In [1]:

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

In [2]:

```
dataset=pd.read_csv("fraud.csv")
```

In [3]:

```
dataset
```

Out[3]:

	step	type	amount	oldbalanceOrg	newbalanceOrig	oldbalanceDest	newbalar
0	1	PAYMENT	9839.64	170136.00	160296.36	0.00	
1	1	PAYMENT	1864.28	21249.00	19384.72	0.00	
2	1	PAYMENT	11668.14	41554.00	29885.86	0.00	
3	1	PAYMENT	7817.71	53860.00	46042.29	0.00	
4	1	PAYMENT	7107.77	183195.00	176087.23	0.00	
...	...	...	...	...	...	...	
16421	743	CASH_OUT	339682.13	339682.13	0.00	0.00	33
16422	743	TRANSFER	6311409.28	6311409.28	0.00	0.00	
16423	743	CASH_OUT	6311409.28	6311409.28	0.00	68488.84	637
16424	743	TRANSFER	850002.52	850002.52	0.00	0.00	
16425	743	CASH_OUT	850002.52	850002.52	0.00	6510099.11	736

16426 rows × 9 columns

In [4]:

```
dataset=dataset.drop("isFlaggedFraud",axis=1)
```

In [5]:

```
dataset
```

Out[5]:

	step	type	amount	oldbalanceOrg	newbalanceOrig	oldbalanceDest	newbalar
0	1	PAYMENT	9839.64	170136.00	160296.36	0.00	
1	1	PAYMENT	1864.28	21249.00	19384.72	0.00	
2	1	PAYMENT	11668.14	41554.00	29885.86	0.00	
3	1	PAYMENT	7817.71	53860.00	46042.29	0.00	
4	1	PAYMENT	7107.77	183195.00	176087.23	0.00	
...	...	...	...	...	...	...	
16421	743	CASH_OUT	339682.13	339682.13	0.00	0.00	33
16422	743	TRANSFER	6311409.28	6311409.28	0.00	0.00	
16423	743	CASH_OUT	6311409.28	6311409.28	0.00	68488.84	637
16424	743	TRANSFER	850002.52	850002.52	0.00	0.00	
16425	743	CASH_OUT	850002.52	850002.52	0.00	6510099.11	736

16426 rows × 8 columns

In [6]:

```
dataset=pd.get_dummies(dataset,drop_first=True)
```

In [7]:

```
dataset
```

Out[7]:

	step	amount	oldbalanceOrg	newbalanceOrig	oldbalanceDest	newbalanceDest	isFr
0	1	9839.64	170136.00	160296.36	0.00	0.00	
1	1	1864.28	21249.00	19384.72	0.00	0.00	
2	1	11668.14	41554.00	29885.86	0.00	0.00	
3	1	7817.71	53860.00	46042.29	0.00	0.00	
4	1	7107.77	183195.00	176087.23	0.00	0.00	
...	...	...	...	...	...	...	
16421	743	339682.13	339682.13	0.00	0.00	339682.13	
16422	743	6311409.28	6311409.28	0.00	0.00	0.00	
16423	743	6311409.28	6311409.28	0.00	68488.84	6379898.11	
16424	743	850002.52	850002.52	0.00	0.00	0.00	
16425	743	850002.52	850002.52	0.00	6510099.11	7360101.63	

16426 rows × 11 columns

In [8]:

```
indep=dataset[["step","amount","oldbalanceOrg","newbalanceOrig","oldbalanceDest","newbalanc
dep=dataset[["isFraud"]]
```

In [9]:

```
indep
```

Out[9]:

	step	amount	oldbalanceOrg	newbalanceOrig	oldbalanceDest	newbalanceDest	type
0	1	9839.64	170136.00	160296.36	0.00	0.00	
1	1	1864.28	21249.00	19384.72	0.00	0.00	
2	1	11668.14	41554.00	29885.86	0.00	0.00	
3	1	7817.71	53860.00	46042.29	0.00	0.00	
4	1	7107.77	183195.00	176087.23	0.00	0.00	
...	...	...	...	...	...	...	
16421	743	339682.13	339682.13	0.00	0.00	339682.13	
16422	743	6311409.28	6311409.28	0.00	0.00	0.00	
16423	743	6311409.28	6311409.28	0.00	68488.84	6379898.11	
16424	743	850002.52	850002.52	0.00	0.00	0.00	
16425	743	850002.52	850002.52	0.00	6510099.11	7360101.63	

16426 rows × 10 columns

In [10]:

```
dep
```

Out[10]:

	isFraud
0	0
1	0
2	0
3	0
4	0
...	...
16421	1
16422	1
16423	1
16424	1
16425	1

16426 rows × 1 columns

In [11]:

```
from sklearn.model_selection import train_test_split
X_train,X_test,Y_train,Y_test=train_test_split(indep,dep,test_size=1/3,random_state=0)
```

In [12]:

```
X_train
```

Out[12]:

	step	amount	oldbalanceOrg	newbalanceOrig	oldbalanceDest	newbalanceDest	typ
14402	561	2763025.35	2763025.35	0.00	0.00	0.00	
11624	302	633384.46	633384.46	0.00	0.00	0.00	
6422	6	9396.00	936212.00	926816.00	0.00	0.00	
9427	103	710385.55	710385.55	0.00	0.00	0.00	
5126	5	59691.76	0.00	0.00	4254247.97	4313939.72	
...	...	...	...	...	...	...	
9225	84	152331.26	152331.26	0.00	0.00	0.00	
13123	438	4626269.69	4626269.69	0.00	4501183.12	9127452.81	
9845	145	1767886.02	1767886.02	0.00	0.00	0.00	
10799	230	10000000.00	13774746.08	3774746.08	0.00	0.00	
2732	2	147141.81	1398839.54	1545981.34	221285.85	74144.04	

10950 rows × 10 columns

In [13]:

```
X_test
```

Out[13]:

	step	amount	oldbalanceOrg	newbalanceOrig	oldbalanceDest	newbalanceDest	type
11741	313	5818863.13	5818863.13	0.00	227196.79	6046059.92	
13248	449	133632.24	133632.24	0.00	0.00	0.00	
7082	6	37887.08	344.00	0.00	0.00	0.00	
12419	378	187761.76	187761.76	0.00	360353.54	687445.20	
6725	6	22923.28	0.00	0.00	0.00	0.00	
...	...	...	...	...	...	...	
10870	235	1369544.19	1369544.19	0.00	0.00	0.00	
6739	6	13916.14	0.00	0.00	0.00	0.00	
11628	303	447009.83	447009.83	0.00	0.00	0.00	
6178	6	8793.16	836517.00	827723.84	0.00	0.00	
15786	689	276253.24	276253.24	0.00	0.00	0.00	

5476 rows × 10 columns

In [45]:

```
from sklearn.naive_bayes import BernoulliNB
classifier=BernoulliNB()
classifier.fit(X_train,Y_train)
```

```
C:\Users\athis\Anaconda3\lib\site-packages\sklearn\utils\validation.py:724:
DataConversionWarning: A column-vector y was passed when a 1d array was expe
cted. Please change the shape of y to (n_samples, ), for example using ravel
().
  y = column_or_1d(y, warn=True)
```

Out[45]:

```
BernoulliNB(alpha=1.0, binarize=0.0, class_prior=None, fit_prior=True)
```

In [46]:

```
Y_pred=classifier.predict(X_test)
```

In [47]:

```
Y_pred
```

Out[47]:

```
array([1, 1, 0, ..., 1, 0, 1], dtype=int64)
```

In [48]:

```
from sklearn.metrics import confusion_matrix
cm=confusion_matrix(Y_test,Y_pred)
print(cm)
```

```
[[2545  177]
 [  64 2690]]
```

In [49]:

```
from sklearn.metrics import classification_report
cr=classification_report(Y_test,Y_pred)
print(cr)
```

```
              precision    recall  f1-score   support

           0       0.98      0.93      0.95      2722
           1       0.94      0.98      0.96      2754

    accuracy                           0.96      5476
   macro avg       0.96      0.96      0.96      5476
weighted avg       0.96      0.96      0.96      5476
```

Accuracy for Naives' bayes Algorithm (Done in BernoulliNB) = 0.96

In [50]:

```
result=classifier.predict([[743,339682.13,339682.13,0.00,0.00,339682.13,1,0,0,0]])
```

In [51]:

```
result
```

Out[51]:

```
array([1], dtype=int64)
```

In [ ]: