fradulent transaction detection

September 7, 2024

[85]:
```
import pandas as pd
import numpy as np

import warnings
warnings.filterwarnings("ignore", category=DeprecationWarning)

import seaborn as sns
import matplotlib.pyplot as plt
```

[86]:
```
data = pd.read_csv('PS_20174392719_1491204439457_log.csv')
```

[90]:
```
print('Data does not have any NULL value.')
data.isnull().any()
```

```
Data does not have any NULL value.
```

[90]:
```
step              False
type              False
amount            False
oldbalanceOrg     False
newbalanceOrg     False
oldbalanceDest    False
newbalanceDest    False
isFraud           False
isFlaggedFraud    False
dtype: bool
```

[91]:
```
data.head()
```

[91]:

	step	type	amount	oldbalanceOrg	newbalanceOrg	oldbalanceDest	newbalanceDest	isFraud	isFlaggedFraud
0	1	PAYMENT	9839.64	170136.0	160296.36	0.0	0.0	0	0
1	1	PAYMENT	1864.28	21249.0	19384.72	0.0			
2	1	TRANSFER	181.00	181.0	0.00	0.0			
3	1	CASH_OUT	181.00	181.0	0.00	21182.0			
4	1	PAYMENT	11668.14	41554.0	29885.86	0.0			

```
1              0.0         0               0
2              0.0         1               0
3              0.0         1               0
4              0.0         0               0
```

[95]:
```
data.rename(columns={'newbalanceOrig':'newbalanceOrg'},inplace=True)
data.drop(labels=['nameOrig','nameDest'],axis=1,inplace=True)
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
Cell In[95], line 2
      1 data.rename(columns={'newbalanceOrig':'newbalanceOrg'},inplace=True)
----> 2 data.drop(labels=['nameOrig','nameDest'],axis=1,inplace=True)

File /Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/
 ↪site-packages/pandas/core/frame.py:5581, in DataFrame.drop(self, labels, axis␣
 ↪index, columns, level, inplace, errors)
   5433 def drop(
   5434     self,
   5435     labels: IndexLabel | None = None,
   (…)
   5442     errors: IgnoreRaise = "raise",
   5443 ) -> DataFrame | None:
   5444     """
   5445     Drop specified labels from rows or columns.
   5446
   (…)
   5579             weight  1.0     0.8
   5580     """
-> 5581     return super().drop(
   5582         labels=labels,
   5583         axis=axis,
   5584         index=index,
   5585         columns=columns,
   5586         level=level,
   5587         inplace=inplace,
   5588         errors=errors,
   5589     )

File /Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/
 ↪site-packages/pandas/core/generic.py:4788, in NDFrame.drop(self, labels, axis␣
 ↪index, columns, level, inplace, errors)
   4786 for axis, labels in axes.items():
   4787     if labels is not None:
-> 4788         obj = obj._drop_axis(labels, axis, level=level, errors=errors)
   4790 if inplace:
   4791     self._update_inplace(obj)
```

```
File /Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/
 ↪site-packages/pandas/core/generic.py:4830, in NDFrame._drop_axis(self, labels␣
 ↪axis, level, errors, only_slice)
   4828         new_axis = axis.drop(labels, level=level, errors=errors)
   4829     else:
-> 4830         new_axis = axis.drop(labels, errors=errors)
   4831     indexer = axis.get_indexer(new_axis)
   4833 # Case for non-unique axis
   4834 else:

File /Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/
 ↪site-packages/pandas/core/indexes/base.py:7070, in Index.drop(self, labels,␣
 ↪errors)
   7068 if mask.any():
   7069     if errors != "ignore":
-> 7070         raise KeyError(f"{labels[mask].tolist()} not found in axis")
   7071     indexer = indexer[~mask]
   7072 return self.delete(indexer)

KeyError: "['nameOrig', 'nameDest'] not found in axis"
```

[96]:
```
print('Minimum value of Amount, Old/New Balance of Origin/Destination:')
data[[ 'amount','oldbalanceOrg', 'newbalanceOrg', 'oldbalanceDest',␣
 ↪'newbalanceDest']].min()
```

```
Minimum value of Amount, Old/New Balance of Origin/Destination:
```

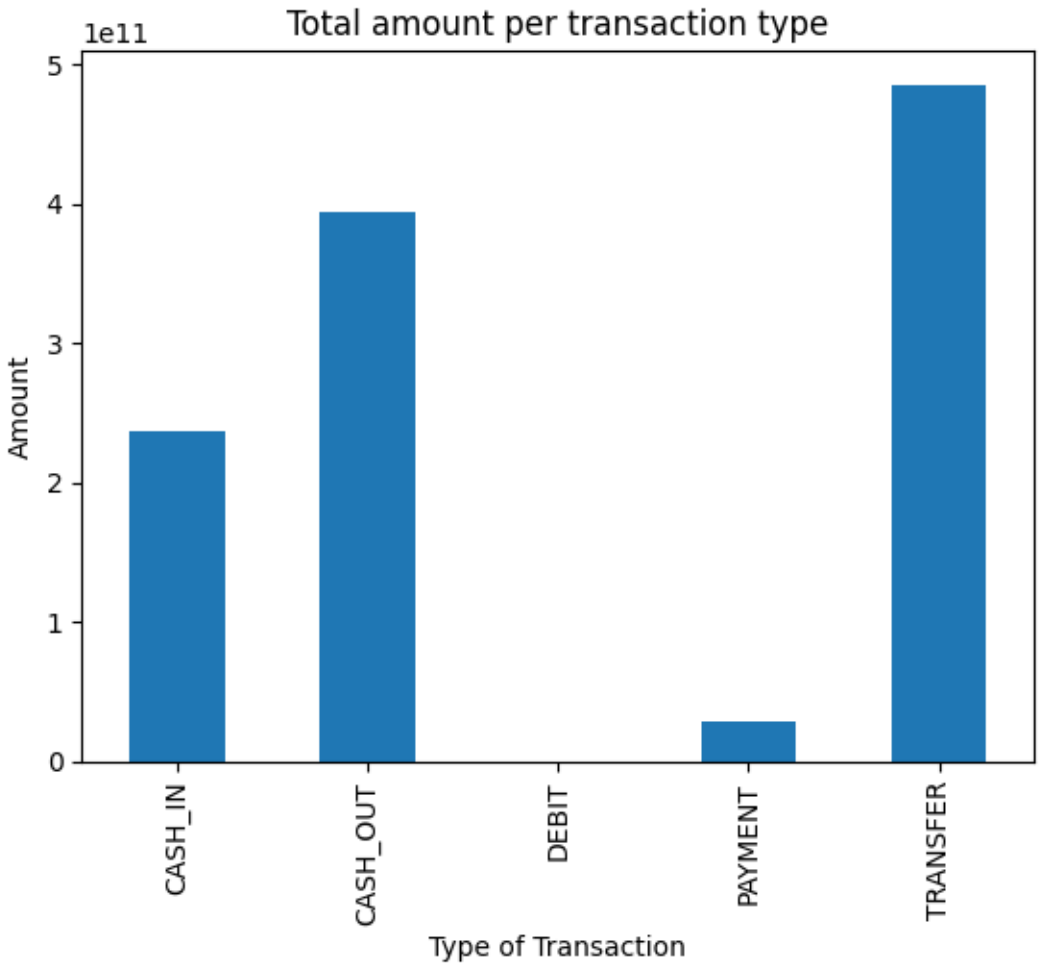
[96]:
```
amount            0.0
oldbalanceOrg     0.0
newbalanceOrg     0.0
oldbalanceDest    0.0
newbalanceDest    0.0
dtype: float64
```

[97]:
```
print('Maximum value of Amount, Old/New Balance of Origin/Destination:')
data[[ 'amount','oldbalanceOrg', 'newbalanceOrg', 'oldbalanceDest',␣
 ↪'newbalanceDest']].max()
```

```
Maximum value of Amount, Old/New Balance of Origin/Destination:
```

[97]:
```
amount            9.244552e+07
oldbalanceOrg     5.958504e+07
newbalanceOrg     4.958504e+07
oldbalanceDest    3.560159e+08
newbalanceDest    3.561793e+08
dtype: float64
```

[98]:
```
var = data.groupby('type').amount.sum()
fig = plt.figure()
ax1 = fig.add_subplot(1,1,1)
var.plot(kind='bar')
ax1.set_title("Total amount per transaction type")
ax1.set_xlabel('Type of Transaction')
ax1.set_ylabel('Amount');
```



[99]:
```
data.loc[data.isFraud == 1].type.unique()
```

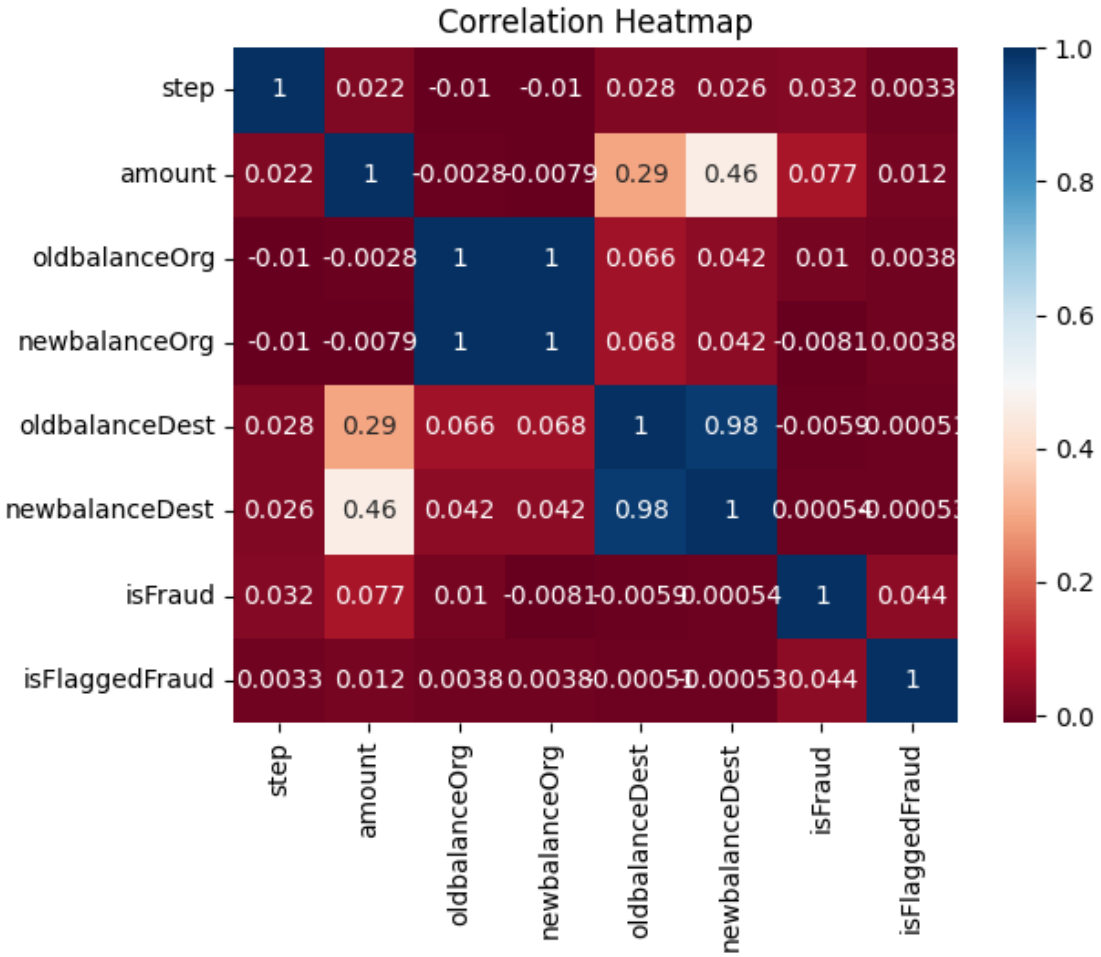
[99]:
```
array(['TRANSFER', 'CASH_OUT'], dtype=object)
```

[100]:
```
data_numeric = data.apply(pd.to_numeric, errors='coerce')
```

[101]:
```
data_numeric = data.select_dtypes(include=['number'])
```

[102]:
```
data_numeric = data_numeric.dropna()
```

[103]:
```
sns.heatmap(data_numeric.corr(), cmap='RdBu', annot=True)
plt.title('Correlation Heatmap')
plt.show()
```



[104]:
```
fraud = data.loc[data.isFraud == 1]
nonfraud = data.loc[data.isFraud == 0]
```

[105]:
```
fraudcount = fraud.isFraud.count()
nonfraudcount = nonfraud.isFraud.count()
```

[ ]:
```

```

[106]:
```
print('The total number of fraud transaction is {}.'.format(data.isFraud.sum()))
print('The total number of fraud transaction which is marked as fraud {}.'.
  ↪format(data.isFlaggedFraud.sum()))
print('Ratio of fraud transaction vs non-fraud transaction is 1:{}.'.
  ↪format(int(nonfraudcount//fraudcount)))
```

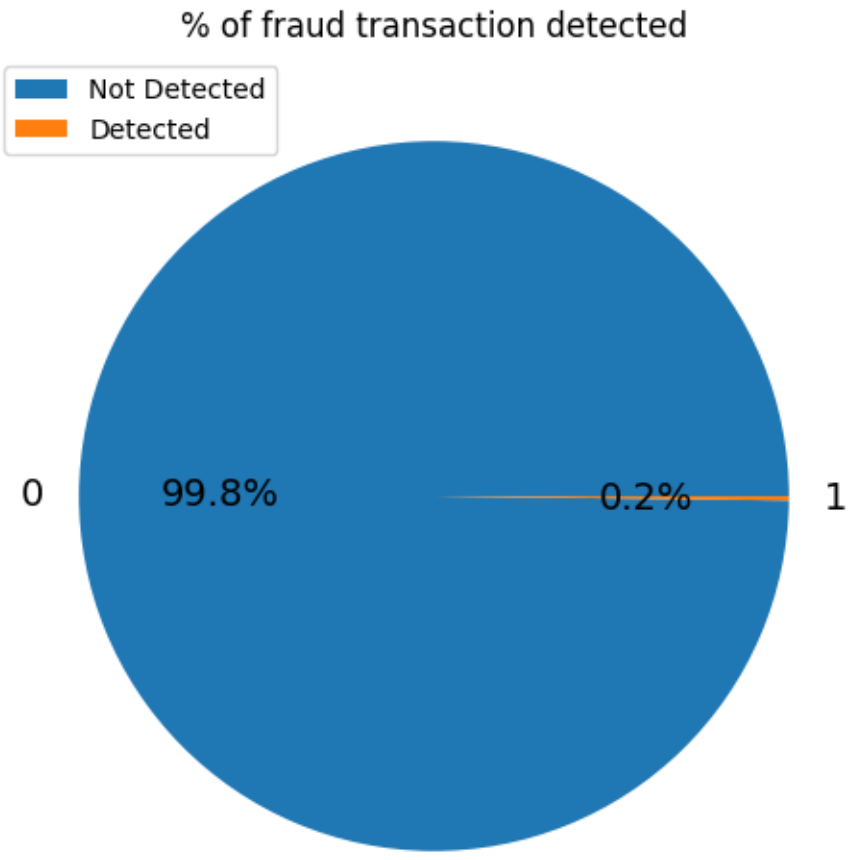
```
The total number of fraud transaction is 8213.
The total number of fraud transaction which is marked as fraud 16.
Ratio of fraud transaction vs non-fraud transaction is 1:773.
```

[107]:
```
print('Thus in every 773 transaction there is 1 fraud transaction happening.')
print('Amount lost due to these fraud transaction is ${}.'.format(int(fraud.
  ↪amount.sum())))
```

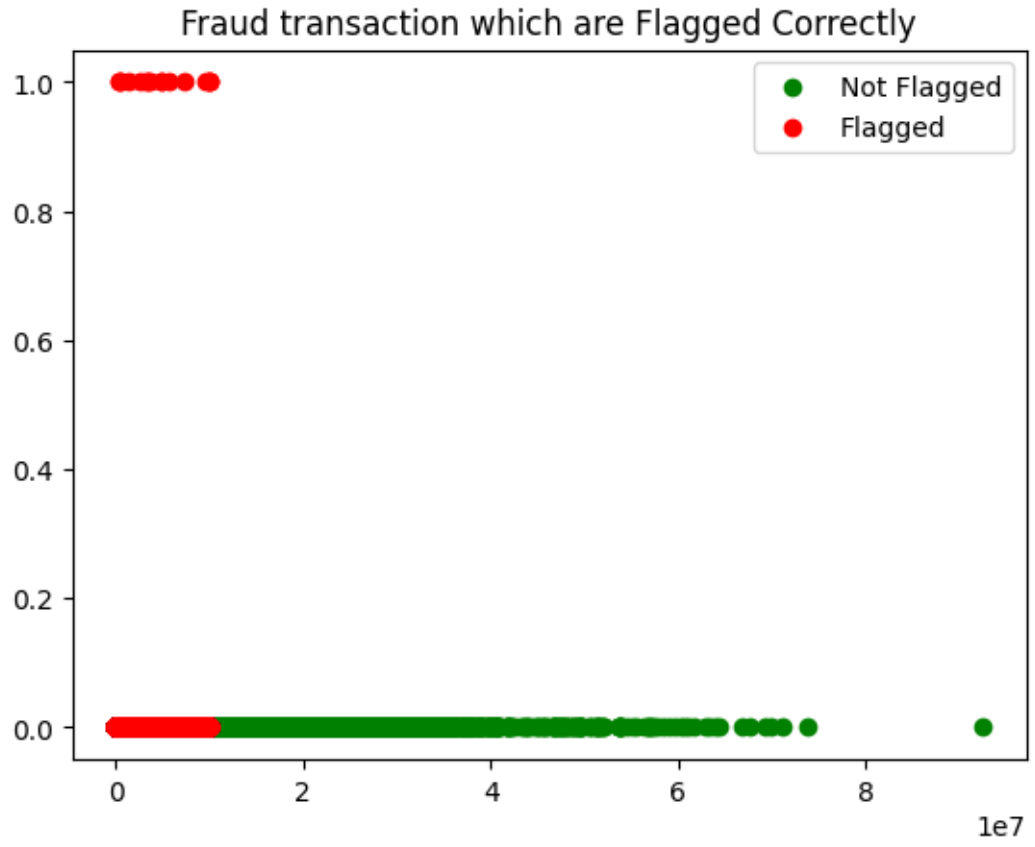
```
Thus in every 773 transaction there is 1 fraud transaction happening.
Amount lost due to these fraud transaction is $12056415427.
```

[108]:
```
piedata = fraud.groupby(['isFlaggedFraud']).sum()
```

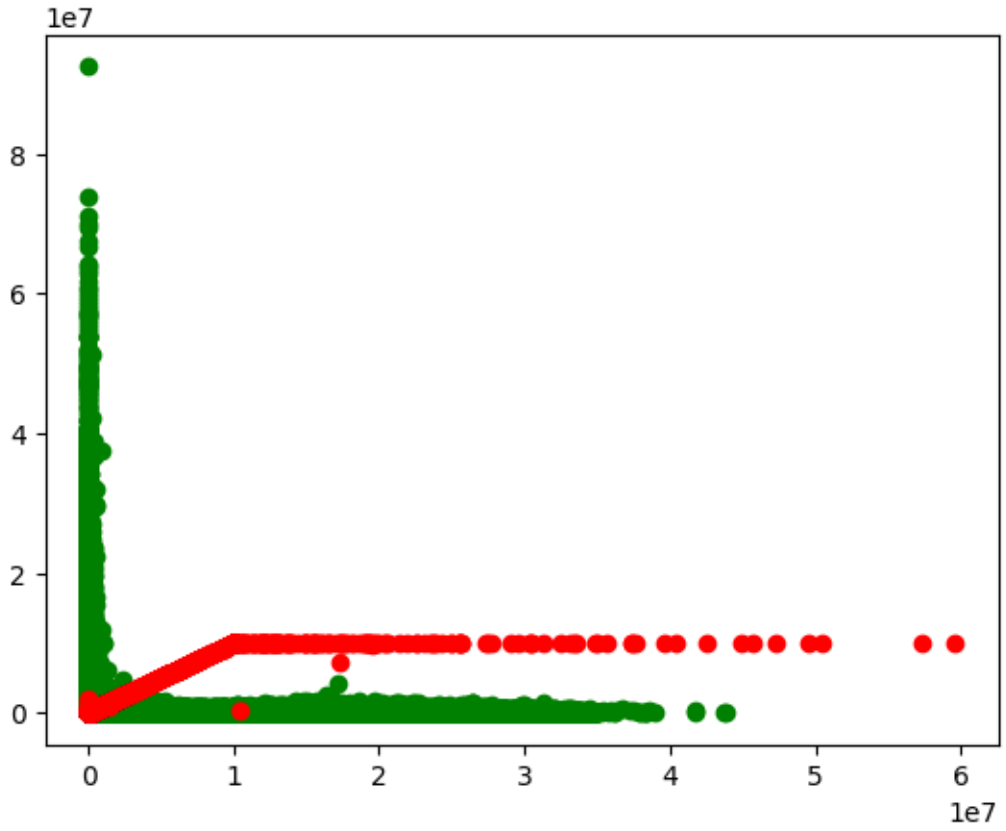
[109]:
```
f, axes = plt.subplots(1,1, figsize=(6,6))
axes.set_title("% of fraud transaction detected")
piedata.plot(kind='pie',y='isFraud',ax=axes,␣
  ↪fontsize=14,shadow=False,autopct='%1.1f%%');
axes.set_ylabel('');
plt.legend(loc='upper left',labels=['Not Detected','Detected'])
plt.show()
```



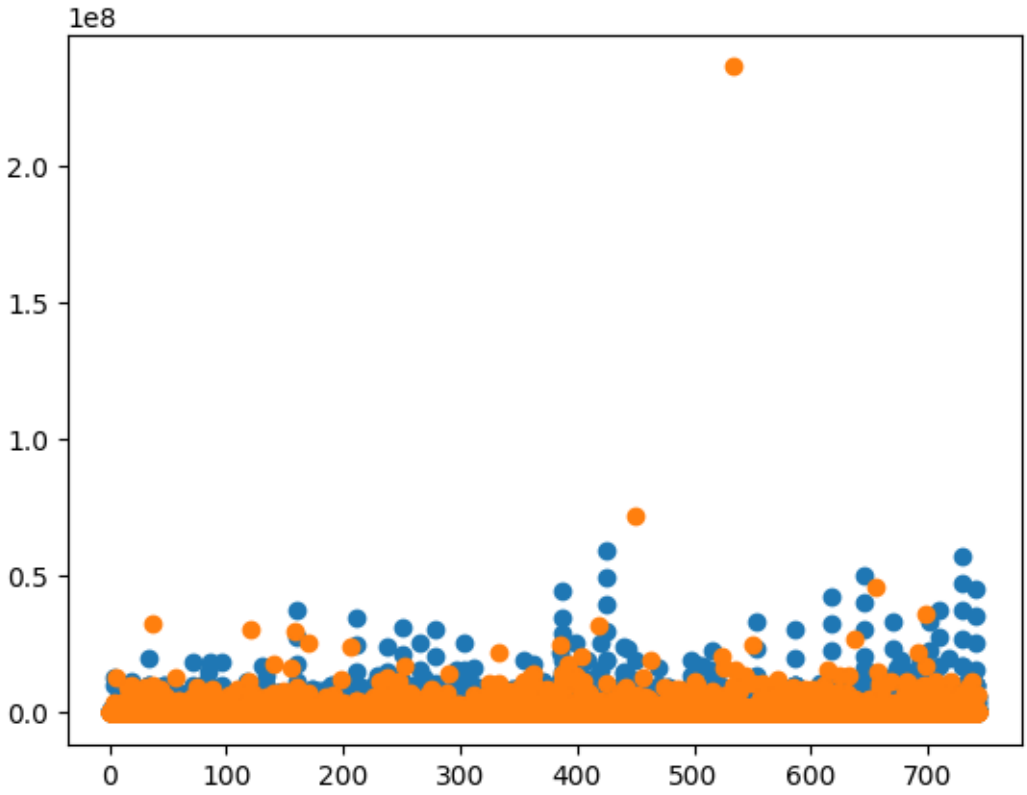
[110]:
```
fig = plt.figure()
axes = fig.add_subplot(1,1,1)
axes.set_title("Fraud transaction which are Flagged Correctly")
axes.scatter(nonfraud['amount'],nonfraud['isFlaggedFraud'],c='g')
axes.scatter(fraud['amount'],fraud['isFlaggedFraud'],c='r')
plt.legend(loc='upper right',labels=['Not Flagged','Flagged'])
plt.show()
```



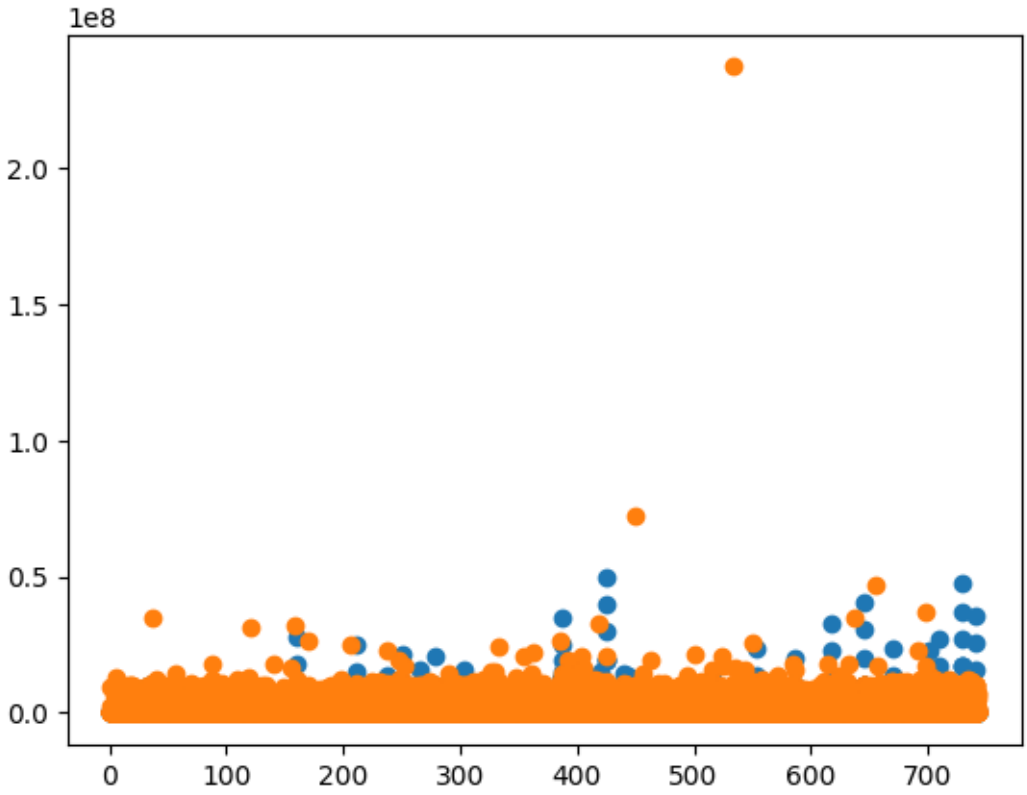
[111]:
```
fig = plt.figure()
ax = fig.add_subplot(1,1,1)
ax.scatter(nonfraud['oldbalanceOrg'],nonfraud['amount'],c='g')
ax.scatter(fraud['oldbalanceOrg'],fraud['amount'],c='r')
plt.show()
```



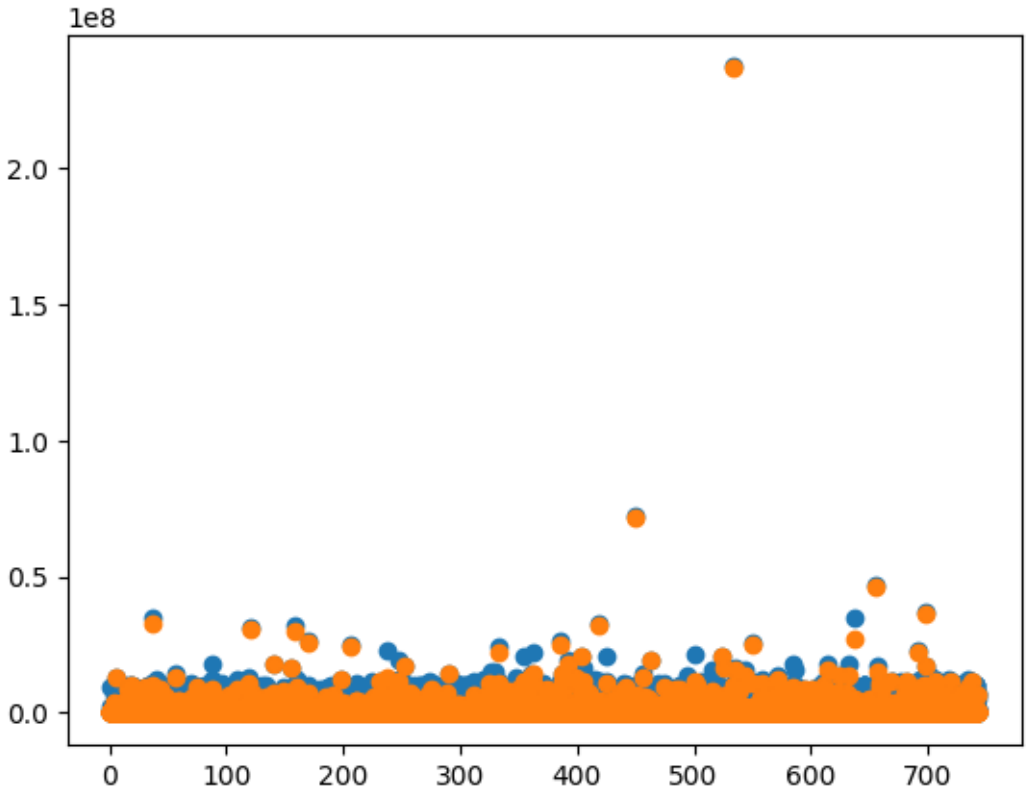
[112]:
```
fig = plt.figure()
ax = fig.add_subplot(1,1,1)
ax.scatter(fraud['step'],fraud['oldbalanceOrg'])
ax.scatter(fraud['step'],fraud['oldbalanceDest'])
plt.show()
```



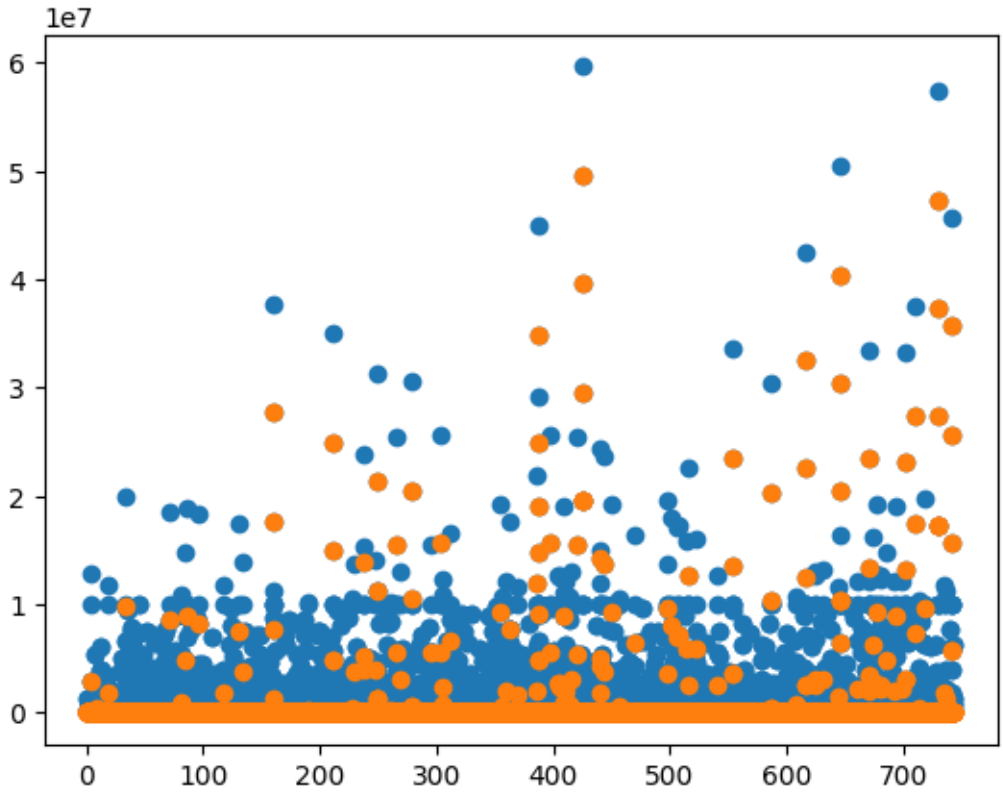
[113]:
```
fig = plt.figure()
ax = fig.add_subplot(1,1,1)
ax.scatter(fraud['step'],fraud['newbalanceOrg'])
ax.scatter(fraud['step'],fraud['newbalanceDest'])
plt.show()
```



[114]:
```
fig = plt.figure()
ax = fig.add_subplot(1,1,1)
ax.scatter(fraud['step'],fraud['newbalanceDest'])
ax.scatter(fraud['step'],fraud['oldbalanceDest'])
plt.show()
```



[115]:
```
fig = plt.figure()
ax = fig.add_subplot(1,1,1)
ax.scatter(fraud['step'],fraud['oldbalanceOrg'])
ax.scatter(fraud['step'],fraud['newbalanceOrg'])
plt.show()
```



[116]:
```
import pickle
```

[117]:
```
import warnings
warnings.filterwarnings("ignore", category=DeprecationWarning)
```

[ ]:
```

```

[118]:
```
data_fraud = pd.read_csv('PS_20174392719_1491204439457_log.csv')
```

[ ]:
```

```

[ ]:
```

```

[119]:
```
data_fraud = data_fraud.replace(to_replace={'PAYMENT':1,'TRANSFER':2,'CASH_OUT':
↪3,
                                            'CASH_IN':4,'DEBIT':5,'No':0,'Yes':
↪1})
```

```
/var/folders/x_/_w99bkln3b125c6p2kgxkrwh0000gn/T/ipykernel_43889/2197214739.py:1
: FutureWarning: Downcasting behavior in `replace` is deprecated and will be
removed in a future version. To retain the old behavior, explicitly call
`result.infer_objects(copy=False)`. To opt-in to the future behavior, set
```

```
`pd.set_option('future.no_silent_downcasting', True)`
  data_fraud =
data_fraud.replace(to_replace={'PAYMENT':1,'TRANSFER':2,'CASH_OUT':3,
```

[120]:
```
data_fraud.drop(['nameOrig','nameDest','isFlaggedFraud'],axis=1,inplace=True)
```

[121]:
```
data_fraud.head()
```

[121]:

	step	type	amount	oldbalanceOrg	newbalanceOrig	oldbalanceDest	newbalanceDest	isFraud
0	1	1	9839.64	170136.0	160296.36	0.0	0.0	0
1	1	1	1864.28	21249.0	19384.72	0.0	0.0	0
2	1	2	181.00	181.0	0.00	0.0	0.0	1
3	1	3	181.00	181.0	0.00	21182.0	0.0	1
4	1	1	11668.14	41554.0	29885.86	0.0	0.0	0

[122]:
```
X = data_fraud.drop(['isFraud'],axis=1)
y = data_fraud[['isFraud']]
```

[123]:
```
from sklearn.model_selection import train_test_split
train_X, test_X, train_y, test_y = train_test_split(X, y, test_size = 0.2,␣
 ↪random_state = 121)
```

[124]:
```
from sklearn.ensemble import RandomForestClassifier
clf = RandomForestClassifier(n_estimators=15)
```

[125]:
```
if True:
    probabilities = clf.fit(train_X, train_y.values.ravel()).predict(test_X)
```

[126]:
```
from sklearn.metrics import average_precision_score
if True:
    print(average_precision_score(test_y,probabilities))
```

```
0.774388676557409
```

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]: