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

In [2]:
```
data = pd.read_csv('creditcard.csv')
```

In [3]:
```
data.head()
```

Out[3]:

	Time	V1	V2	V3	V4	V5	V6	V7	V8	V9	...	V21	V22	V23	V24	V25	V26	V27	V28	Amount	Class
0	0.0	-1.359807	-0.072781	2.536347	1.378155	-0.338321	0.462388	0.239599	0.098698	0.363787	...	-0.018307	0.277838	-0.110474	0.066928	0.128539	-0.189115	0.133558	-0.021053	149.62	0
1	0.0	1.191857	0.266151	0.166480	0.448154	0.060018	-0.082361	-0.078803	0.085102	-0.255425	...	-0.225775	-0.638672	0.101288	-0.339846	0.167170	0.125895	-0.008983	0.014724	2.69	0
2	1.0	-1.358354	-1.340163	1.773209	0.379780	-0.503198	1.800499	0.791461	0.247676	-1.514654	...	0.247998	0.771679	0.909412	-0.689281	-0.327642	-0.139097	-0.055353	-0.059752	378.66	0
3	1.0	-0.966272	-0.185226	1.792993	-0.863291	-0.010309	1.247203	0.237609	0.377436	-1.387024	...	-0.108300	0.005274	-0.190321	-1.175575	0.647376	-0.221929	0.062723	0.061458	123.50	0
4	2.0	-1.158233	0.877737	1.548718	0.403034	-0.407193	0.095921	0.592941	-0.270533	0.817739	...	-0.009431	0.798278	-0.137458	0.141267	-0.206010	0.502292	0.219422	0.215153	69.99	0

5 rows × 31 columns

Describing the Data

In [4]:
```
#Print the shape of data
data.shape
```

Out[4]:
```
(284807, 31)
```

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

Out[5]:

	Time	V1	V2	V3	V4	V5	V6	V7	V8	V9	...	V21	V22	V23	V24	
count	284807.000000	2.848070e+05	2.848070e+05	2.848070e+05	2.848070e+05	2.848070e+05	2.848070e+05	2.848070e+05	2.848070e+05	2.848070e+05	...	2.848070e+05	2.848070e+05	2.848070e+05	2.848070e+05	2.848
mean	94813.859575	3.918649e-15	5.682686e-16	-8.761736e-15	2.811118e-15	-1.552103e-15	2.040130e-15	-1.698953e-15	-1.893285e-16	-3.147640e-15	...	1.473120e-16	8.042109e-16	5.282512e-16	4.456271e-15	1.426
std	47488.145955	1.958696e+00	1.651309e+00	1.516255e+00	1.415869e+00	1.380247e+00	1.332271e+00	1.237094e+00	1.194353e+00	1.098632e+00	...	7.345240e-01	7.257016e-01	6.244603e-01	6.056471e-01	5.212
min	0.000000	-5.640751e+01	-7.271573e+01	-4.832559e+01	-5.683171e+00	-1.137433e+02	-2.616051e+01	-4.355724e+01	-7.321672e+01	-1.343407e+01	...	-3.483038e+01	-1.093314e+01	-4.480774e+01	-2.836627e+00	-1.029
25%	54201.500000	-9.203734e-01	-5.985499e-01	-8.903648e-01	-8.486401e-01	-6.915971e-01	-7.682956e-01	-5.540759e-01	-2.086297e-01	-6.430976e-01	...	-2.283949e-01	-5.423504e-01	-1.618463e-01	-3.545861e-01	-3.171
50%	84692.000000	1.810880e-02	6.548556e-02	1.798463e-01	-1.984653e-02	-5.433583e-02	-2.741871e-01	4.010308e-02	2.235804e-02	-5.142873e-02	...	-2.945017e-02	6.781943e-03	-1.119293e-02	4.097606e-02	1.659
75%	139320.500000	1.315642e+00	8.037239e-01	1.027196e+00	7.433413e-01	6.119264e-01	3.985649e-01	5.704361e-01	3.273459e-01	5.971390e-01	...	1.863772e-01	5.285536e-01	1.476421e-01	4.395266e-01	3.507
max	172792.000000	2.454930e+00	2.205773e+01	9.382558e+00	1.687534e+01	3.480167e+01	7.330163e+01	1.205895e+02	2.000721e+01	1.559499e+01	...	2.720284e+01	1.050309e+01	2.252841e+01	4.584549e+00	7.519

8 rows × 31 columns

Imbalance in the data

Time to explain the data we are dealing with.

In [6]:
```
# Determine number of fraud cases in datasheet
fraud = data[data['Class'] == 1]
valid = data[data['Class'] == 0]
outlierfraction = len(fraud)/float(len(valid))
print(outlierfraction)
print('Fraud cases:{}'.format(len(data[data['Class'] ==1])))
print('Valid transactions:{}'.format(len(data[data['Class'] ==0])))
```

```
0.0017304750013189597
fraud cases:492
valid transactions:284315
```

Print the amount details for Fraudulent Transaction

In [7]:
```
print("Amount details of the fraudulent transaction")
fraud.Amount.describe()
```

Out[7]:
```
Amount details of the fraudulent transaction
count     492.000000
mean      122.211321
std       256.683288
min         0.000000
25%         1.000000
50%         9.250000
75%       105.890000
max      2125.870000
Name: Amount, dtype: float64
```

Print the amount details for normal transaction

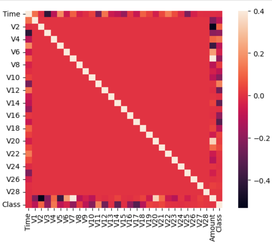
In [8]:
```
print("Amount details of the valid transaction")
valid.Amount.describe()
```

Out[8]:
```
Amount details of the valid transaction
count    284315.000000
mean         88.291022
std         250.105092
min           0.000000
25%           5.650000
50%          22.000000
75%          77.050000
max       25691.160000
Name: Amount, dtype: float64
```

Ploting the correlation Matrix

The correlationn matrix graphically gives an idea of how feature correlate with each other and can helps us predict what aree the ferture that are most relevant for the prediction.

In [9]:
```
#Correlation matrix
cormmet = data.corr()
fig = plt.figure(figsize = (8 ,5))
sns.heatmap(cormmet, vmax = .4, square = True)
plt.show()
```



Seprating the X and the Y Values

Dividing the data into inputs parameters and outputs values format

In [10]:
```
#dividing thew X and the Y from the datasets
X = data.drop(['Class'], axis = 1)
Y = data['Class']
print(X.shape)
print(Y.shape)
#getting just the values for the sake of processing
#(its a numpy array with noo columns)
xData = X.values
yData = Y.values
```

```
(284807, 30)
(284807,)
```

Training and Testing Data Bifurcation

we will be dividing the dataset into two main groups. One for training the model and the other for testing our trained models performances

In [11]:
```
# Using scikit_learn to split data into training and testing sets
from sklearn.model_selection import train_test_split
xTrain, xTest, yTrain, yTest = train_test_split(xData, yData, test_size = 0.2, random_state = 42)
```

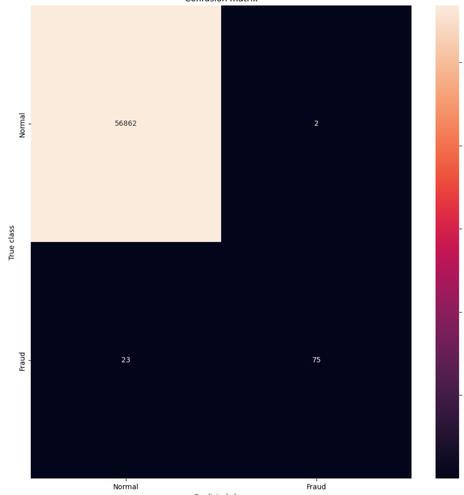
Building a Random Forest Model Using scikit learn

In [12]:
```
# Building the random forest classifire (RANDOM FOREST)
from sklearn.ensemble import RandomForestClassifier
#Random forest model creation

rfc = RandomForestClassifier()
rfc.fit(xTrain, yTrain)
#Prediction
yPred = rfc.predict(xTest)
```

In [13]:
```
# Evaluating the classifier
# printing every score of the classifier
# scoring in anything

from sklearn.metrics import classification_report, accuracy_score
from sklearn.metrics import precision_score, recall_score
from sklearn.metrics import f1_score, matthews_corrcoef
from sklearn.metrics import confusion_matrix


n_outliers = len(fraud)
n_errors = (yPred != yTest ).sum()
print("The model used in Random Forest Classifier")

acc = accuracy_score(yTest, yPred)
print("The accuracy is {}".format(acc))

prec = precision_score(yTest, yPred)
print("The precision is {}".format(prec))

rec = recall_score(yTest, yPred)
print("The recall is {}".format(rec))

f1 = f1_score(yTest, yPred)
print("The f1-score is {}".format(f1))

MCC = matthews_corrcoef(yTest, yPred)
print("The matthews correlation coefficient is {}".format(MCC))
```

```
The model used in Random Forest Classifier
The accuracy is 0.9995611109160493
The precision is 0.9740259740259574
The recall is 0.7653061224489796
The f1-score is 0.8571428571428571
The matthews correlation coefficient is 0.8631826952924256
```

In [14]:
```
# Printing the confusion matrix

LABELS = ['Normal', 'Fraud']
conf_matrix = confusion_matrix(yTest, yPred)
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()
```



In [ ]:
```

```