In [2]:
```
pip install joblib
```

```
Requirement already satisfied: joblib in c:\users\nadee\anaconda\lib\site-packages (1.1.
0)
Note: you may need to restart the kernel to use updated packages.
```

In [2]:
```
pip install scikit-learn --upgrade
```

```
Requirement already satisfied: scikit-learn in c:\users\nadee\anaconda\lib\site-packages
(1.0.2)
Collecting scikit-learn
  Downloading scikit_learn-1.3.0-cp39-cp39-win_amd64.whl (9.3 MB)
Note: you may need to restart the kernel to use updated packages.
Requirement already satisfied: threadpoolctl>=2.0.0 in c:\users\nadee\anaconda\lib\site-
packages (from scikit-learn) (2.2.0)
Collecting joblib>=1.1.1
  Downloading joblib-1.3.1-py3-none-any.whl (301 kB)
Requirement already satisfied: scipy>=1.5.0 in c:\users\nadee\anaconda\lib\site-packages
(from scikit-learn) (1.7.3)
Requirement already satisfied: numpy>=1.17.3 in c:\users\nadee\anaconda\lib\site-package
s (from scikit-learn) (1.21.5)
Installing collected packages: joblib, scikit-learn
  Attempting uninstall: joblib
    Found existing installation: joblib 1.1.0
    Uninstalling joblib-1.1.0:
      Successfully uninstalled joblib-1.1.0
  Attempting uninstall: scikit-learn
    Found existing installation: scikit-learn 1.0.2
    Uninstalling scikit-learn-1.0.2:
      Successfully uninstalled scikit-learn-1.0.2
Successfully installed joblib-1.3.1 scikit-learn-1.3.0
```

In [4]:
```
# main libraries
import pandas as pd
import numpy as np
import time

# visual libraries
from matplotlib import pyplot as plt
import matplotlib.gridspec as gridspec
import seaborn as sns
from mpl_toolkits.mplot3d import Axes3D
plt.style.use('ggplot')

# sklearn libraries
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import normalize
from sklearn.metrics import confusion_matrix,accuracy_score,precision_score,recall_score
#from sklearn.externals import joblib
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
```

In [5]:
```
##Reading the dataset

df = pd.read_csv("D:\Accredian task\Fraud.csv")
```

In [7]:
```
df.head(2)
```

Out[7]:

	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	

In [8]:
```
df.shape
```

Out[8]:
```
(6362620, 11)
```

In [9]:
```
df.isnull().sum()
```

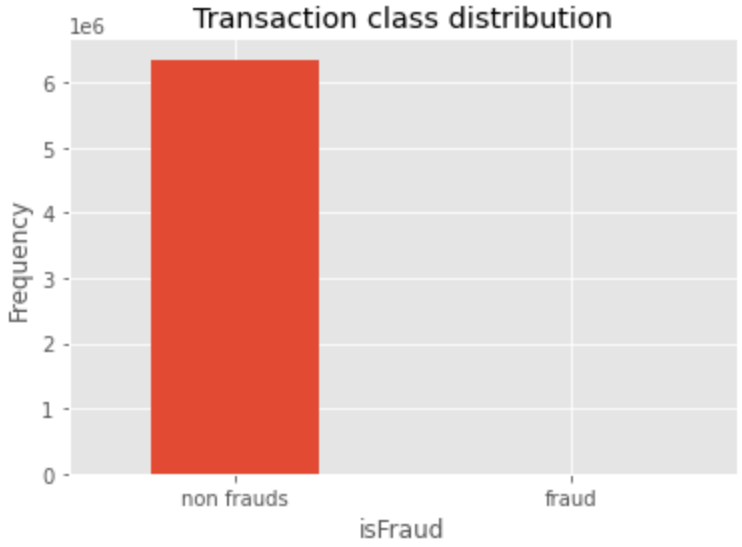
Out[9]:
```
step              0
type              0
amount            0
nameOrig          0
oldbalanceOrg     0
newbalanceOrig    0
nameDest          0
oldbalanceDest    0
newbalanceDest    0
isFraud           0
isFlaggedFraud    0
dtype: int64
```

In [10]:
```
All = df.shape[0]
fraud = df[df['isFraud'] == 1]
nonFraud = df[df['isFraud'] == 0]

x = len(fraud)/All
y = len(nonFraud)/All

print('frauds :',x*100,'%')
print('non frauds :',y*100,'%')
```

```
frauds : 0.12908204481801522 %
non frauds : 99.87091795518198 %
```

In [11]:
```
# Let's plot the Transaction class against the Frequency
labels = ['non frauds','fraud']
classes = pd.value_counts(df['isFraud'], sort = True)
classes.plot(kind = 'bar', rot=0)
plt.title("Transaction class distribution")
plt.xticks(range(2), labels)
plt.xlabel("isFraud")
plt.ylabel("Frequency")
```

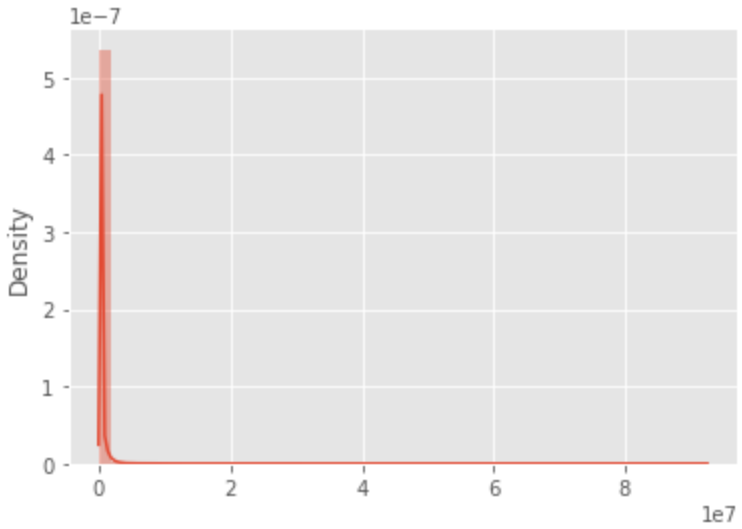
Out[11]:
```
Text(0, 0.5, 'Frequency')
```



In [12]:
```
# distribution of Amount
amount = [df['amount'].values]
sns.distplot(amount)
```

```
C:\Users\nadee\Anaconda\lib\site-packages\seaborn\distributions.py:2619: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

Out[12]:
```
<AxesSubplot:ylabel='Density'>
```



In [18]:
```
df2 = df.drop(['type','nameOrig','nameDest'],axis=1)
```

In [19]:
```
df2.head()
```

Out[19]:

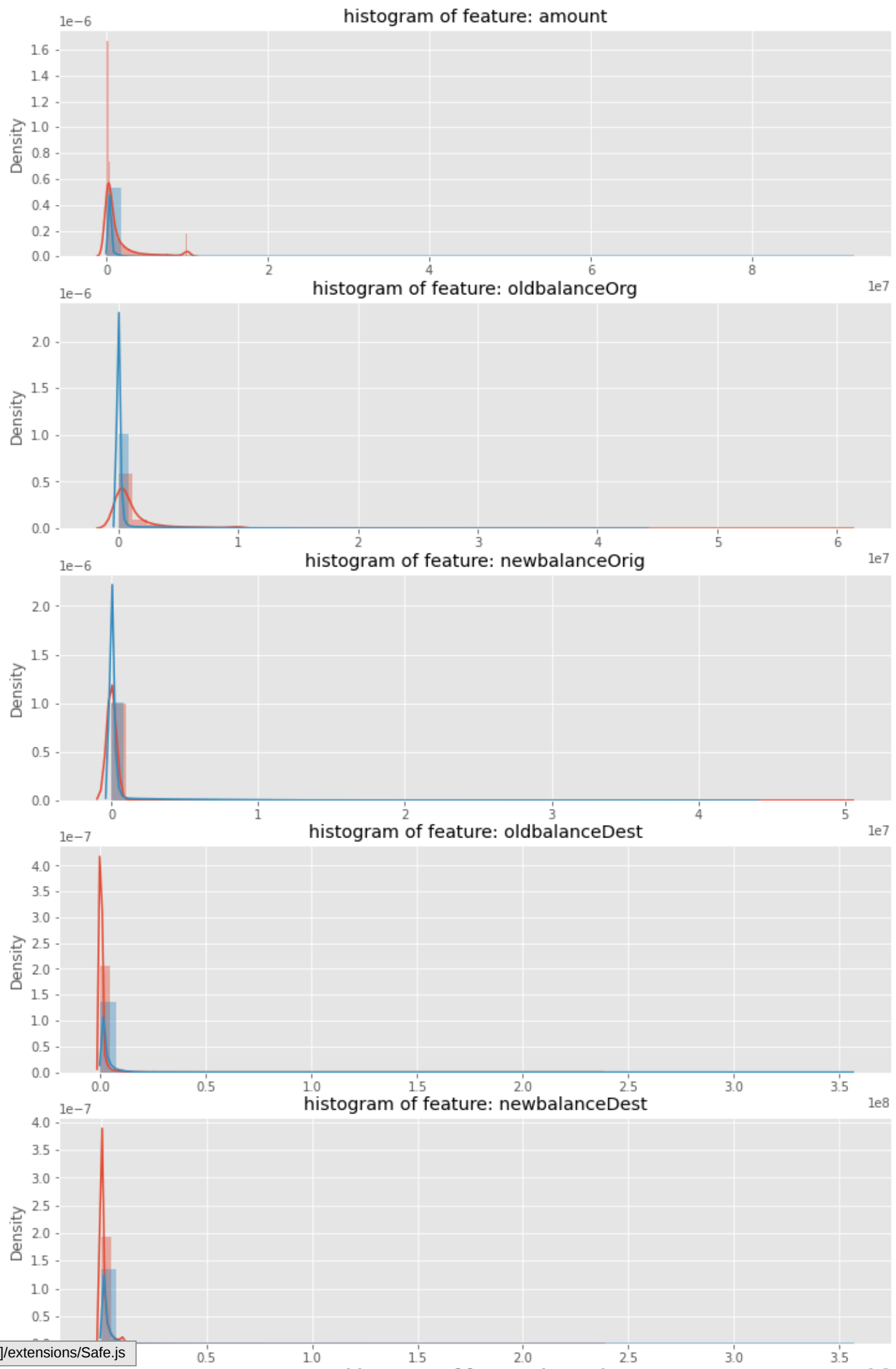
	step	amount	oldbalanceOrg	newbalanceOrig	oldbalanceDest	newbalanceDest	isFraud	isFlaggedFraud
0	1	9839.64	170136.0	160296.36	0.0	0.0	0	0
1	1	1864.28	21249.0	19384.72	0.0	0.0	0	0
2	1	181.00	181.0	0.00	0.0	0.0	1	0
3	1	181.00	181.0	0.00	21182.0	0.0	1	0
4	1	11668.14	41554.0	29885.86	0.0	0.0	0	0

In [20]:
```
# distribution of anomalous features
atures = df2.iloc[:,1:29].columns
```

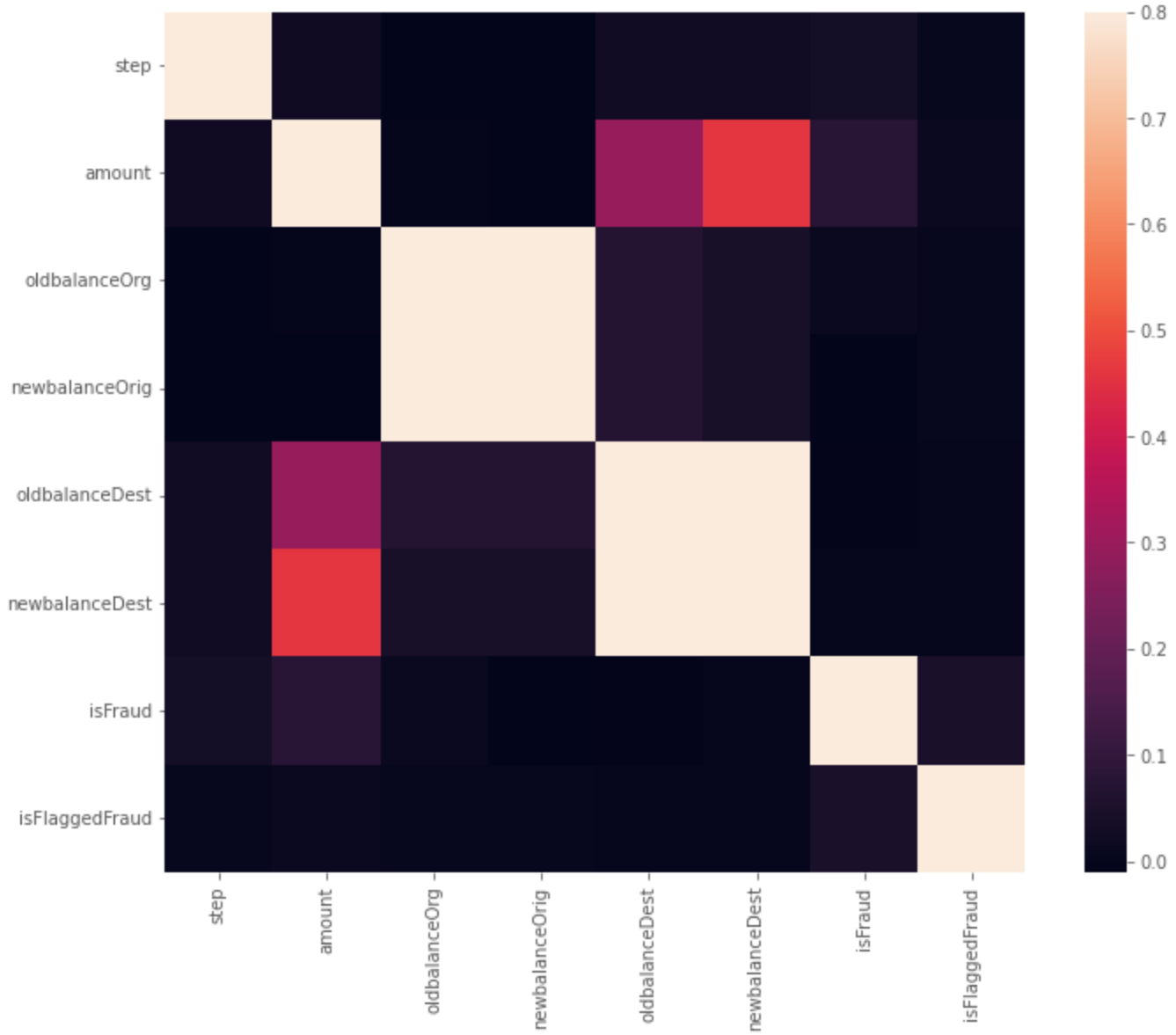
```
plt.figure(figsize=(12,28*4))
gs = gridspec.GridSpec(28, 1)
for i, cn in enumerate(df2[anomalous_features]):
    ax = plt.subplot(gs[i])
    sns.distplot(df[cn][df2.isFraud == 1], bins=50)
    sns.distplot(df[cn][df2.isFraud == 0], bins=50)
    ax.set_xlabel('')
    ax.set_title('histogram of feature: ' + str(cn))
plt.show()
```

```
C:\Users\nadee\Anaconda\lib\site-packages\seaborn\distributions.py:2619: FutureWarning:
`distplot` is a deprecated function and will be removed in a future version. Please adap
t your code to use either `displot` (a figure-level function with similar flexibility) o
r `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
C:\Users\nadee\Anaconda\lib\site-packages\seaborn\distributions.py:2619: FutureWarning:
`distplot` is a deprecated function and will be removed in a future version. Please adap
t your code to use either `displot` (a figure-level function with similar flexibility) o
r `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
C:\Users\nadee\Anaconda\lib\site-packages\seaborn\distributions.py:2619: FutureWarning:
`distplot` is a deprecated function and will be removed in a future version. Please adap
t your code to use either `displot` (a figure-level function with similar flexibility) o
r `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
C:\Users\nadee\Anaconda\lib\site-packages\seaborn\distributions.py:2619: FutureWarning:
`distplot` is a deprecated function and will be removed in a future version. Please adap
t your code to use either `displot` (a figure-level function with similar flexibility) o
r `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
C:\Users\nadee\Anaconda\lib\site-packages\seaborn\distributions.py:2619: FutureWarning:
`distplot` is a deprecated function and will be removed in a future version. Please adap
t your code to use either `displot` (a figure-level function with similar flexibility) o
r `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
C:\Users\nadee\Anaconda\lib\site-packages\seaborn\distributions.py:2619: FutureWarning:
`distplot` is a deprecated function and will be removed in a future version. Please adap
t your code to use either `displot` (a figure-level function with similar flexibility) o
r `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
C:\Users\nadee\Anaconda\lib\site-packages\seaborn\distributions.py:2619: FutureWarning:
`distplot` is a deprecated function and will be removed in a future version. Please adap
t your code to use either `displot` (a figure-level function with similar flexibility) o
r `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
C:\Users\nadee\Anaconda\lib\site-packages\seaborn\distributions.py:2619: FutureWarning:
`distplot` is a deprecated function and will be removed in a future version. Please adap
t your code to use either `displot` (a figure-level function with similar flexibility) o
r `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
C:\Users\nadee\Anaconda\lib\site-packages\seaborn\distributions.py:2619: FutureWarning:
`distplot` is a deprecated function and will be removed in a future version. Please adap
t your code to use either `displot` (a figure-level function with similar flexibility) o
r `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
C:\Users\nadee\Anaconda\lib\site-packages\seaborn\distributions.py:2619: FutureWarning:
`distplot` is a deprecated function and will be removed in a future version. Please adap
t your code to use either `displot` (a figure-level function with similar flexibility) o
r `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
C:\Users\nadee\Anaconda\lib\site-packages\seaborn\distributions.py:2619: FutureWarning:
`distplot` is a deprecated function and will be removed in a future version. Please adap
t your code to use either `displot` (a figure-level function with similar flexibility) o
r `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
C:\Users\nadee\Anaconda\lib\site-packages\seaborn\distributions.py:316: UserWarning: Dat
aset has 0 variance; skipping density estimate. Pass `warn_singular=False` to disable th
is warning.
  warnings.warn(msg, UserWarning)
C:\Users\nadee\Anaconda\lib\site-packages\seaborn\distributions.py:2619: FutureWarning:
`distplot` is a deprecated function and will be removed in a future version. Please adap
t your code to use either `displot` (a figure-level function with similar flexibility) o
r `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

```
C:\Users\nadee\Anaconda\lib\site-packages\seaborn\distributions.py:316: UserWarning: Dat
aset has 0 variance; skipping density estimate. Pass `warn_singular=False` to disable th
is warning.
  warnings.warn(msg, UserWarning)
C:\Users\nadee\Anaconda\lib\site-packages\seaborn\distributions.py:2619: FutureWarning:
`distplot` is a deprecated function and will be removed in a future version. Please adap
t your code to use either `displot` (a figure-level function with similar flexibility) o
r `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
C:\Users\nadee\Anaconda\lib\site-packages\seaborn\distributions.py:2619: FutureWarning:
`distplot` is a deprecated function and will be removed in a future version. Please adap
t your code to use either `displot` (a figure-level function with similar flexibility) o
r `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
C:\Users\nadee\Anaconda\lib\site-packages\seaborn\distributions.py:316: UserWarning: Dat
aset has 0 variance; skipping density estimate. Pass `warn_singular=False` to disable th
is warning.
  warnings.warn(msg, UserWarning)
```



In [24]:
```
# heat map of correlation of features
correlation_matrix = df2.corr()
fig = plt.figure(figsize=(12,9))
sns.heatmap(correlation_matrix,vmax=0.8,square = True)
plt.show()
```



In [29]:
```
df2.columns
```

Out[29]:
```
Index(['step', 'oldbalanceOrg', 'newbalanceOrig', 'oldbalanceDest',
       'newbalanceDest', 'isFraud', 'isFlaggedFraud', 'Vamount'],
      dtype='object')
```

In [30]:
```
df.Vamount
```

Out[30]:
```
0          -0.281560
1          -0.294767
2          -0.297555
3          -0.297555
4          -0.278532
              ...
6362615     0.264665
6362616    10.153953
6362617    10.153953
6362618     1.109765
6362619     1.109765
Name: Vamount, Length: 6362620, dtype: float64
```

In [ ]:
```

```

In [32]:
```
X = df2.drop(['isFraud'], axis = 1)
y = df2['isFraud']

omponents=2)
```

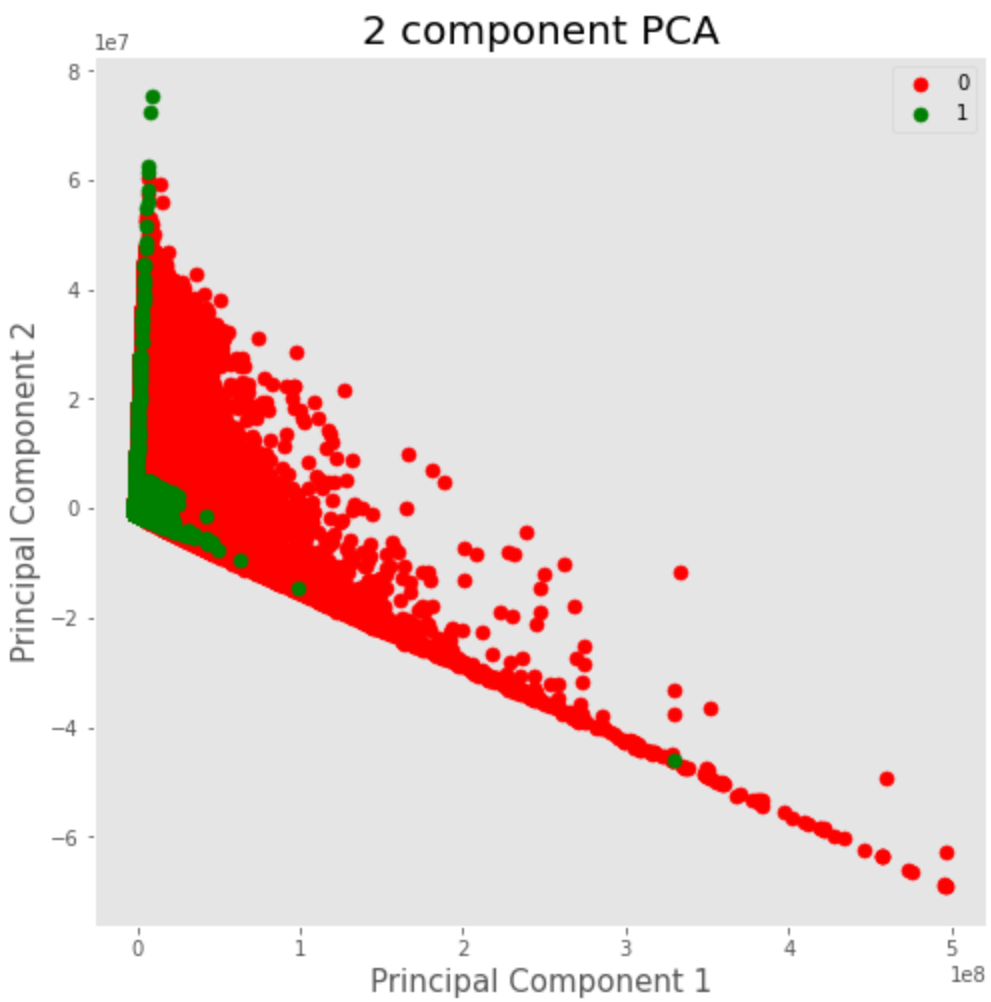
```
principalComponents = pca.fit_transform(X.values)
principalDf = pd.DataFrame(data = principalComponents
             , columns = ['principal component 1', 'principal component 2'])
```

In [33]:
```
finalDf = pd.concat([principalDf, y], axis = 1)
finalDf.head()
```

Out[33]:

	principal component 1	principal component 2	isFraud
0	-1.761912e+06	-727735.538528	0
1	-1.789736e+06	-930692.073601	0
2	-1.793620e+06	-959020.140622	1
3	-1.779360e+06	-960729.657342	1
4	-1.786785e+06	-909155.830553	0

In [35]:
```
# 2D visualization
fig = plt.figure(figsize = (8,8))
ax = fig.add_subplot(1,1,1)
ax.set_xlabel('Principal Component 1', fontsize = 15)
ax.set_ylabel('Principal Component 2', fontsize = 15)
ax.set_title('2 component PCA', fontsize = 20)
targets = [0, 1]
colors = ['r', 'g']
for target, color in zip(targets,colors):
    indicesToKeep = finalDf['isFraud'] == target
    ax.scatter(finalDf.loc[indicesToKeep, 'principal component 1']
               , finalDf.loc[indicesToKeep, 'principal component 2']
               , c = color
               , s = 50)
ax.legend(targets)
ax.grid()
```



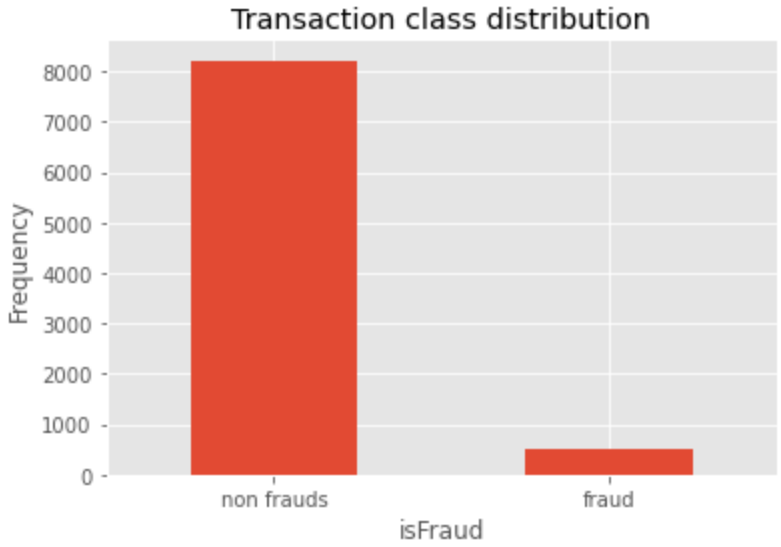
In [37]:
```
# Lets shuffle the data before creating the subsamples
df2 = df2.sample(frac=1)

frauds = df2[df2['isFraud'] == 1]
non_frauds = df2[df2['isFraud'] == 0][:492]

new_df = pd.concat([non_frauds, frauds])
# Shuffle dataframe rows
new_df = new_df.sample(frac=1, random_state=42)

new_df.head()
```

Out[37]:

	step	oldbalanceOrg	newbalanceOrig	oldbalanceDest	newbalanceDest	isFraud	isFlaggedFraud	Vam
5563764	390	1195842.37	0.0	0.00	1195842.37	1	0	1.68
6280929	644	4438931.81	0.0	20066.68	4458998.50	1	0	7.05
1051569	95	3492056.91	0.0	0.00	0.00	1	0	5.48
12467	7	441445.58	0.0	0.00	0.00	1	0	0.43
2849894	227	114889.61	0.0	0.00	0.00	1	0	-0.10

In [38]:
```
# Let's plot the Transaction class against the Frequency
labels = ['non frauds','fraud']
classes = pd.value_counts(new_df['isFraud'], sort = True)
classes.plot(kind = 'bar', rot=0)
plt.title("Transaction class distribution")
plt.xticks(range(2), labels)
plt.xlabel("isFraud")
plt.ylabel("Frequency")
```

Out[38]: Text(0, 0.5, 'Frequency')

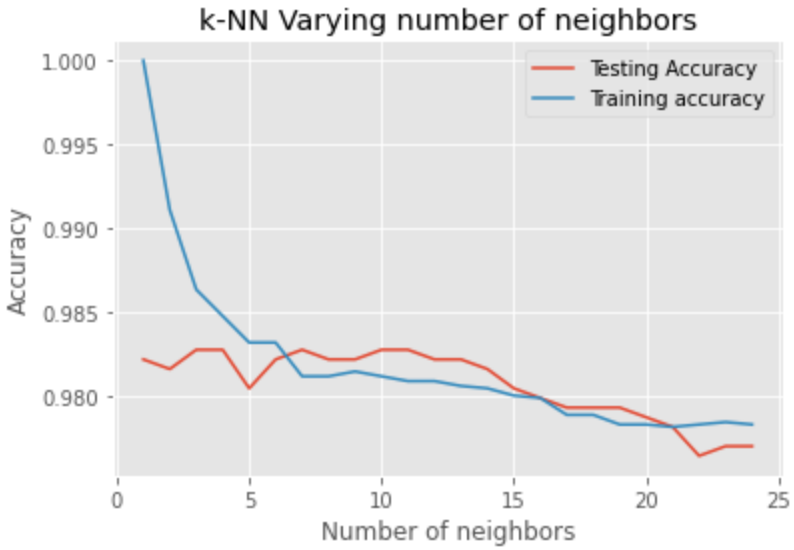


In [39]:
```
# prepare the data
features = new_df.drop(['isFraud'], axis = 1)
labels = pd.DataFrame(new_df['isFraud'])

feature_array = features.values
label_array = labels.values
```

In [40]:
```
# splitting the faeture array and label array keeping 80% for the trainnig sets
X_train,X_test,y_train,y_test = train_test_split(feature_array,label_array,test_size=0.2

# normalize: Scale input vectors individually to unit norm (vector length).
X_train = normalize(X_train)
X_test=normalize(X_test)
```

In [41]:
```
neighbours = np.arange(1,25)
train_accuracy =np.empty(len(neighbours))
test_accuracy = np.empty(len(neighbours))

for i,k in enumerate(neighbours):
    #Setup a knn classifier with k neighbors
    knn=KNeighborsClassifier(n_neighbors=k,algorithm="kd_tree",n_jobs=-1)

    #Fit the model
    knn.fit(X_train,y_train.ravel())

    #Compute accuracy on the training set
    train_accuracy[i] = knn.score(X_train, y_train.ravel())

    #Compute accuracy on the test set
    test_accuracy[i] = knn.score(X_test, y_test.ravel())
```

In [42]:
```
#Generate plot
plt.title('k-NN Varying number of neighbors')
plt.plot(neighbours, test_accuracy, label='Testing Accuracy')
plt.plot(neighbours, train_accuracy, label='Training accuracy')
plt.legend()
plt.xlabel('Number of neighbors')
plt.ylabel('Accuracy')
plt.show()
```



In [43]:
```
idx = np.where(test_accuracy == max(test_accuracy))
x = neighbours[idx]

#k_nearest_neighbours_classification
knn=KNeighborsClassifier(n_neighbors=x[0],algorithm="kd_tree",n_jobs=-1)
knn.fit(X_train,y_train.ravel())
```

Out[43]:
```
KNeighborsClassifier(algorithm='kd_tree', n_jobs=-1, n_neighbors=3)
```

In [45]:
```
import joblib
```

In [46]:
```
# save the model to disk
filename = 'finalized_model.sav'
joblib.dump(knn, filename)
```

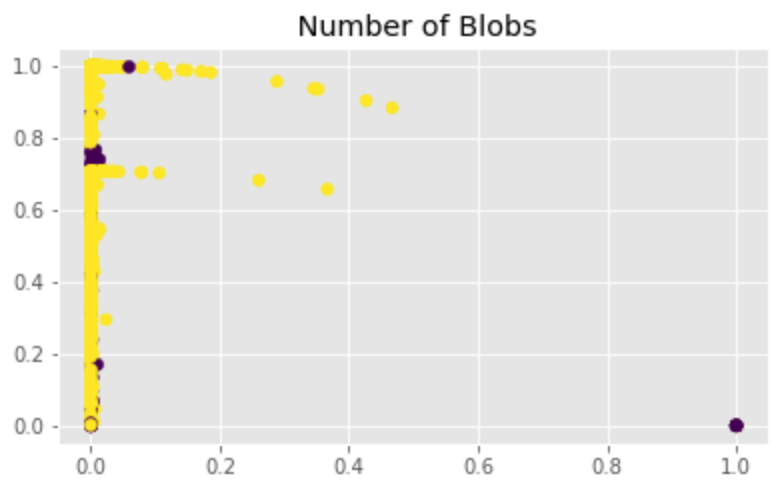
Out[46]:
```
['finalized_model.sav']
```

In [47]:
```
# load the model from disk
knn = joblib.load(filename)
```

In [48]:
```
# predicting labels for testing set
knn_predicted_test_labels=knn.predict(X_test)
```

In [49]:
```
from pylab import rcParams
#plt.figure(figsize=(12, 12))
rcParams['figure.figsize'] = 14, 8
plt.subplot(222)
plt.scatter(X_test[:, 0], X_test[:, 1], c=knn_predicted_test_labels)
plt.title(" Number of Blobs")
```

Out[49]:
```
Text(0.5, 1.0, ' Number of Blobs')
```

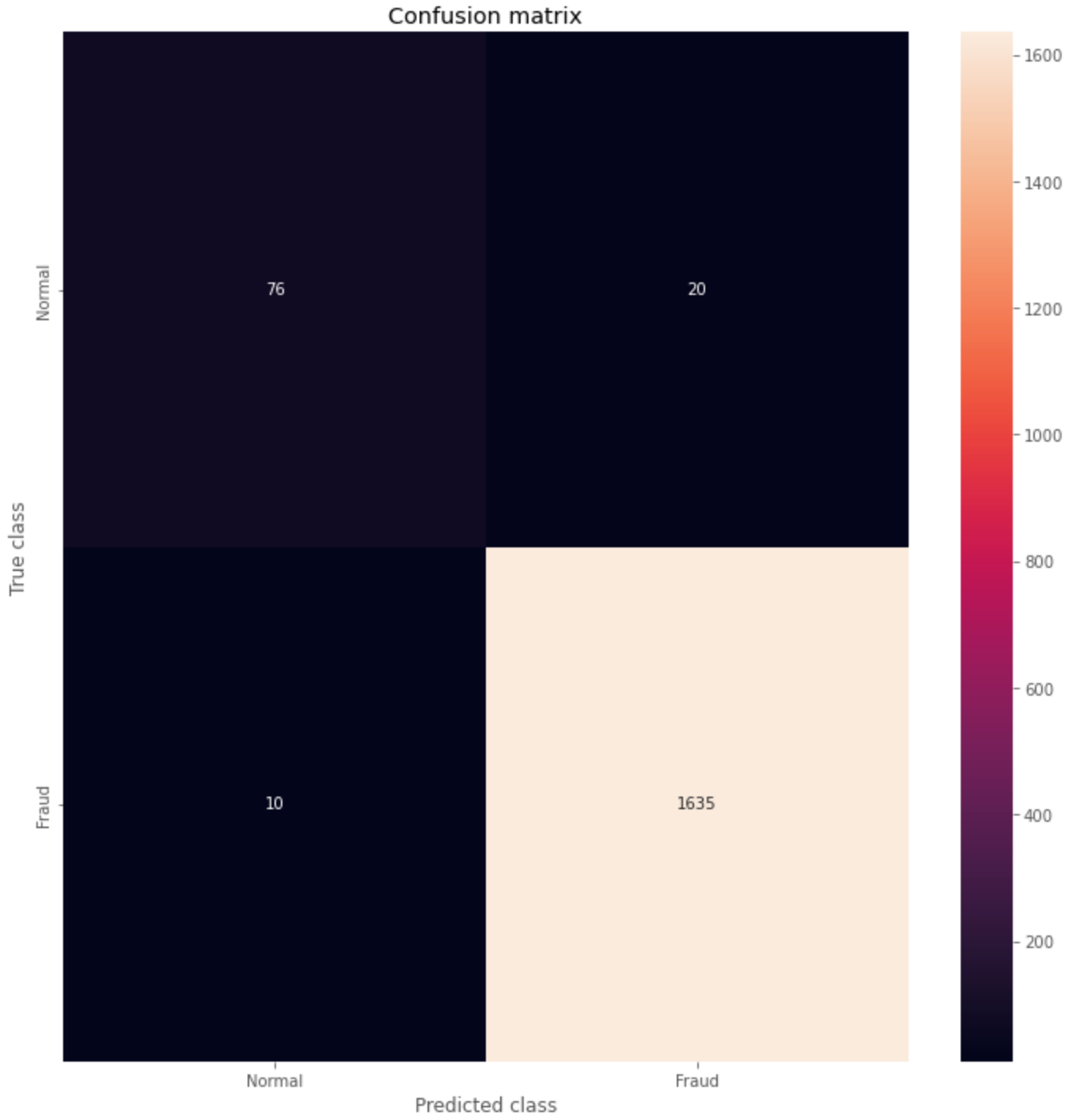


In [50]:
```
#scoring knn
knn_accuracy_score  = accuracy_score(y_test,knn_predicted_test_labels)
knn_precison_score  = precision_score(y_test,knn_predicted_test_labels)
knn_recall_score    = recall_score(y_test,knn_predicted_test_labels)
knn_f1_score        = f1_score(y_test,knn_predicted_test_labels)
knn_MCC             = matthews_corrcoef(y_test,knn_predicted_test_labels)
```

In [51]:
```
#printing
print("")
print("K-Nearest Neighbours")
print("Scores")
print("Accuracy -->",knn_accuracy_score)
print("Precison -->",knn_precison_score)
print("Recall -->",knn_recall_score)
print("F1 -->",knn_f1_score)
print("MCC -->",knn_MCC)
print(classification_report(y_test,knn_predicted_test_labels))
```

```
K-Nearest Neighbours
Scores
Accuracy --> 0.9827685238368754
Precison --> 0.9879154078549849
Recall --> 0.993920972644377
F1 --> 0.990909090909091
MCC --> 0.8274942496331413
              precision    recall  f1-score   support

           0       0.88      0.79      0.84        96
           1       0.99      0.99      0.99      1645

    accuracy                           0.98      1741
   macro avg       0.94      0.89      0.91      1741
weighted avg       0.98      0.98      0.98      1741
```

In [52]:
```
import seaborn as sns
LABELS = ['Normal', 'Fraud']
conf_matrix = confusion_matrix(y_test, knn_predicted_test_labels)
plt.figure(figsize=(12, 12))
sns.heatmap(conf_matrix, xticklabels=LABELS, yticklabels=LABELS, annot=True, fmt="d");
plt.title("Confusion matrix")
plt.ylabel('True class')
plt.xlabel('Predicted class')
plt.show()
```



Conclusion I tried without standardizing the data to get a better accuracy. But after I learnt this method and applied it, it gave a promising result. I am still doing experiments and still learning about data preprocessing techniques. I only used KNN algorithm for this dataset. Feel free to comment and give an Upvote if you find this kernel helpful.