Import Libraries

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

Read the dataset

In [2]:
```
data = pd.read_csv('Downloads/Fraud.csv')
data.head()
```

Out[2]:

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

Data types

In [3]:
```
data.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
```

Checking for null values

In [4]:
```
data.isnull().sum()
```

Out[4]:
```
step              0
type              0
amount            0
nameOrig          0
```

```
oldbalanceOrg     0
newbalanceOrig    0
nameDest          0
oldbalanceDest    0
newbalanceDest    0
isFraud           0
isFlaggedFraud    0
dtype: int64
```

Describe some statistic

In [5]:
```
data.describe()
```

Out[5]:

	step	amount	oldbalanceOrg	newbalanceOrig	oldbalanceDest	newbalanceDest	isFrau
count	6.362620e+06	6.362620e+06	6.362620e+06	6.362620e+06	6.362620e+06	6.362620e+06	6.362620e+(
mean	2.433972e+02	1.798619e+05	8.338831e+05	8.551137e+05	1.100702e+06	1.224996e+06	1.290820e-(
std	1.423320e+02	6.038582e+05	2.888243e+06	2.924049e+06	3.399180e+06	3.674129e+06	3.590480e-(
min	1.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+(
25%	1.560000e+02	1.338957e+04	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+(
50%	2.390000e+02	7.487194e+04	1.420800e+04	0.000000e+00	1.327057e+05	2.146614e+05	0.000000e+(
75%	3.350000e+02	2.087215e+05	1.073152e+05	1.442584e+05	9.430367e+05	1.111909e+06	0.000000e+(
max	7.430000e+02	9.244552e+07	5.958504e+07	4.958504e+07	3.560159e+08	3.561793e+08	1.000000e+(

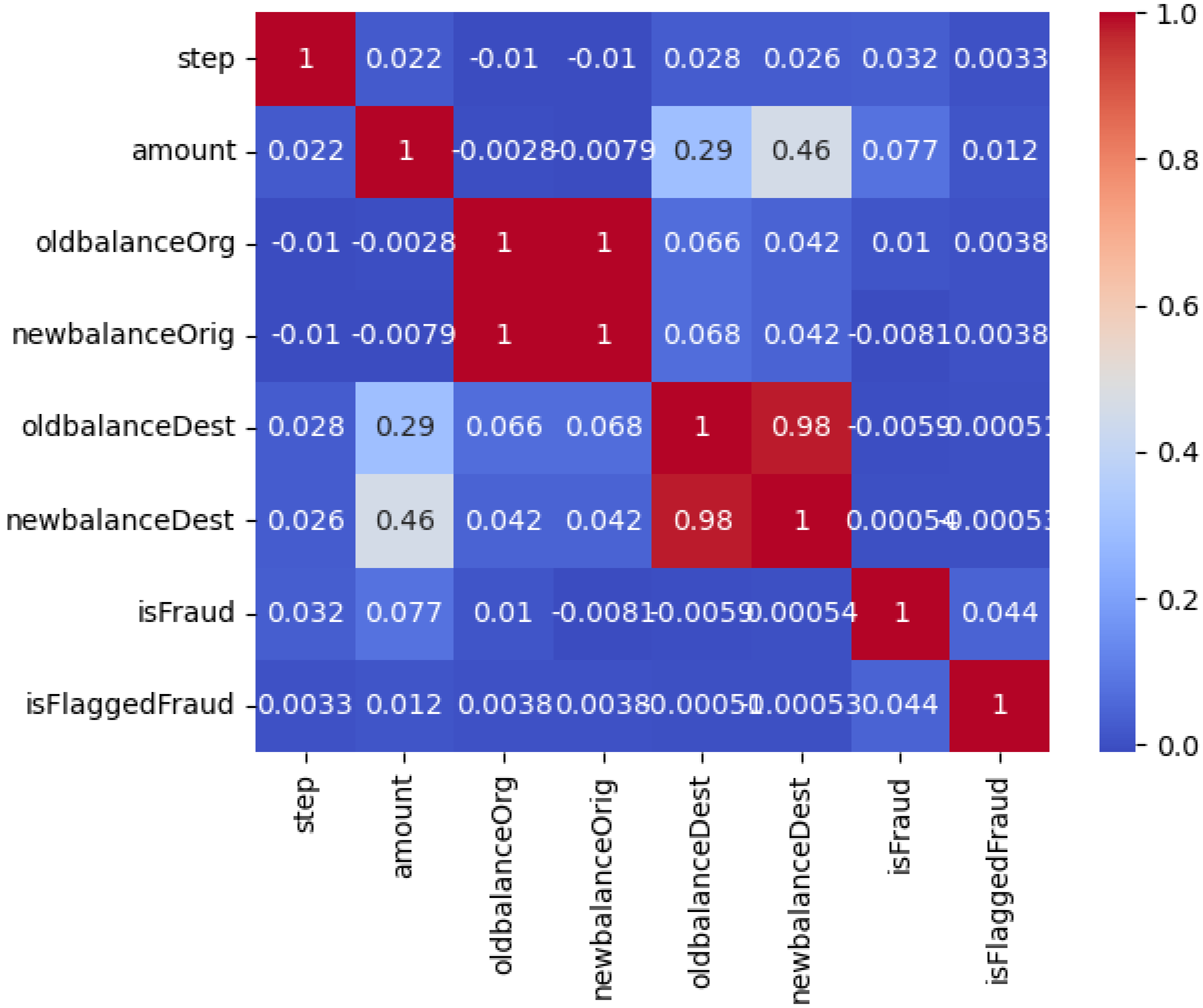
Correlation of the variables

In [ ]:
```
data.corr()
```

In [7]:
```
sns.heatmap(data.corr(),cmap='coolwarm',annot =True)
```

Out[7]:

```
<AxesSubplot:>
```



Frequency of target value

In [8]:

```
data.isFraud.value_counts()
```

Out[8]:

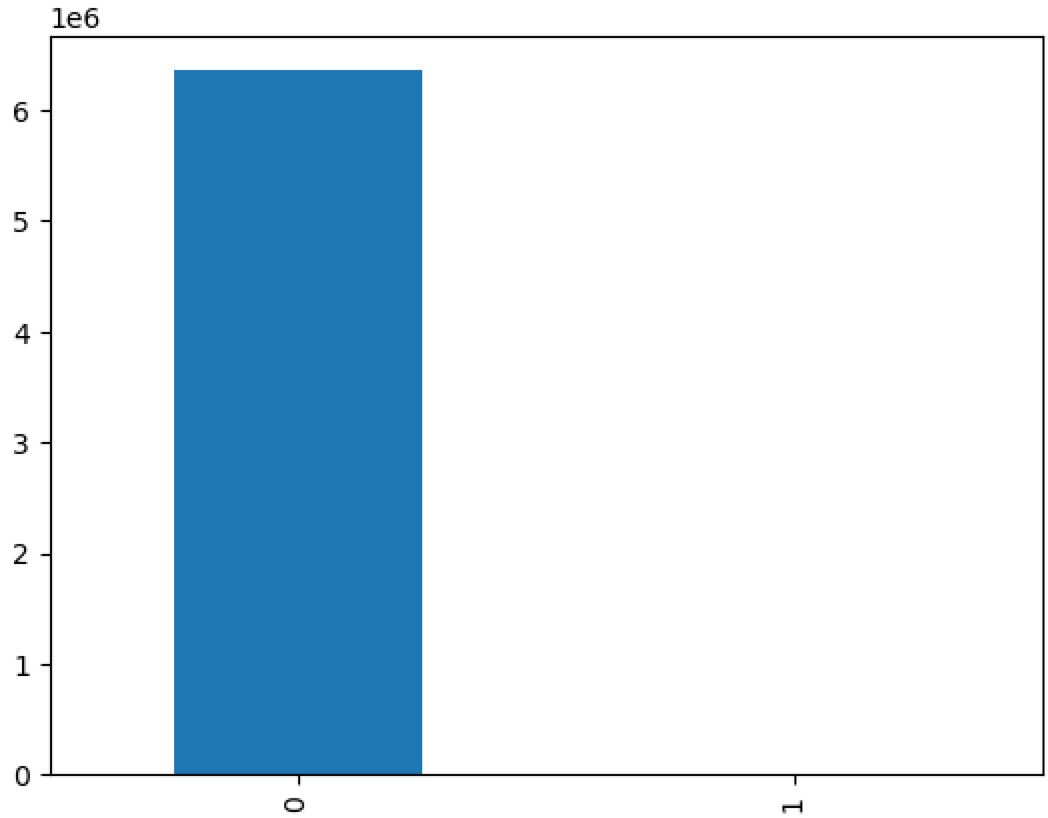
```
0    6354407
1       8213
Name: isFraud, dtype: int64
```

In [9]:

```
data.isFraud.value_counts().plot(kind ='bar')
```

Out[9]:

```
<AxesSubplot:>
```



In [10]:
```
data[data.isFlaggedFraud ==1]
```

Out[10]:

	step	type	amount	nameOrig	oldbalanceOrg	newbalanceOrig	nameDest	oldbalanceDe
2736446	212	TRANSFER	4953893.08	C728984460	4953893.08	4953893.08	C639921569	(
3247297	250	TRANSFER	1343002.08	C1100582606	1343002.08	1343002.08	C1147517658	(
3760288	279	TRANSFER	536624.41	C1035541766	536624.41	536624.41	C1100697970	(
5563713	387	TRANSFER	4892193.09	C908544136	4892193.09	4892193.09	C891140444	(
5996407	425	TRANSFER	10000000.00	C689608084	19585040.37	19585040.37	C1392803603	(
5996409	425	TRANSFER	9585040.37	C452586515	19585040.37	19585040.37	C1109166882	(
6168499	554	TRANSFER	3576297.10	C193696150	3576297.10	3576297.10	C484597480	(
6205439	586	TRANSFER	353874.22	C1684585475	353874.22	353874.22	C1770418982	(
6266413	617	TRANSFER	2542664.27	C786455622	2542664.27	2542664.27	C661958277	(
6281482	646	TRANSFER	10000000.00	C19004745	10399045.08	10399045.08	C1806199534	(
6281484	646	TRANSFER	399045.08	C724693370	10399045.08	10399045.08	C1909486199	(
6296014	671	TRANSFER	3441041.46	C917414431	3441041.46	3441041.46	C1082139865	(
6351225	702	TRANSFER	3171085.59	C1892216157	3171085.59	3171085.59	C1308068787	(
6362460	730	TRANSFER	10000000.00	C2140038573	17316255.05	17316255.05	C1395467927	(
6362462	730	TRANSFER	7316255.05	C1869569059	17316255.05	17316255.05	C1861208726	(
6362584	741	TRANSFER	5674547.89	C992223106	5674547.89	5674547.89	C1366804249	(

In [11]:
```
# feature engineering add new feature to the dataframe which will show the transaction i
data['above200000'] = data['amount'] >= 200000.00
data['above200000'] = data.above200000.map({False:0,True:1})
```

In [12]:
```
data
```

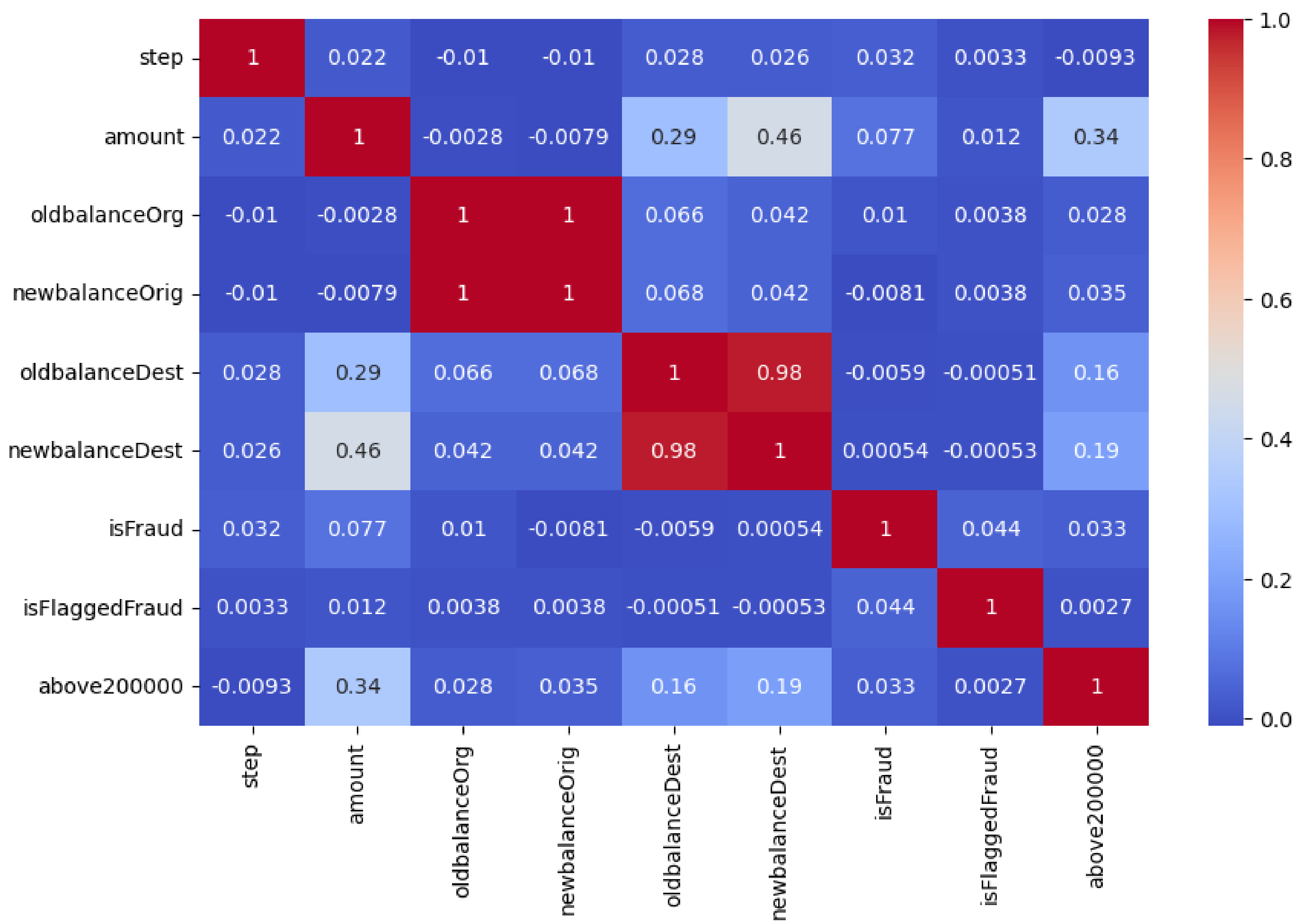
Out[12]:

	step	type	amount	nameOrig	oldbalanceOrg	newbalanceOrig	nameDest	oldbalanceDe
0	1	PAYMENT	9839.64	C1231006815	170136.00	160296.36	M1979787155	0.
1	1	PAYMENT	1864.28	C1666544295	21249.00	19384.72	M2044282225	0.
2	1	TRANSFER	181.00	C1305486145	181.00	0.00	C553264065	0.
3	1	CASH_OUT	181.00	C840083671	181.00	0.00	C38997010	21182.
4	1	PAYMENT	11668.14	C2048537720	41554.00	29885.86	M1230701703	0.
...	...	...	...	...	...	...	...	
6362615	743	CASH_OUT	339682.13	C786484425	339682.13	0.00	C776919290	0.
6362616	743	TRANSFER	6311409.28	C1529008245	6311409.28	0.00	C1881841831	0.
6362617	743	CASH_OUT	6311409.28	C1162922333	6311409.28	0.00	C1365125890	68488.
6362618	743	TRANSFER	850002.52	C1685995037	850002.52	0.00	C2080388513	0.
6362619	743	CASH_OUT	850002.52	C1280323807	850002.52	0.00	C873221189	6510099.

6362620 rows × 12 columns

In [13]:
```
plt.figure(figsize=(10,6))
sns.heatmap(data.corr(),cmap='coolwarm',annot =True,)
```

Out[13]: <AxesSubplot:>



In [14]:
```
data.columns
```

```
Out[14]: Index(['step', 'type', 'amount', 'nameOrig', 'oldbalanceOrg', 'newbalanceOrig',
               'nameDest', 'oldbalanceDest', 'newbalanceDest', 'isFraud',
               'isFlaggedFraud', 'above200000'],
              dtype='object')
```

```
In [15]: # it gives low score so we can drop this
         from sklearn.metrics.cluster import mutual_info_score
         mutual_info_score(data.type,data.isFraud)
```

```
Out[15]: 0.0013803993713611398
```

```
In [16]: #create new dataframe for training and drop unnecessary features
         x_train = data.drop(columns=['step','type','nameOrig','nameDest','isFlaggedFraud','isFra
```

```
In [17]: data.amount.max()
```

```
Out[17]: 92445516.64
```

```
In [18]: data.amount.mean()
```

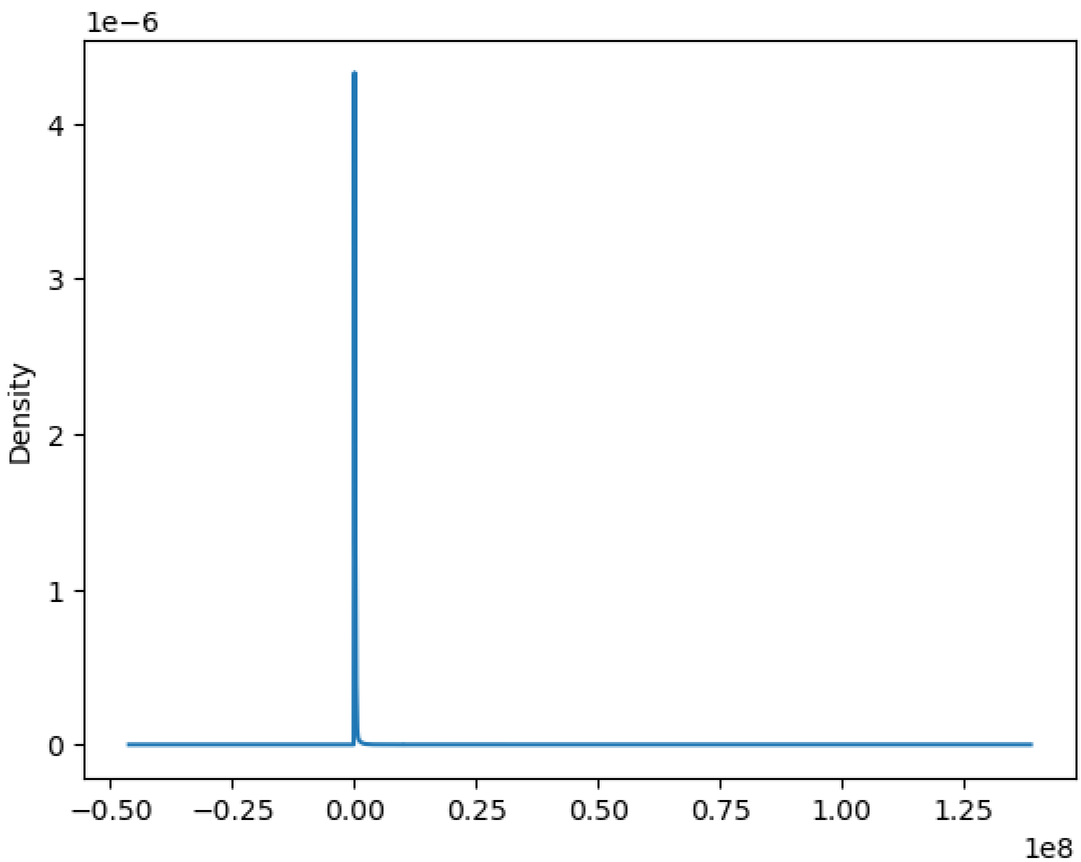
```
Out[18]: 179861.90354912292
```

```
In [19]: data.amount.std()
```

```
Out[19]: 603858.2314629931
```

```
In [20]: data.amount.plot.kde()
```

```
Out[20]: <AxesSubplot:ylabel='Density'>
```




```
In [21]: # the data does't follow n-dist so we can use normalization instead of standadization
         # normalize
         df_norm = pd.DataFrame(columns=[x_train.columns])
```

In [22]:
```
df_norm['amount'] = (data['amount'] - data['amount'].min()) / (data['amount'].max() -dat
```

In [23]:
```
df_norm['oldbalanceOrg'] = (data['oldbalanceOrg'] - data['oldbalanceOrg'].min()) / (data
df_norm['newbalanceOrig'] = (data['newbalanceOrig'] - data['newbalanceOrig'].min()) / (d
df_norm['oldbalanceDest'] = (data['oldbalanceDest'] - data['oldbalanceDest'].min()) / (d
df_norm['newbalanceDest'] = (data['newbalanceDest'] - data['newbalanceDest'].min()) / (d
```

In [24]:
```
df_norm['above200000'] = data.above200000
```

In [25]:
```
# new dataframe , it is scaled
df_norm
```

Out[25]:

	amount	oldbalanceOrg	newbalanceOrig	oldbalanceDest	newbalanceDest	above200000
0	0.000106	0.002855	0.003233	0.000000	0.000000	0
1	0.000020	0.000357	0.000391	0.000000	0.000000	0
2	0.000002	0.000003	0.000000	0.000000	0.000000	0
3	0.000002	0.000003	0.000000	0.000059	0.000000	0
4	0.000126	0.000697	0.000603	0.000000	0.000000	0
...	...	...	...	...	...	...
6362615	0.003674	0.005701	0.000000	0.000000	0.000954	1
6362616	0.068272	0.105923	0.000000	0.000000	0.000000	1
6362617	0.068272	0.105923	0.000000	0.000192	0.017912	1
6362618	0.009195	0.014265	0.000000	0.000000	0.000000	1
6362619	0.009195	0.014265	0.000000	0.018286	0.020664	1

6362620 rows × 6 columns

In [26]:
```
# split the data for training and validation
from sklearn.model_selection import train_test_split
X_train,X_test, Y_train,Y_test  = train_test_split(df_norm, data.isFraud,test_size=0.2)
```

In [27]:
```
Y_train.shape
```

Out[27]:
```
(5090096,)
```

In [28]:
```
X_test.shape
```

Out[28]:
```
(1272524, 6)
```

In [29]:
```
from sklearn.ensemble import RandomForestClassifier
```

In [30]:
```
classifier = RandomForestClassifier()
classifier.fit(X_train, Y_train)
```

```
C:\Users\goura\anaconda3\lib\site-packages\sklearn\utils\validation.py:1688: FutureWarni
ng: Feature names only support names that are all strings. Got feature names with dtype
s: ['tuple']. An error will be raised in 1.2.
  warnings.warn(
```

Out[30]:
```
RandomForestClassifier()
```

In [31]:
```
classifier.get_params()
```

Out[31]:
```
{'bootstrap': True,
 'ccp_alpha': 0.0,
 'class_weight': None,
 'criterion': 'gini',
 'max_depth': None,
 'max_features': 'auto',
 'max_leaf_nodes': None,
 'max_samples': None,
 'min_impurity_decrease': 0.0,
 'min_samples_leaf': 1,
 'min_samples_split': 2,
 'min_weight_fraction_leaf': 0.0,
 'n_estimators': 100,
 'n_jobs': None,
 'oob_score': False,
 'random_state': None,
 'verbose': 0,
 'warm_start': False}
```

In [32]:
```
pred = classifier.predict(X_test)
```

```
C:\Users\goura\anaconda3\lib\site-packages\sklearn\utils\validation.py:1688: FutureWarni
ng: Feature names only support names that are all strings. Got feature names with dtype
s: ['tuple']. An error will be raised in 1.2.
  warnings.warn(
```

In [33]:
```
pred
```

Out[33]:
```
array([0, 0, 0, ..., 0, 0, 0], dtype=int64)
```

In [34]:
```
from sklearn.metrics import confusion_matrix,accuracy_score,f1_score,classification_repo
```

In [35]:
```
confusion_matrix(Y_test,pred)
```

Out[35]:
```
array([[1270832,      44],
       [    581,    1067]], dtype=int64)
```

In [36]:
```
accuracy_score(Y_test,pred)
```

Out[36]:
```
0.9995088501277776
```

In [37]:
```
precision_score(Y_test,pred)
```

Out[37]:
```
0.9603960396039604
```

In [38]:
```
recall_score(Y_test,pred)
```

Out[38]:
```
0.6474514563106796
```

In [39]:
```
f1_score(Y_test,pred)
```

Out[39]:
```
0.7734686480608916
```

In [40]:
```
print(classification_report(Y_test,pred))
```

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00   1270876
           1       0.96      0.65      0.77      1648

    accuracy                           1.00   1272524
   macro avg       0.98      0.82      0.89   1272524
```

```
weighted avg       1.00      1.00      1.00   1272524
```

In [ ]:
```python
from sklearn.ensemble import GradientBoostingClassifier
```

In [ ]:
```python
classifier2 = GradientBoostingClassifier()
classifier2.fit(X_train, Y_train)
pred2 = classifier2.predict(X_test)
```

```
C:\Users\goura\anaconda3\lib\site-packages\sklearn\utils\validation.py:1688: FutureWarni
ng: Feature names only support names that are all strings. Got feature names with dtype
s: ['tuple']. An error will be raised in 1.2.
  warnings.warn(
C:\Users\goura\anaconda3\lib\site-packages\sklearn\utils\validation.py:1688: FutureWarni
ng: Feature names only support names that are all strings. Got feature names with dtype
s: ['tuple']. An error will be raised in 1.2.
  warnings.warn(
```

In [ ]:
```python
confusion_matrix(Y_test,pred2)
```

Out[ ]:
```
array([[1270860,      16],
       [   1614,      34]], dtype=int64)
```

In [ ]:
```python
accuracy_score(Y_test,pred2)
```

Out[ ]:
```
0.9987190811332438
```

In [ ]:
```python
print(classification_report(Y_test,pred2))
```

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00   1270876
           1       0.68      0.02      0.04      1648

    accuracy                           1.00   1272524
   macro avg       0.84      0.51      0.52   1272524
weighted avg       1.00      1.00      1.00   1272524
```

In [ ]:
```python
# Hyperparameter tunning
from sklearn.model_selection import RandomizedSearchCV,GridSearchCV
```

In [51]:
```python
from sklearn.metrics import roc_auc_score, confusion_matrix

# Get predicted probabilities for fraud
y_pred_proba = classifier.predict_proba(X_test)[:,1]

# Find optimal threshold using roc_auc_score
threshold = 0
best_score = 0
for i in np.arange(0,1,0.05):
    y_pred_val = y_pred_proba > i
    score = roc_auc_score(Y_test, y_pred_val)
    if score > best_score:
        best_score = score
        threshold = i

# Use the optimal threshold to make predictions
y_pred = y_pred_proba > threshold

# Get the recall score
recall = recall_score(Y_test, y_pred)
print("Recall: ", recall)

# Get the confusion matrix
```

```
conf_mat = confusion_matrix(Y_test, y_pred)
print("Confusion matrix: \n", conf_mat)
```

```
C:\Users\goura\anaconda3\lib\site-packages\sklearn\utils\validation.py:1688: FutureWarni
ng: Feature names only support names that are all strings. Got feature names with dtype
s: ['tuple']. An error will be raised in 1.2.
  warnings.warn(
Recall:  0.820995145631068
Confusion matrix:
 [[1269974     902]
 [    295    1353]]
```

In [52]:
```
print(classification_report(Y_test, y_pred));
```

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00   1270876
           1       0.60      0.82      0.69      1648

    accuracy                           1.00   1272524
   macro avg       0.80      0.91      0.85   1272524
weighted avg       1.00      1.00      1.00   1272524
```

In [ ]:
```
from imblearn.over_sampling import SMOTE
```

In [57]:
```
print(classification_report(Y_test,pred))
```

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00   1270876
           1       0.96      0.65      0.77      1648

    accuracy                           1.00   1272524
   macro avg       0.98      0.82      0.89   1272524
weighted avg       1.00      1.00      1.00   1272524
```

1. Data cleaning including missing values, outliers and multi-collinearity.

For data cleaning, I would first check for missing values and remove any rows with missing- values. I would then check for outliers in the amount column and remove any extreme values that do not fall within a certain range. I would also check for multi-collinearity among the variables and remove any highly correlated variables.

2. Describe your fraud detection model in elaboration.

I would use a supervised machine learning model, such as Random Forest or GB classifier, to classify transactions as fraudulent or non-fraudulent. I would also use feature engineering to create new variables that may be useful in detecting fraud, such as the ratio of the transaction amount to the initial balance or amount thrushold.

3. How did you select variables to be included in the model?

I would select variables to be included in the model based on their correlation with the target variable (isFraud) and their importance in detecting fraud. Variables such as type, amount, and the initial and final balances of both the customer and the recipient would be important for the model.

4. Demonstrate the performance of the model by using best set of tools.

I would use metrics such as precision, recall, F1 score, accuracy to evaluate the performance of the model. I would also use cross-validation to ensure that the model is not overfitting to the training data.

5. What are the key factors that predict fraudulent customer?

The key factors that predict fraudulent customers would likely include high transaction amounts, sudden changes in account balances, and transactions involving multiple parties.

6. Do these factors make sense? If yes, How? If not, How not?

These factors make sense as they are indicative of suspicious or unusual activity that may be indicative of fraud. High transaction amounts and sudden changes in account balances may be indicative of an attempt to steal funds, while transactions involving multiple parties may be indicative of money laundering or other illegal activities.

7. What kind of prevention should be adopted while company update its infrastructure?

To prevent fraud, the company could implement measures such as transaction monitoring, account monitoring, and two-factor authentication. Additionally, the company could also train its employees to identify and report suspicious activity.

8. Assuming these actions have been implemented, how would you determine if they work?

To determine if these actions are effective, the company could track the number of fraudulent transactions before and after the implementation of these measures. Additionally, the company could also conduct regular audits and assessments to identify any areas where the system may be vulnerable to fraud.

In [ ]: