In [85]:
```
#Importing the libraries
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import tensorflow as tf
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix, accuracy_score, precision_score, recall_score, f1_score, mean_squared_error
```

In [3]:
```
#Importing the dataset
data = pd.read_csv('creditcard.csv')
```

In [4]:
```
#Preprocessing Step: Checking for NULL values
print("Any null values in the dataset ", data.isnull().values.any())
print("Number of Unique Labels ", len(data['Class'].unique()))
print("Label values ", data.Class.unique())
print("Total Count of Normal & Fraud Transactions ", pd.value_counts(data['Class'], sort = True))
```

```
Any null values in the dataset  False
Number of Unique Labels  2
Label values  [0 1]
Total Count of Normal & Fraud Transactions  Class
0    284315
1       492
Name: count, dtype: int64
```

In [5]:
```
countofclass = pd.value_counts(data['Class'], sort = True)
countofclass.plot(kind = 'bar')
plt.title("Frequencies for Number of Classes")
plt.xlabel("No. of Classes")
plt.ylabel("Frequencies")
```

Out[5]:
```
Text(0, 0.5, 'Frequencies')
```

In [6]:
```
data.values
```

Out[6]:
```
array([[ 0.00000000e+00, -1.35980713e+00, -7.27811733e-02, ...,
        -2.10530535e-02,  1.49620000e+02,  0.00000000e+00],
       [ 0.00000000e+00,  1.19185711e+00,  2.66150712e-01, ...,
         1.47241692e-02,  2.69000000e+00,  0.00000000e+00],
       [ 1.00000000e+00, -1.35835406e+00, -1.34016307e+00, ...,
        -5.97514404e-02,  3.78660000e+02,  0.00000000e+00],
       ...,
       [ 1.72788000e+05,  1.91956501e+00, -3.01525386e-01, ...,
         2.65608286e-02,  6.78800000e+01,  0.00000000e+00],
       [ 1.72788000e+05, -2.40440050e-01,  5.39482513e-01, ...,
         1.04532821e-01,  1.00000000e+01,  0.00000000e+00],
       [ 1.72792000e+05, -5.33412522e-01, -1.89733337e-01, ...,
         1.36489143e-02,  2.17000000e+02,  0.00000000e+00]])
```

In [36]:
```
df = data.values
target = df[:, -1]
featureval = df[:, 0:-1]

traindata, testdata, trainlabel, testlabel = train_test_split(data, target, test_size = 0.3, random_state = 10)
```

In [ ]:
```
train_data =
```

In [47]:
```
min_val = tf.reduce_min(traindata)
max_val = tf.reduce_max(traindata)

# Assuming min_val and max_val are TensorFlow tensors
traindata = tf.divide(tf.subtract(traindata, min_val), tf.subtract(max_val, min_val))
testdata = tf.divide(tf.subtract(testdata, min_val), tf.subtract(max_val, min_val))

# traindata = (traindata - min_val) / (max_val - min_val)
# testdata = (testdata - min_val) / (max_val - min_val)
traindata = tf.cast(traindata, tf.float32)
testdata = tf.cast(testdata, tf.float32)
```

In [46]:
```
traindata
```

Out[46]:
```
<tf.Tensor: shape=(199364, 31), dtype=float64, numpy=
array([[7.18112313e-01, 6.55693795e-04, 6.57522760e-04, ...,
        6.58698790e-04, 1.00253065e-03, 6.57834173e-04],
       [3.24545282e-01, 6.65288969e-04, 6.65295884e-04, ...,
        6.57852579e-04, 8.47243729e-04, 6.57834173e-04],
       [4.92428659e-01, 6.54059302e-04, 6.60169396e-04, ...,
        6.58401275e-04, 7.73504131e-04, 6.57834173e-04],
       ...,
       [4.03235555e-01, 6.49146535e-04, 6.62575795e-04, ...,
        6.58114098e-04, 7.41463553e-04, 6.57834173e-04],
       [3.75606695e-01, 6.49927494e-04, 6.66104439e-04, ...,
        6.58633543e-04, 8.32842819e-04, 6.57834173e-04],
       [8.61849586e-01, 6.69824622e-04, 6.57296994e-04, ...,
        6.57471499e-04, 6.65294886e-04, 6.57834173e-04]])>
```

In [21]:
```
print("Maximum val ", maxval)
print("Minimum value ",minval)
```

```
Maximum val  tf.Tensor(172792.0, shape=(), dtype=float64)
Minimum value  tf.Tensor(-113.74330671146, shape=(), dtype=float64)
```

In [57]:
```
trainlabel = trainlabel.astype(bool)
testlabel = testlabel.astype(bool)

normaltraindata = traindata[~trainlabel]
normaltestdata = testdata[~testlabel]

frauttraindata = traindata[trainlabel]
frauttestdata = testdata[testlabel]

print("Normal transactions in Training data ", len(normaltraindata))
print("Normal transactions in Testing data ", len(normaltestdata))
print("Fraud transactions in Training data ", len(frauttraindata))
print("Fraud transactions in Testing data ", len(frauttestdata))
```

```
Normal transactions in Training data  199013
Normal transactions in Testing data  85302
Fraud transactions in Training data  351
Fraud transactions in Testing data  141
```

In [58]:
```
epochs = 50
batchsize = 64
inputdimension = normaltraindata.shape[1]
encoded_dimension = 14
hiddenlayer1 = int(encoded_dimension/2)
hiddenlayer2 = 4
learningrate = 1e-7
```

In [61]:
```
#Autoencoder Creation
#1. Input layer
input_layer = tf.keras.layers.Input(shape = (inputdimension, ))

#2.Encoder
encoder = tf.keras.layers.Dense(encoded_dimension, activation = 'tanh',
                                activity_regularizer = tf.keras.regularizers.l2(learningrate))(input_layer)
encoder = tf.keras.layers.Dropout(0.2)(encoder)
encoder = tf.keras.layers.Dense(hiddenlayer1, activation = 'relu')(encoder)
encoder = tf.keras.layers.Dense(hiddenlayer2, activation = tf.nn.leaky_relu)(encoder)

#3.Decoder
decoder = tf.keras.layers.Dense(hiddenlayer1, activation = 'relu')(encoder)
decoder = tf.keras.layers.Dropout(0.2)(decoder)
decoder = tf.keras.layers.Dense(encoded_dimension, activation = 'relu')(decoder)
decoder = tf.keras.layers.Dense(inputdimension, activation = 'tanh')(decoder)

#4. Autoencoder
autoencoder = tf.keras.Model(inputs = input_layer, outputs = decoder)
autoencoder.summary()
```

```
Model: "model"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_3 (InputLayer)        [(None, 31)]              0

 dense_7 (Dense)             (None, 14)                448

 dropout_2 (Dropout)         (None, 14)                0

 dense_8 (Dense)             (None, 7)                 105

 dense_9 (Dense)             (None, 4)                 32

 dense_10 (Dense)            (None, 7)                 35

 dropout_3 (Dropout)         (None, 7)                 0

 dense_11 (Dense)            (None, 14)                112

 dense_12 (Dense)            (None, 31)                465

=================================================================
Total params: 1197 (4.68 KB)
Trainable params: 1197 (4.68 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

In [64]:
```
cp = tf.keras.callbacks.ModelCheckpoint(filepath = "p4", mode = "min", monitor = "val_loss", verbose = 2, bestmodel = True)
```

In [67]:
```
earlystop = tf.keras.callbacks.EarlyStopping(monitor = "val_loss",
                                             patience = 10,
                                             restore_best_weights = True)
```

In [69]:
```
autoencoder.compile(loss = "mean_squared_error", metrics = "accuracy", optimizer = "adam")
```

In [70]:
```
history = autoencoder.fit(normaltraindata, normaltraindata,
                          validation_data = (testdata, testdata),
                          epochs = epochs,
                          batch_size = batchsize,
                          callbacks = [cp, earlystop]).history
```

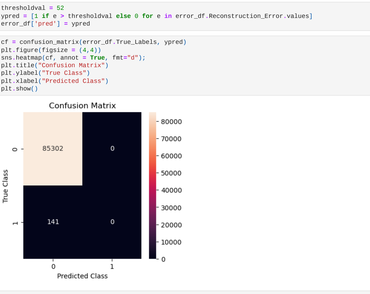
```
Epoch 1/50
3099/3110 [============================>.] - ETA: 0s - loss: 3.6846e-04 - accuracy: 0.9969
Epoch 1: saving model to p4
INFO:tensorflow:Assets written to: p4\assets
INFO:tensorflow:Assets written to: p4\assets
3110/3110 [==============================] - 23s 7ms/step - loss: 3.6743e-04 - accuracy: 0.9969 - val_loss: 6.3219e-05 - val_accuracy: 0.9996
Epoch 2/50
3107/3110 [============================>.] - ETA: 0s - loss: 5.7064e-05 - accuracy: 0.9985
Epoch 2: saving model to p4
INFO:tensorflow:Assets written to: p4\assets
INFO:tensorflow:Assets written to: p4\assets
3110/3110 [==============================] - 22s 7ms/step - loss: 5.7065e-05 - accuracy: 0.9985 - val_loss: 1.0667e-04 - val_accuracy: 0.9996
Epoch 3/50
3099/3110 [============================>.] - ETA: 0s - loss: 3.5185e-05 - accuracy: 0.9996
Epoch 3: saving model to p4
INFO:tensorflow:Assets written to: p4\assets
INFO:tensorflow:Assets written to: p4\assets
3110/3110 [==============================] - 20s 6ms/step - loss: 3.5220e-05 - accuracy: 0.9996 - val_loss: 1.4140e-04 - val_accuracy: 0.9996
Epoch 4/50
3107/3110 [============================>.] - ETA: 0s - loss: 2.4512e-05 - accuracy: 0.9997
Epoch 4: saving model to p4
INFO:tensorflow:Assets written to: p4\assets
INFO:tensorflow:Assets written to: p4\assets
3110/3110 [==============================] - 19s 6ms/step - loss: 2.4509e-05 - accuracy: 0.9997 - val_loss: 1.3969e-04 - val_accuracy: 0.9996
Epoch 5/50
3105/3110 [============================>.] - ETA: 0s - loss: 2.0239e-05 - accuracy: 0.9996
Epoch 5: saving model to p4
INFO:tensorflow:Assets written to: p4\assets
INFO:tensorflow:Assets written to: p4\assets
3110/3110 [==============================] - 20s 6ms/step - loss: 2.0226e-05 - accuracy: 0.9996 - val_loss: 2.6539e-04 - val_accuracy: 0.9996
Epoch 6/50
3108/3110 [============================>.] - ETA: 0s - loss: 1.9006e-05 - accuracy: 0.9996
Epoch 6: saving model to p4
INFO:tensorflow:Assets written to: p4\assets
INFO:tensorflow:Assets written to: p4\assets
3110/3110 [==============================] - 21s 7ms/step - loss: 1.9003e-05 - accuracy: 0.9996 - val_loss: 2.6354e-04 - val_accuracy: 0.9996
Epoch 7/50
3103/3110 [============================>.] - ETA: 0s - loss: 1.8045e-05 - accuracy: 0.9995
Epoch 7: saving model to p4
INFO:tensorflow:Assets written to: p4\assets
INFO:tensorflow:Assets written to: p4\assets
3110/3110 [==============================] - 23s 8ms/step - loss: 1.8037e-05 - accuracy: 0.9995 - val_loss: 2.4839e-04 - val_accuracy: 0.9996
Epoch 8/50
3106/3110 [============================>.] - ETA: 0s - loss: 1.6806e-05 - accuracy: 0.9995
Epoch 8: saving model to p4
INFO:tensorflow:Assets written to: p4\assets
INFO:tensorflow:Assets written to: p4\assets
3110/3110 [==============================] - 24s 8ms/step - loss: 1.6804e-05 - accuracy: 0.9995 - val_loss: 2.8880e-04 - val_accuracy: 0.9996
Epoch 9/50
3110/3110 [==============================] - ETA: 0s - loss: 1.6312e-05 - accuracy: 0.9994
Epoch 9: saving model to p4
INFO:tensorflow:Assets written to: p4\assets
INFO:tensorflow:Assets written to: p4\assets
3110/3110 [==============================] - 24s 8ms/step - loss: 1.6312e-05 - accuracy: 0.9994 - val_loss: 2.7952e-04 - val_accuracy: 0.9996
Epoch 10/50
3101/3110 [============================>.] - ETA: 0s - loss: 1.7076e-05 - accuracy: 0.9993
Epoch 10: saving model to p4
INFO:tensorflow:Assets written to: p4\assets
INFO:tensorflow:Assets written to: p4\assets
3110/3110 [==============================] - 25s 8ms/step - loss: 1.7055e-05 - accuracy: 0.9993 - val_loss: 2.5668e-04 - val_accuracy: 0.9996
Epoch 11/50
3110/3110 [==============================] - ETA: 0s - loss: 1.7195e-05 - accuracy: 0.9994
Epoch 11: saving model to p4
INFO:tensorflow:Assets written to: p4\assets
INFO:tensorflow:Assets written to: p4\assets
3110/3110 [==============================] - 34s 11ms/step - loss: 1.7195e-05 - accuracy: 0.9994 - val_loss: 2.9389e-04 - val_accuracy: 0.9996
```

In [83]:
```
predictions = autoencoder.predict(testdata)
mse = mean_squared_error(testdata, predictions)
print("Mean Squared Error ", mse)
error_df = pd.DataFrame({'Reconstruction_Error': mse, 'True_Labels': testlabel})
```

```
2671/2671 [==============================] - 7s 3ms/step
Mean Squared Error  0.31963Be-05
```

In [82]:
```
thresholdval = 52
ypred = [1 if e > thresholdval else 0 for e in error_df.Reconstruction_Error.values]
error_df['pred'] = ypred
```

In [86]:
```
cf = confusion_matrix(error_df.True_Labels, ypred)
plt.figure(figsize = (4,4))
sns.heatmap(cf, annot = True, fmt="d");
plt.title("Confusion Matrix")
plt.ylabel("True Class")
plt.xlabel("Predicted Class")
plt.show()
```



In [ ]:
```
```