nc), 2, figsize=(16, 16))

for i, col in enumerate(nc):
    sns.boxplot(data=df, x='type', y=col, ax=axs[i, 0])
    axs[i, 0].set_title(f"Box Plot of {col} Column (All Data)")
```

```
    axs[i, 0].set_xticklabels(axs[i, 0].get_xticklabels(), rotation=45)

    # Filter the DataFrame for fraud data
    df_fraud = df[df['isFraud'] == 1]

    # Plot box plot for fraud data
    sns.boxplot(data=df_fraud, x='type', y=col, ax=axs[i, 1])
    axs[i, 1].set_title(f"Box Plot of {col} Column (Fraud Data)")
    axs[i, 1].set_xticklabels(axs[i, 1].get_xticklabels(), rotation=45)

plt.tight_layout()
plt.suptitle('Fig-10')
plt.show()
```



[22]:
```
df
```

[22]:
```
         step      type     amount     nameOrig  oldbalanceOrg  \
0           1   PAYMENT    9839.64  C1231006815      170136.00
1           1   PAYMENT    1864.28  C1666544295       21249.00
2           1  TRANSFER     181.00  C1305486145         181.00
3           1  CASH_OUT     181.00   C840083671         181.00
4           1   PAYMENT   11668.14  C2048537720       41554.00
…         …       …          …          …            …
6362615   743  CASH_OUT  339682.13   C786484425      339682.13
6362616   743  TRANSFER 6311409.28  C1529008245     6311409.28
6362617   743  CASH_OUT 6311409.28  C1162922333     6311409.28
6362618   743  TRANSFER  850002.52  C1685995037      850002.52
6362619   743  CASH_OUT  850002.52  C1280323807      850002.52

         newbalanceOrig     nameDest  oldbalanceDest  newbalanceDest  isFraud  \
0             160296.36  M1979787155            0.00            0.00        0
1              19384.72  M2044282225            0.00            0.00        0
2                  0.00   C553264065            0.00            0.00        1
3                  0.00    C38997010        21182.00            0.00        1
4              29885.86  M1230701703            0.00            0.00        0
…                   …          …            …             …       …
6362615            0.00   C776919290            0.00       339682.13        1
6362616            0.00  C1881841831            0.00            0.00        1
6362617            0.00  C1365125890        68488.84      6379898.11        1
6362618            0.00  C2080388513            0.00            0.00        1
6362619            0.00   C873221189      6510099.11      7360101.63        1

         isFlaggedFraud
0                     0
1                     0
2                     0
3                     0
4                     0
…                   …
6362615               0
6362616               0
6362617               0
6362618               0
6362619               0

[6362620 rows x 11 columns]
```

9.2 Feature Engineering

[23]:
```
dfn=df
dfn=dfn.drop(['nameOrig','nameDest','isFlaggedFraud'],axis=1)  #removing␣
↪unwanted columns
```

[24]:
```
dfn.columns
```

[24]:
```
Index(['step', 'type', 'amount', 'oldbalanceOrg', 'newbalanceOrig',
       'oldbalanceDest', 'newbalanceDest', 'isFraud'],
      dtype='object')
```

[25]:
```
# making it easier for machine learning models to identify patterns and␣
↪anomalies by adding log and square root.
dfn['log_amount']=np.log1p(dfn['amount'])
dfn['sqr_amount']=np.sqrt(dfn['amount'])


dfn['diff_org']=dfn['newbalanceOrig']-dfn['oldbalanceOrg']
dfn['diff_Dest']=dfn['newbalanceDest']-dfn['oldbalanceDest']


# Feature smoothing reduces data noise, aiding machine learning models in␣
↪recognizing
#meaningful patterns over short-term fluctuations.
dfn['amountmeanrolling3']=dfn['amount'].rolling(window=3).mean()
dfn['amountsumrolling7']=dfn['amount'].rolling(window=7).sum()


# create these new columns to capture potential patterns
dfn['amount+oldorg']=dfn['amount']*dfn['oldbalanceOrg']
dfn['amount+neworg']=dfn['amount']*dfn['newbalanceOrig']

# one hot encoding
dfn_e=pd.get_dummies(dfn['type'],prefix='type',drop_first=True)
dfn = pd.concat([dfn_e, dfn], axis=1)
```

[26]:
```
dfn   #Viewing Newly added featurs
```

[26]:

	type_CASH_OUT	type_DEBIT	type_PAYMENT	type_TRANSFER	step \
0	0	0	1	0	1
1	0	0	1	0	1
2	0	0	0	1	1
3	1	0	0	0	1
4	0	0	1	0	1
…	…	…	…	…	…
6362615	1	0	0	0	743
6362616	0	0	0	1	743

```
6362617               1            0             0               0   743
6362618               0            0             0               1   743
6362619               1            0             0               0   743

             type      amount  oldbalanceOrg  newbalanceOrig  oldbalanceDest  \
0         PAYMENT     9839.64      170136.00       160296.36            0.00
1         PAYMENT     1864.28       21249.00        19384.72            0.00
2        TRANSFER      181.00         181.00            0.00            0.00
3        CASH_OUT      181.00         181.00            0.00        21182.00
4         PAYMENT    11668.14       41554.00        29885.86            0.00
…           …          …             …              …              …
6362615  CASH_OUT   339682.13      339682.13            0.00            0.00
6362616  TRANSFER  6311409.28     6311409.28            0.00            0.00
6362617  CASH_OUT  6311409.28     6311409.28            0.00        68488.84
6362618  TRANSFER   850002.52      850002.52            0.00            0.00
6362619  CASH_OUT   850002.52      850002.52            0.00      6510099.11

         newbalanceDest  isFraud  log_amount   sqr_amount    diff_org  \
0                  0.00        0    9.194276    99.194960    -9839.64
1                  0.00        0    7.531166    43.177309    -1864.28
2                  0.00        1    5.204007    13.453624     -181.00
3                  0.00        1    5.204007    13.453624     -181.00
4                  0.00        0    9.364703   108.019165   -11668.14
…                    …       …         …           …           …
6362615       339682.13        1   12.735768   582.822554  -339682.13
6362616            0.00        1   15.657870  2512.251835 -6311409.28
6362617      6379898.11        1   15.657870  2512.251835 -6311409.28
6362618            0.00        1   13.652996   921.955812  -850002.52
6362619      7360101.63        1   13.652996   921.955812  -850002.52

          diff_Dest  amountmeanrolling3  amountsumrolling7  amount+oldorg  \
0              0.00                 NaN                NaN   1.674077e+09
1              0.00                 NaN                NaN   3.961409e+07
2              0.00        3.961640e+03                NaN   3.276100e+04
3         -21182.00        7.420933e+02                NaN   3.276100e+04
4              0.00        4.010047e+03                NaN   4.848579e+08
…             …                 …                  …               …
6362615   339682.13        6.460610e+05         3582191.30   1.153839e+11
6362616        0.00        2.330258e+06         9635245.16   3.983389e+13
6362617  6311409.27        4.320834e+06        15883237.45   3.983389e+13
6362618        0.00        4.490940e+06        16669822.98   7.225043e+11
6362619   850002.52        2.670471e+06        16261006.68   7.225043e+11

         amount+neworg
0         1.577258e+09
1         3.613855e+07
2         0.000000e+00
```

```
3          0.000000e+00
4          3.487124e+08
…                 …
6362615    0.000000e+00
6362616    0.000000e+00
6362617    0.000000e+00
6362618    0.000000e+00
6362619    0.000000e+00

[6362620 rows x 20 columns]
```

[27]:
```
dfn=dfn.dropna()
```

[28]:
```
dfncorr=dfn.corr()
```

```
C:\Users\Kundan Mourya\AppData\Local\Temp\ipykernel_4396\2371628715.py:1:
FutureWarning: The default value of numeric_only in DataFrame.corr is
deprecated. In a future version, it will default to False. Select only valid
columns or specify the value of numeric_only to silence this warning.
  dfncorr=dfn.corr()
```

[29]:
```
dfncorr
```

[29]:
```
                     type_CASH_OUT  type_DEBIT  type_PAYMENT  type_TRANSFER  \
type_CASH_OUT             1.000000   -0.059626     -0.526422      -0.222672
type_DEBIT               -0.059626    1.000000     -0.057868      -0.024478
type_PAYMENT             -0.526422   -0.057868      1.000000      -0.216109
type_TRANSFER            -0.222672   -0.024478     -0.216109       1.000000
step                     -0.012919    0.002869      0.004927       0.006926
amount                   -0.004376   -0.023379     -0.197444       0.365896
oldbalanceOrg            -0.200899   -0.021450     -0.189486      -0.081593
newbalanceOrig           -0.210978   -0.021872     -0.193915      -0.087355
oldbalanceDest            0.086028    0.009347     -0.231455       0.130476
newbalanceDest            0.093476    0.006346     -0.238315       0.191701
isFraud                   0.011254   -0.002910     -0.025694       0.053862
log_amount                0.340432   -0.130937     -0.735820       0.354747
sqr_amount                0.173215   -0.075616     -0.562094       0.515101
diff_org                 -0.250010   -0.013646     -0.134576      -0.134808
diff_Dest                 0.062758   -0.010400     -0.109286       0.320840
amountmeanrolling3       -0.010360   -0.017269     -0.171910       0.337824
amountsumrolling7        -0.010028   -0.014414     -0.163131       0.324055
amount+oldorg            -0.063959   -0.008280     -0.072851      -0.004318
amount+neworg            -0.108410   -0.012065     -0.106187      -0.035199

                         step    amount  oldbalanceOrg  newbalanceOrig  \
type_CASH_OUT       -0.012919 -0.004376      -0.200899       -0.210978
type_DEBIT           0.002869 -0.023379      -0.021450       -0.021872
type_PAYMENT         0.004927 -0.197444      -0.189486       -0.193915
```

type_TRANSFER	0.006926	0.365896	-0.081593	-0.087355
step	1.000000	0.022373	-0.010059	-0.010299
amount	0.022373	1.000000	-0.002763	-0.007861
oldbalanceOrg	-0.010059	-0.002763	1.000000	0.998803
newbalanceOrig	-0.010299	-0.007861	0.998803	1.000000
oldbalanceDest	0.027665	0.294137	0.066242	0.067811
newbalanceDest	0.025888	0.459304	0.042029	0.041837
isFraud	0.031596	0.076700	0.010158	-0.008147
log_amount	0.007374	0.387260	0.106980	0.111452
sqr_amount	0.013037	0.761408	0.048295	0.048499
diff_org	-0.007255	-0.102337	0.220297	0.267750
diff_Dest	0.001325	0.845964	-0.087032	-0.094456
amountmeanrolling3	0.032245	0.666454	-0.006214	-0.011236
amountsumrolling7	0.043479	0.465911	-0.010834	-0.014821
amount+oldorg	0.009109	0.140615	0.361253	0.339068
amount+neworg	0.001672	0.062757	0.521439	0.518760

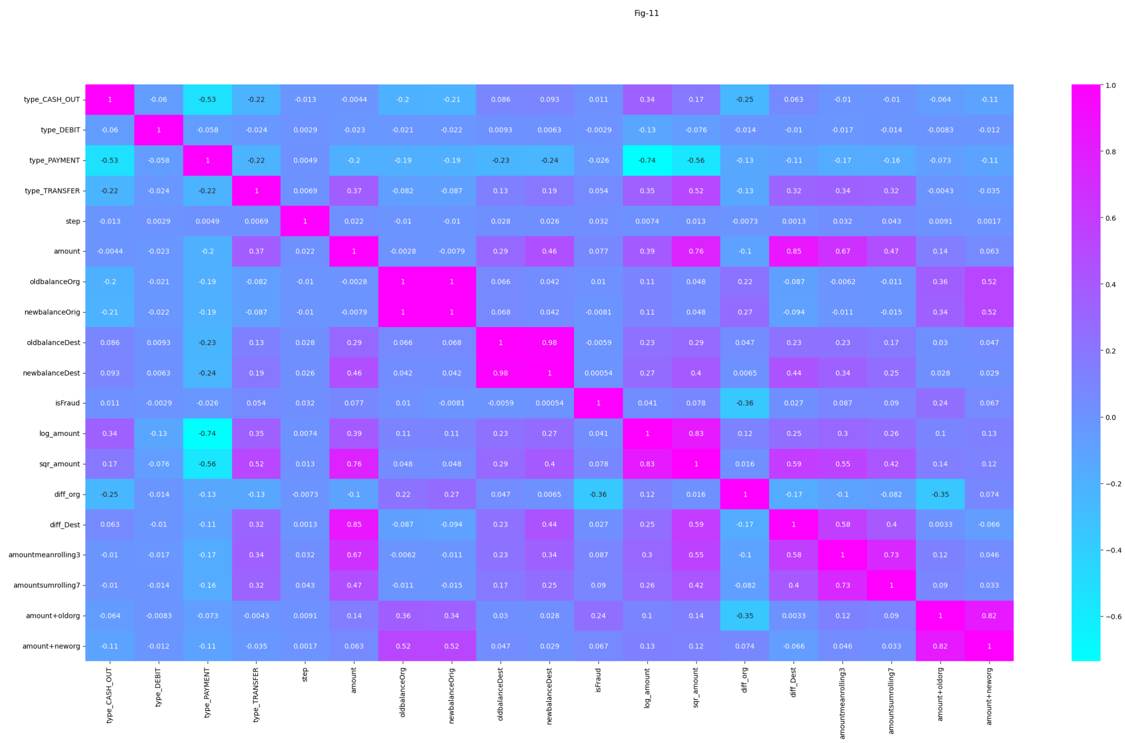
	oldbalanceDest	newbalanceDest	isFraud	log_amount \
type_CASH_OUT	0.086028	0.093476	0.011254	0.340432
type_DEBIT	0.009347	0.006346	-0.002910	-0.130937
type_PAYMENT	-0.231455	-0.238315	-0.025694	-0.735820
type_TRANSFER	0.130476	0.191701	0.053862	0.354747
step	0.027665	0.025888	0.031596	0.007374
amount	0.294137	0.459304	0.076700	0.387260
oldbalanceOrg	0.066242	0.042029	0.010158	0.106980
newbalanceOrig	0.067811	0.041837	-0.008147	0.111452
oldbalanceDest	1.000000	0.976569	-0.005883	0.227789
newbalanceDest	0.976569	1.000000	0.000538	0.266022
isFraud	-0.005883	0.000538	1.000000	0.040672
log_amount	0.227789	0.266022	0.040672	1.000000
sqr_amount	0.287169	0.396059	0.078083	0.828564
diff_org	0.047460	0.006451	-0.362515	0.115298
diff_Dest	0.232316	0.436191	0.027033	0.249841
amountmeanrolling3	0.226389	0.336735	0.087167	0.299610
amountsumrolling7	0.172984	0.249214	0.090157	0.255366
amount+oldorg	0.029967	0.028452	0.243875	0.100867
amount+neworg	0.046916	0.028875	0.066871	0.125567

	sqr_amount	diff_org	diff_Dest	amountmeanrolling3 \
type_CASH_OUT	0.173215	-0.250010	0.062758	-0.010360
type_DEBIT	-0.075616	-0.013646	-0.010400	-0.017269
type_PAYMENT	-0.562094	-0.134576	-0.109286	-0.171910
type_TRANSFER	0.515101	-0.134808	0.320840	0.337824
step	0.013037	-0.007255	0.001325	0.032245
amount	0.761408	-0.102337	0.845964	0.666454
oldbalanceOrg	0.048295	0.220297	-0.087032	-0.006214
newbalanceOrig	0.048499	0.267750	-0.094456	-0.011236

```
oldbalanceDest        0.287169  0.047460   0.232316             0.226389
newbalanceDest        0.396059  0.006451   0.436191             0.336735
isFraud               0.078083 -0.362515   0.027033             0.087167
log_amount            0.828564  0.115298   0.249841             0.299610
sqr_amount            1.000000  0.015851   0.589256             0.548342
diff_org              0.015851  1.000000  -0.169292            -0.101654
diff_Dest             0.589256 -0.169292   1.000000             0.575280
amountmeanrolling3    0.548342 -0.101654   0.575280             1.000000
amountsumrolling7     0.420259 -0.082157   0.403032             0.734990
amount+oldorg         0.143785 -0.354147   0.003287             0.120159
amount+neworg         0.117788  0.073904  -0.065673             0.045839

                    amountsumrolling7  amount+oldorg  amount+neworg
type_CASH_OUT               -0.010028      -0.063959      -0.108410
type_DEBIT                  -0.014414      -0.008280      -0.012065
type_PAYMENT                -0.163131      -0.072851      -0.106187
type_TRANSFER                0.324055      -0.004318      -0.035199
step                         0.043479       0.009109       0.001672
amount                       0.465911       0.140615       0.062757
oldbalanceOrg               -0.010834       0.361253       0.521439
newbalanceOrig              -0.014821       0.339068       0.518760
oldbalanceDest               0.172984       0.029967       0.046916
newbalanceDest               0.249214       0.028452       0.028875
isFraud                      0.090157       0.243875       0.066871
log_amount                   0.255366       0.100867       0.125567
sqr_amount                   0.420259       0.143785       0.117788
diff_org                    -0.082157      -0.354147       0.073904
diff_Dest                    0.403032       0.003287      -0.065673
amountmeanrolling3           0.734990       0.120159       0.045839
amountsumrolling7            1.000000       0.089535       0.032637
amount+oldorg                0.089535       1.000000       0.824067
amount+neworg                0.032637       0.824067       1.000000
```

[30]:

```python
plt.figure(figsize=(30,15))
sns.heatmap(dfncorr,annot=True,cmap='cool')
plt.suptitle('Fig-11')
plt.show()
```



9.3 Viewing most retated Feature with Fraud

[31]:
```python
corrwithtargetfeature=dfn.corr()['isFraud'].abs().sort_values(ascending=False)
corrwithtargetfeature
```

```
C:\Users\Kundan Mourya\AppData\Local\Temp\ipykernel_4396\3211210775.py:1:
FutureWarning: The default value of numeric_only in DataFrame.corr is
deprecated. In a future version, it will default to False. Select only valid
columns or specify the value of numeric_only to silence this warning.
  corrwithtargetfeature=dfn.corr()['isFraud'].abs().sort_values(ascending=False)
```

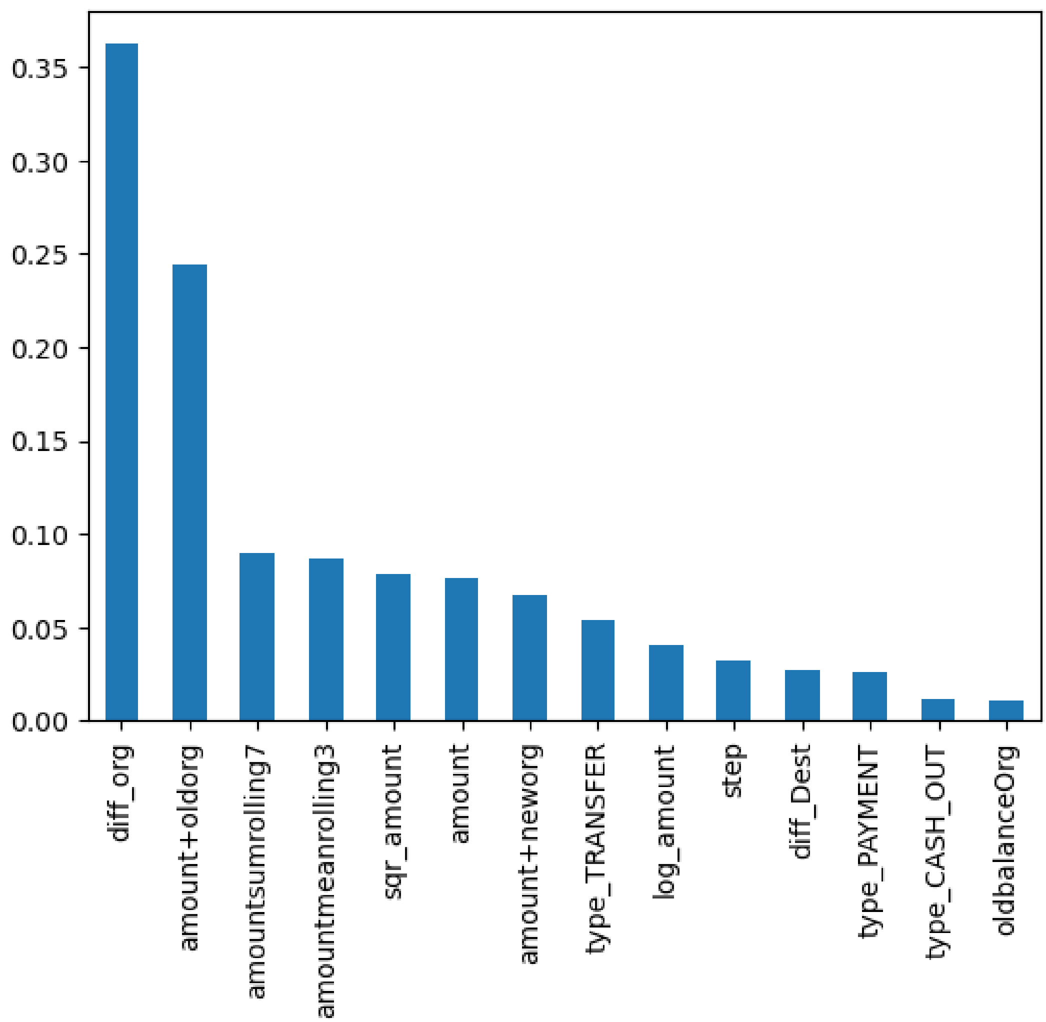
[31]:
```
isFraud                 1.000000
diff_org                0.362515
amount+oldorg           0.243875
amountsumrolling7       0.090157
amountmeanrolling3      0.087167
sqr_amount              0.078083
amount                  0.076700
amount+neworg           0.066871
type_TRANSFER           0.053862
log_amount              0.040672
step                    0.031596
diff_Dest               0.027033
type_PAYMENT            0.025694
```

```
type_CASH_OUT         0.011254
oldbalanceOrg         0.010158
newbalanceOrig        0.008147
oldbalanceDest        0.005883
type_DEBIT            0.002910
newbalanceDest        0.000538
Name: isFraud, dtype: float64
```

Feature Importance (Keeping Important features only)

```
[33]: correlation_threshold = 0.01  # 1%
newcorrwithtargetfeature = corrwithtargetfeature[(corrwithtargetfeature > 0.01)␣
 ↪& (corrwithtargetfeature != 1)]
newcorrwithtargetfeature.plot(kind='bar')
plt.suptitle('Fig-12')
plt.show()
```

Fig-12



[34]:
```
list_imp_feature=newcorrwithtargetfeature.index.tolist()  # list of important␣
↪feature
list_imp_feature.append('isFraud')
```

[35]:
```
dfnn=dfn[list_imp_feature]
```

[36]:
```
dfnn.corr()
```

[36]:
```
                   diff_org  amount+oldorg  amountsumrolling7  \
diff_org           1.000000      -0.354147          -0.082157
amount+oldorg     -0.354147       1.000000           0.089535
amountsumrolling7 -0.082157       0.089535           1.000000
```

```
amountmeanrolling3 -0.101654       0.120159           0.734990
sqr_amount          0.015851       0.143785           0.420259
amount             -0.102337       0.140615           0.465911
amount+neworg       0.073904       0.824067           0.032637
type_TRANSFER      -0.134808      -0.004318           0.324055
log_amount          0.115298       0.100867           0.255366
step               -0.007255       0.009109           0.043479
diff_Dest          -0.169292       0.003287           0.403032
type_PAYMENT       -0.134576      -0.072851          -0.163131
type_CASH_OUT      -0.250010      -0.063959          -0.010028
oldbalanceOrg       0.220297       0.361253          -0.010834
isFraud            -0.362515       0.243875           0.090157

                    amountmeanrolling3  sqr_amount    amount  amount+neworg  \
diff_org                     -0.101654    0.015851 -0.102337       0.073904
amount+oldorg                 0.120159    0.143785  0.140615       0.824067
amountsumrolling7             0.734990    0.420259  0.465911       0.032637
amountmeanrolling3            1.000000    0.548342  0.666454       0.045839
sqr_amount                    0.548342    1.000000  0.761408       0.117788
amount                        0.666454    0.761408  1.000000       0.062757
amount+neworg                 0.045839    0.117788  0.062757       1.000000
type_TRANSFER                 0.337824    0.515101  0.365896      -0.035199
log_amount                    0.299610    0.828564  0.387260       0.125567
step                          0.032245    0.013037  0.022373       0.001672
diff_Dest                     0.575280    0.589256  0.845964      -0.065673
type_PAYMENT                 -0.171910   -0.562094 -0.197444      -0.106187
type_CASH_OUT                -0.010360    0.173215 -0.004376      -0.108410
oldbalanceOrg                -0.006214    0.048295 -0.002763       0.521439
isFraud                       0.087167    0.078083  0.076700       0.066871

                    type_TRANSFER  log_amount      step  diff_Dest  \
diff_org                -0.134808    0.115298 -0.007255  -0.169292
amount+oldorg           -0.004318    0.100867  0.009109   0.003287
amountsumrolling7        0.324055    0.255366  0.043479   0.403032
amountmeanrolling3       0.337824    0.299610  0.032245   0.575280
sqr_amount               0.515101    0.828564  0.013037   0.589256
amount                   0.365896    0.387260  0.022373   0.845964
amount+neworg           -0.035199    0.125567  0.001672  -0.065673
type_TRANSFER            1.000000    0.354747  0.006926   0.320840
log_amount               0.354747    1.000000  0.007374   0.249841
step                     0.006926    0.007374  1.000000   0.001325
diff_Dest                0.320840    0.249841  0.001325   1.000000
type_PAYMENT            -0.216109   -0.735820  0.004927  -0.109286
type_CASH_OUT           -0.222672    0.340432 -0.012919   0.062758
oldbalanceOrg           -0.081593    0.106980 -0.010059  -0.087032
isFraud                  0.053862    0.040672  0.031596   0.027033
```

	type_PAYMENT	type_CASH_OUT	oldbalanceOrg	isFraud
diff_org	-0.134576	-0.250010	0.220297	-0.362515
amount+oldorg	-0.072851	-0.063959	0.361253	0.243875
amountsumrolling7	-0.163131	-0.010028	-0.010834	0.090157
amountmeanrolling3	-0.171910	-0.010360	-0.006214	0.087167
sqr_amount	-0.562094	0.173215	0.048295	0.078083
amount	-0.197444	-0.004376	-0.002763	0.076700
amount+neworg	-0.106187	-0.108410	0.521439	0.066871
type_TRANSFER	-0.216109	-0.222672	-0.081593	0.053862
log_amount	-0.735820	0.340432	0.106980	0.040672
step	0.004927	-0.012919	-0.010059	0.031596
diff_Dest	-0.109286	0.062758	-0.087032	0.027033
type_PAYMENT	1.000000	-0.526422	-0.189486	-0.025694
type_CASH_OUT	-0.526422	1.000000	-0.200899	0.011254
oldbalanceOrg	-0.189486	-0.200899	1.000000	0.010158
isFraud	-0.025694	0.011254	0.010158	1.000000

10 Removing features with correlations above 70% addresses multicollinearity, enhancing model stability and interpretability in machine learning.

[37]:
```
dfnn1=dfnn

# Set the correlation threshold
correlation_threshold = 0.7

# Calculate the correlation matrix
correlation_matrix = dfnn1.corr().abs()

# Create a mask for features to drop
mask = correlation_matrix >= correlation_threshold

# Get a set of feature names to drop
features_to_drop = set()

for i in range(len(dfnn1.columns)):
    for j in range(i + 1, len(dfnn1.columns)):
        if mask.iloc[i, j]:
            colname_i = dfnn1.columns[i]
            colname_j = dfnn1.columns[j]
            if colname_i not in features_to_drop:
                features_to_drop.add(colname_j)
```

[38]:
```
# Drop highly correlated features from your DataFrame ('df' in this case)
#dfnn1.drop(columns=features_to_drop, inplace=True)
features_to_drop
```

[38]: {'amount', 'amount+neworg', 'amountmeanrolling3', 'log_amount'}

[39]:
```
dfnn2=dfnn1.drop(columns=features_to_drop)
```

[40]:
```
dfnn2.head()
```

[40]:

	diff_org	amount+oldorg	amountsumrolling7	sqr_amount	type_TRANSFER
6	-7107.77	1.302108e+09	38659.54	84.307592	0
7	-7861.64	1.384334e+09	36681.54	88.665890	0
8	-2671.00	1.074907e+07	38841.62	63.437844	0
9	-5337.77	2.226918e+08	43998.39	73.060044	0
10	-4465.00	4.306466e+07	53462.33	98.208655	0

	step	diff_Dest	type_PAYMENT	type_CASH_OUT	oldbalanceOrg	isFraud
6	1	0.00	1	0	183195.00	0
7	1	0.00	1	0	176087.23	0
8	1	0.00	1	0	2671.00	0
9	1	-1549.21	0	0	41720.00	0
10	1	147137.12	0	0	4465.00	0

11 Normalization

[42]:
```
from sklearn.preprocessing import MinMaxScaler
slr=MinMaxScaler()
scl_feat=['diff_org','amount+oldorg','isFraud','amountsumrolling7','sqr_amount',
    'diff_Dest','oldbalanceOrg']
scl_featdf=slr.fit_transform(dfnn2[scl_feat])
scaled_df=dfnn2.copy()
scaled_df[scl_feat]=scl_featdf
```

[44]:
```
X=scaled_df.drop('isFraud',axis=1)
y=scaled_df['isFraud']
```

[45]:
```
X
```

[45]:

	diff_org	amount+oldorg	amountsumrolling7	sqr_amount
6	0.838663	2.185293e-06	0.000321	0.008768
7	0.838600	2.323292e-06	0.000302	0.009222
8	0.839035	1.803987e-08	0.000323	0.006598
9	0.838811	3.737377e-07	0.000370	0.007599
10	0.838885	7.227428e-08	0.000458	0.010214
…	…	…	…	…
6362615	0.810751	1.936458e-04	0.033254	0.060617
6362616	0.309568	6.685216e-02	0.089509	0.261288
6362617	0.309568	6.685216e-02	0.147577	0.261288
6362618	0.767922	1.212560e-03	0.154887	0.095889
6362619	0.767922	1.212560e-03	0.151088	0.095889

```
         type_TRANSFER  step  diff_Dest  type_PAYMENT  type_CASH_OUT  \
6                    0     1   0.109987             1              0
7                    0     1   0.109987             1              0
8                    0     1   0.109987             1              0
9                    0     1   0.109974             0              0
10                   0     1   0.111226             0              0
…                  …   …        …           …            …
6362615              0   743   0.112848             0              1
6362616              1   743   0.109987             0              0
6362617              0   743   0.163136             0              1
6362618              1   743   0.109987             0              0
6362619              0   743   0.117145             0              1

         oldbalanceOrg
6             0.003075
7             0.002955
8             0.000045
9             0.000700
10            0.000075
…                  …
6362615       0.005701
6362616       0.105923
6362617       0.105923
6362618       0.014265
6362619       0.014265

[6362614 rows x 10 columns]
```

[46]:

```
y
```

[46]:

```
6          0.0
7          0.0
8          0.0
9          0.0
10         0.0
          …
6362615    1.0
6362616    1.0
6362617    1.0
6362618    1.0
6362619    1.0
Name: isFraud, Length: 6362614, dtype: float64
```

[47]:

```
from imblearn.over_sampling import SMOTE
smote=SMOTE()
X_resamp,y_resamp=smote.fit_resample(X,y)
```

[48]:
```
from sklearn.model_selection import train_test_split
X_trn,X_tst,y_trn,y_tst=train_test_split(X_resamp,y_resamp,test_size=0.2)
```

[49]:
```
y_resamp.value_counts()
```

[49]:
```
0.0    6354403
1.0    6354403
Name: isFraud, dtype: int64
```

[81]:
```
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
import xgboost as xgb
```

[41]:
```
from sklearn.metrics import accuracy_score, precision_score, recall_score,␣
↪f1_score, confusion_matrix, classification_report
```

11.1 Logistic Regression

[83]:
```
logr=LogisticRegression()
logr.fit(X_trn,y_trn)
```

```
C:\Users\Kundan Mourya\anaconda3\lib\site-
packages\sklearn\linear_model\_logistic.py:460: ConvergenceWarning: lbfgs failed
to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression
  n_iter_i = _check_optimize_result(
```

[83]:
```
LogisticRegression()
```

11.1.1 Make Predictions

[84]:
```
logrypred=logr.predict(X_tst)
```

11.1.2 Evaluate the Model

[85]:
```
print("Logistic Regression:")
print("Accuracy:", accuracy_score(y_tst,logrypred))
print("Precision:", precision_score(y_tst,logrypred))
print("Recall:", recall_score(y_tst,logrypred))
print("F1 Score:", f1_score(y_tst,logrypred))
```

```
print("Confusion Matrix:")
print(confusion_matrix(y_tst,logrypred))

print("Classification Report:")
print(classification_report(y_tst,logrypred))
print()
```

```
Logistic Regression:
Accuracy: 0.9555048820463914
Precision: 0.9465172019162253
Recall: 0.9655859297557722
F1 Score: 0.9559564829988348
Confusion Matrix:
[[1201302   69352]
 [  43744 1227364]]
Classification Report:
              precision    recall  f1-score   support

         0.0       0.96      0.95      0.96   1270654
         1.0       0.95      0.97      0.96   1271108

    accuracy                           0.96   2541762
   macro avg       0.96      0.96      0.96   2541762
weighted avg       0.96      0.96      0.96   2541762
```

[86]:
```
logr_accuracy= accuracy_score(y_tst,logrypred)
```

11.1.3 Build and Train the Decision Tree Model

[55]:
```
dt=DecisionTreeClassifier(max_depth=5)  # to avoid overfitting max_depth=5
dt.fit(X_trn,y_trn)
```

[55]:
```
DecisionTreeClassifier(max_depth=5)
```

11.1.4 Make Predictions

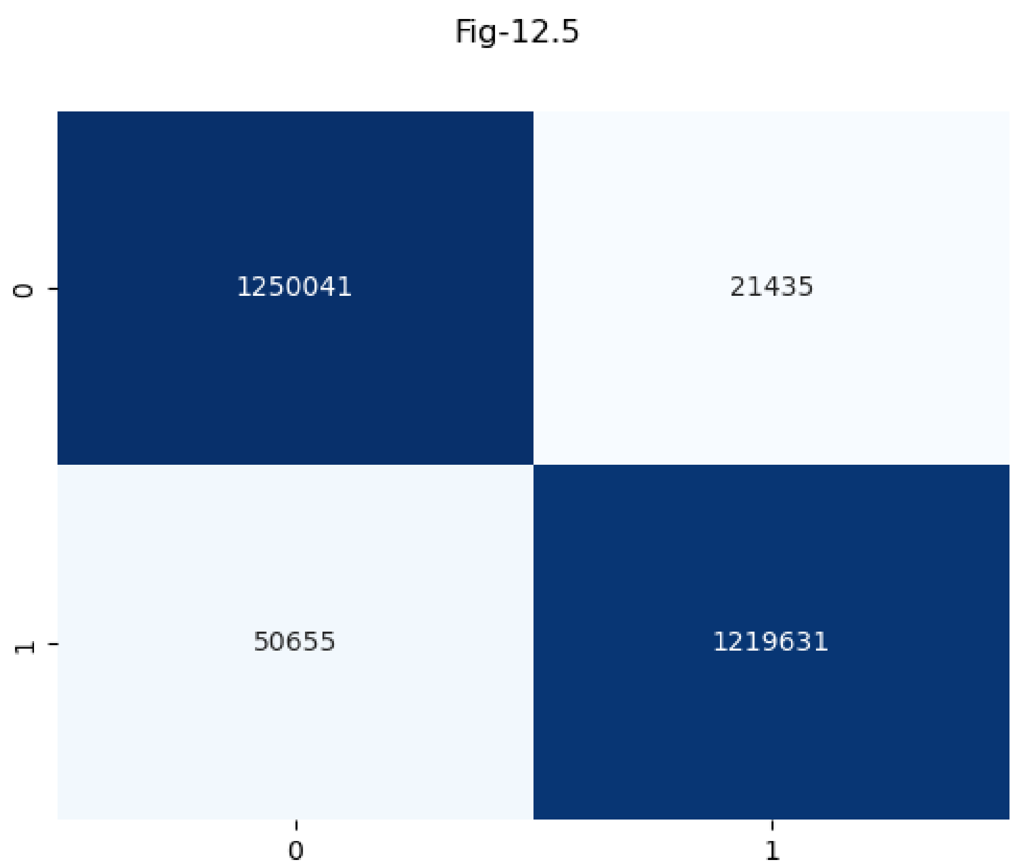
[56]:
```
dtypred=dt.predict(X_tst)
```

11.1.5 Evaluate the Model

[57]:
```
print("Accuracy:", round(accuracy_score(y_tst, dtypred)*100,2),"%")
print("Precision:", round(precision_score(y_tst, dtypred)*100,2),"%")
print("Recall:", round(recall_score(y_tst, dtypred)*100,2),"%")
print("F1 Score:", round(f1_score(y_tst, dtypred)*100,2),"%")
print("Confusion Matrix:")
```

```python
sns.heatmap(confusion_matrix(y_tst, dtypred),annot=True, fmt='d', cmap='Blues',
    cbar=False)
plt.suptitle('Fig-12.5')
plt.show()
print("Classification Report:")
print(classification_report(y_tst, dtypred))
print()
```

```
Accuracy: 97.16 %
Precision: 98.27 %
Recall: 96.01 %
F1 Score: 97.13 %
Confusion Matrix:
```




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

         0.0       0.96      0.98      0.97   1271476
         1.0       0.98      0.96      0.97   1270286
```

```
    accuracy                           0.97   2541762
   macro avg       0.97      0.97      0.97   2541762
weighted avg       0.97      0.97      0.97   2541762
```

[59]:
```python
# Plot the decision tree
feature_names_list = X_resamp.columns.tolist()

# Plot the decision tree
plt.figure(figsize=(15, 10))  # Adjust the figure size as needed
plot_tree(dt, feature_names=feature_names_list, class_names=['Not Fraud',␣
 ↪'Fraud'], filled=True)
plt.title('Decision tree using Entropy', fontsize=16)
plt.suptitle('Fig-13')
plt.show()
```

Fig-13

Decision tree using Entropy