import pandas as pd import numpy as np import tensorflow as tf import matplotlib.pyplot as plt from sklearn.model\_selection import train\_test\_split

**import** pandas **as** pd

**import** numpy **as** np

**import** tensorflow **as** tf

**import** matplotlib.pyplot **as** plt

**from** sklearn.model\_selection **import** train\_test\_split

C:\Users\LENOVO\AppData\Roaming\Python\Python39\site-packages\scipy\\_\_init\_\_.py:146: UserWarning: A NumPy version >=1.17.3 and <1.25.0 is required for this version of SciPy (detected version 1.26.1

warnings.warn(f"A NumPy version >={np\_minversion} and <{np\_maxversion}"

In [2]:

**from** sklearn.preprocessing **import** StandardScaler

**from** sklearn.metrics **import** confusion\_matrix, recall\_score, accuracy\_score, precision\_score

RANDOM\_SEED **=** 2021

TEST\_PCT **=** 0.3

LABELS **=** ["Normal","Fraud"]

In [3]:

dataset **=** pd.read\_csv("creditcard.csv")

print(list(dataset.columns))

dataset.describe()

['Time', 'V1', 'V2', 'V3', 'V4', 'V5', 'V6', 'V7', 'V8', 'V9', 'V10', 'V11', 'V12', 'V13', 'V14', 'V15', 'V16', 'V17', 'V18', 'V19', 'V20', 'V21', 'V22', 'V23', 'V24', 'V25', 'V26', 'V27', 'V28', 'Amount', 'Class']

Out[3]:

|  | **Time** | **V1** | **V2** | **V3** | **V4** | **V5** | **V6** | **V7** | **V8** | **V9** | **...** | **V21** | **V22** | **V23** | **V24** | **V25** | **V26** | **V27** | **V28** | **Amount** | **Class** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **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 | 2.848070e+05 | 2.848070e+05 | 2.848070e+05 | 2.848070e+05 | 284807.000000 | 284807.000000 |
| **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 | 1.426896e-15 | 1.701640e-15 | -3.662252e-16 | -1.217809e-16 | 88.349619 | 0.001727 |
| **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 | 5.212781e-01 | 4.822270e-01 | 4.036325e-01 | 3.300833e-01 | 250.120109 | 0.041527 |
| **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 | -1.029540e+01 | -2.604551e+00 | -2.256568e+01 | -1.543008e+01 | 0.000000 | 0.000000 |
| **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 | -3.171451e-01 | -3.269839e-01 | -7.083953e-02 | -5.295979e-02 | 5.600000 | 0.000000 |
| **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 | 1.659350e-02 | -5.213911e-02 | 1.342146e-03 | 1.124383e-02 | 22.000000 | 0.000000 |
| **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 | 3.507156e-01 | 2.409522e-01 | 9.104512e-02 | 7.827995e-02 | 77.165000 | 0.000000 |
| **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 | 7.519589e+00 | 3.517346e+00 | 3.161220e+01 | 3.384781e+01 | 25691.160000 | 1.000000 |

8 rows × 31 columns

In [4]:

*#check for any nullvalues*

print("Any nulls in the dataset ",dataset.isnull().values.any() )

print('-------')

print("No. of unique labels ", len(dataset['Class'].unique()))

print("Label values ",dataset.Class.unique())

*#0 is for normal credit card transaction*

*#1 is for fraudulent credit card transaction*

print('-------')

print("Break down of the Normal and Fraud Transactions")

print(pd.value\_counts(dataset['Class'], sort **=** **True**) )

Any nulls in the dataset False

-------

No. of unique labels 2

Label values [0 1]

-------

Break down of the Normal and Fraud Transactions

0 284315

1 492

Name: Class, dtype: int64

In [5]:

*#Visualizing the imbalanced dataset*

count\_classes **=** pd.value\_counts(dataset['Class'], sort **=** **True**)

count\_classes.plot(kind **=** 'bar', rot**=**0)

plt.xticks(range(len(dataset['Class'].unique())), dataset.Class.unique())

plt.title("Frequency by observation number")

plt.xlabel("Class")

plt.ylabel("Number of Observations");



In [6]:

*# Save the normal and fradulent transactions in separate dataframe*

normal\_dataset **=** dataset[dataset.Class **==** 0]

fraud\_dataset **=** dataset[dataset.Class **==** 1]

*#Visualize transaction amounts for normal and fraudulent transactions*

bins **=** np.linspace(200, 2500, 100) *# Return evenly spaced numbers over a specified interval.*

plt.hist(normal\_dataset.Amount, bins**=**bins, alpha**=**1, density**=True**, label**=**'Normal')

plt.hist(fraud\_dataset.Amount, bins**=**bins, alpha**=**0.5, density**=True**, label**=**'Fraud')

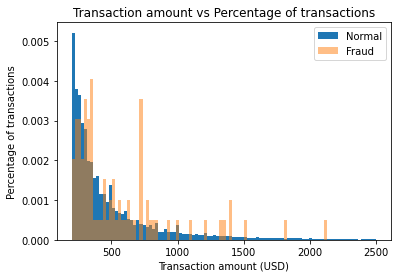
plt.legend(loc**=**'upper right')

plt.title("Transaction amount vs Percentage of transactions")

plt.xlabel("Transaction amount (USD)")

plt.ylabel("Percentage of transactions");

plt.show()



In [7]:

'''Time and Amount are the columns that are not scaled, so applying StandardScaler to only Amount and Time columns.

Normalizing the values between 0 and 1 did not work great for the dataset.'''

Out[7]:

'Time and Amount are the columns that are not scaled, so applying StandardScaler to only Amount and Time columns. \nNormalizing the values between 0 and 1 did not work great for the dataset.'

In [8]:

sc**=**StandardScaler()

dataset['Time'] **=** sc.fit\_transform(dataset['Time'].values.reshape(**-**1, 1))

dataset['Amount'] **=** sc.fit\_transform(dataset['Amount'].values.reshape(**-**1, 1))

In [9]:

dataset['Amount']

Out[9]:

0 0.244964

1 -0.342475

2 1.160686

3 0.140534

4 -0.073403

...

284802 -0.350151

284803 -0.254117

284804 -0.081839

284805 -0.313249

284806 0.514355

Name: Amount, Length: 284807, dtype: float64

In [10]:

train\_x, test\_x **=** train\_test\_split(dataset, test\_size**=**TEST\_PCT, random\_state**=**RANDOM\_SEED)

train\_x **=** train\_x[train\_x.Class **==** 0] *# where normal transactions*

train\_x **=** train\_x.drop(['Class'], axis**=**1) *# drop the class column*

​

​

test\_y **=** test\_x['Class'] *# save the class column for the test set*

test\_x **=** test\_x.drop(['Class'], axis**=**1) *# drop the class column*

​

train\_x **=** train\_x.values *# transform to ndarray*

test\_x **=** test\_x.values *# transform to ndarray*

**Autoencoder Layer Structure and Parameters**

Autoencoder has symmetric encoding and decoding layers that are "dense". We are reducing the input into some form of simplified encoding and then expanding it again. The input and output dimension is the feature space (e.g. 30 columns), so the encoding layer should be smaller by an amount that expect to represent some feature. In this case, I am encoding 30 columns into 14 dimensions so I am expecting high-level features to be represented by roughly two columns (30/14 = 2.1). Of those high-level features, I am expecting them to map to roughly seven hidden/latent features in the data.

Additionally, the epochs, batch size, learning rate, learning policy, and activation functions were all set to values empirically good values.

In [11]:

nb\_epoch **=** 50

batch\_size **=** 64

input\_dim **=** train\_x.shape[1] *#num of columns, 30*

encoding\_dim **=** 14

hidden\_dim\_1 **=** int(encoding\_dim **/** 2) *#*

hidden\_dim\_2**=**4

learning\_rate **=** 1e-7

In [12]:

*#input Layer*

input\_layer **=** tf.keras.layers.Input(shape**=**(input\_dim, ))

​

*#Encoder*

encoder **=** tf.keras.layers.Dense(encoding\_dim, activation**=**"tanh",

activity\_regularizer**=**tf.keras.regularizers.l2(learning\_rate))(input\_layer)

*# encoder=tf.keras.layers.Dropout(0.2)(encoder)*

encoder **=** tf.keras.layers.Dense(hidden\_dim\_1, activation**=**'relu')(encoder)

encoder **=** tf.keras.layers.Dense(hidden\_dim\_2, activation**=**tf.nn.leaky\_relu)(encoder)

​

*# Decoder*

decoder **=** tf.keras.layers.Dense(hidden\_dim\_1, activation**=**'relu')(encoder)

*# decoder=tf.keras.layers.Dropout(0.2)(decoder)*

decoder **=** tf.keras.layers.Dense(encoding\_dim, activation**=**'relu')(decoder)

decoder **=** tf.keras.layers.Dense(input\_dim, activation**=**'tanh')(decoder)

​

*#Autoencoder*

autoencoder **=** tf.keras.Model(inputs**=**input\_layer, outputs**=**decoder)

autoencoder.summary()

Model: "model"

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Layer (type) Output Shape Param #

=================================================================

input\_1 (InputLayer) [(None, 30)] 0

dense (Dense) (None, 14) 434

dense\_1 (Dense) (None, 7) 105

dense\_2 (Dense) (None, 4) 32

dense\_3 (Dense) (None, 7) 35

dense\_4 (Dense) (None, 14) 112

dense\_5 (Dense) (None, 30) 450

=================================================================

Total params: 1168 (4.56 KB)

Trainable params: 1168 (4.56 KB)

Non-trainable params: 0 (0.00 Byte)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

In [13]:

"""Define the callbacks for checkpoints and early stopping"""

​

cp **=** tf.keras.callbacks.ModelCheckpoint(filepath**=**"autoencoder\_fraud.h5",

mode**=**'min', monitor**=**'val\_loss', verbose**=**2, save\_best\_only**=True**)

*# define our early stopping*

early\_stop **=** tf.keras.callbacks.EarlyStopping(

monitor**=**'val\_loss',

min\_delta**=**0.0001,

patience**=**10,

verbose**=**1,

mode**=**'min',

restore\_best\_weights**=True**)

In [14]:

*#Compile the Autoencoder*

​

autoencoder.compile(metrics**=**['accuracy'],

loss**=**'mean\_squared\_error',

optimizer**=**'adam')

In [15]:

*#Train the Autoencoder*

​

history **=** autoencoder.fit(train\_x, train\_x,

epochs**=**nb\_epoch,

batch\_size**=**batch\_size,

shuffle**=True**,

validation\_data**=**(test\_x, test\_x),

verbose**=**1,

callbacks**=**[cp, early\_stop]

).history

​

Epoch 1/50

3105/3110 [============================>.] - ETA: 0s - loss: 0.7846 - accuracy: 0.3701

Epoch 1: val\_loss improved from inf to 0.75476, saving model to autoencoder\_fraud.h5

3110/3110 [==============================] - 6s 2ms/step - loss: 0.7845 - accuracy: 0.3702 - val\_loss: 0.7548 - val\_accuracy: 0.4070

Epoch 2/50

90/3110 [..............................] - ETA: 3s - loss: 0.7314 - accuracy: 0.3932

C:\Users\LENOVO\AppData\Roaming\Python\Python39\site-packages\keras\src\engine\training.py:3000: UserWarning: You are saving your model as an HDF5 file via `model.save()`. This file format is considered legacy. We recommend using instead the native Keras format, e.g. `model.save('my\_model.keras')`.

saving\_api.save\_model(

3093/3110 [============================>.] - ETA: 0s - loss: 0.7082 - accuracy: 0.4106

Epoch 2: val\_loss improved from 0.75476 to 0.72966, saving model to autoencoder\_fraud.h5

3110/3110 [==============================] - 5s 2ms/step - loss: 0.7076 - accuracy: 0.4105 - val\_loss: 0.7297 - val\_accuracy: 0.4156

Epoch 3/50

3091/3110 [============================>.] - ETA: 0s - loss: 0.6912 - accuracy: 0.4201

Epoch 3: val\_loss improved from 0.72966 to 0.71782, saving model to autoencoder\_fraud.h5

3110/3110 [==============================] - 5s 2ms/step - loss: 0.6909 - accuracy: 0.4200 - val\_loss: 0.7178 - val\_accuracy: 0.4222

Epoch 4/50

3068/3110 [============================>.] - ETA: 0s - loss: 0.6798 - accuracy: 0.4275

Epoch 4: val\_loss improved from 0.71782 to 0.70770, saving model to autoencoder\_fraud.h5

3110/3110 [==============================] - 5s 2ms/step - loss: 0.6801 - accuracy: 0.4273 - val\_loss: 0.7077 - val\_accuracy: 0.4316

Epoch 5/50

3098/3110 [============================>.] - ETA: 0s - loss: 0.6707 - accuracy: 0.4316

Epoch 5: val\_loss improved from 0.70770 to 0.70018, saving model to autoencoder\_fraud.h5

3110/3110 [==============================] - 5s 2ms/step - loss: 0.6706 - accuracy: 0.4316 - val\_loss: 0.7002 - val\_accuracy: 0.4318

Epoch 6/50

3105/3110 [============================>.] - ETA: 0s - loss: 0.6643 - accuracy: 0.4340

Epoch 6: val\_loss improved from 0.70018 to 0.69467, saving model to autoencoder\_fraud.h5

3110/3110 [==============================] - 8s 2ms/step - loss: 0.6644 - accuracy: 0.4340 - val\_loss: 0.6947 - val\_accuracy: 0.4352

Epoch 7/50

3091/3110 [============================>.] - ETA: 0s - loss: 0.6611 - accuracy: 0.4315

Epoch 7: val\_loss improved from 0.69467 to 0.69111, saving model to autoencoder\_fraud.h5

3110/3110 [==============================] - 8s 3ms/step - loss: 0.6604 - accuracy: 0.4315 - val\_loss: 0.6911 - val\_accuracy: 0.4300

Epoch 8/50

3102/3110 [============================>.] - ETA: 0s - loss: 0.6577 - accuracy: 0.4303

Epoch 8: val\_loss improved from 0.69111 to 0.68898, saving model to autoencoder\_fraud.h5

3110/3110 [==============================] - 8s 3ms/step - loss: 0.6573 - accuracy: 0.4304 - val\_loss: 0.6890 - val\_accuracy: 0.4268

Epoch 9/50

3091/3110 [============================>.] - ETA: 0s - loss: 0.6549 - accuracy: 0.4303

Epoch 9: val\_loss improved from 0.68898 to 0.68669, saving model to autoencoder\_fraud.h5

3110/3110 [==============================] - 5s 2ms/step - loss: 0.6549 - accuracy: 0.4302 - val\_loss: 0.6867 - val\_accuracy: 0.4284

Epoch 10/50

3104/3110 [============================>.] - ETA: 0s - loss: 0.6531 - accuracy: 0.4310

Epoch 10: val\_loss improved from 0.68669 to 0.68515, saving model to autoencoder\_fraud.h5

3110/3110 [==============================] - 5s 2ms/step - loss: 0.6529 - accuracy: 0.4310 - val\_loss: 0.6852 - val\_accuracy: 0.4325

Epoch 11/50

3110/3110 [==============================] - ETA: 0s - loss: 0.6516 - accuracy: 0.4326

Epoch 11: val\_loss improved from 0.68515 to 0.68225, saving model to autoencoder\_fraud.h5

3110/3110 [==============================] - 6s 2ms/step - loss: 0.6516 - accuracy: 0.4326 - val\_loss: 0.6823 - val\_accuracy: 0.4327

Epoch 12/50

3102/3110 [============================>.] - ETA: 0s - loss: 0.6500 - accuracy: 0.4325

Epoch 12: val\_loss improved from 0.68225 to 0.68099, saving model to autoencoder\_fraud.h5

3110/3110 [==============================] - 8s 3ms/step - loss: 0.6498 - accuracy: 0.4325 - val\_loss: 0.6810 - val\_accuracy: 0.4312

Epoch 13/50

3097/3110 [============================>.] - ETA: 0s - loss: 0.6488 - accuracy: 0.4341

Epoch 13: val\_loss improved from 0.68099 to 0.67866, saving model to autoencoder\_fraud.h5

3110/3110 [==============================] - 8s 2ms/step - loss: 0.6484 - accuracy: 0.4342 - val\_loss: 0.6787 - val\_accuracy: 0.4369

Epoch 14/50

3102/3110 [============================>.] - ETA: 0s - loss: 0.6473 - accuracy: 0.4364

Epoch 14: val\_loss improved from 0.67866 to 0.67797, saving model to autoencoder\_fraud.h5

3110/3110 [==============================] - 8s 3ms/step - loss: 0.6470 - accuracy: 0.4364 - val\_loss: 0.6780 - val\_accuracy: 0.4357

Epoch 15/50

3105/3110 [============================>.] - ETA: 0s - loss: 0.6454 - accuracy: 0.4380

Epoch 15: val\_loss improved from 0.67797 to 0.67683, saving model to autoencoder\_fraud.h5

3110/3110 [==============================] - 8s 3ms/step - loss: 0.6455 - accuracy: 0.4380 - val\_loss: 0.6768 - val\_accuracy: 0.4374

Epoch 16/50

3089/3110 [============================>.] - ETA: 0s - loss: 0.6442 - accuracy: 0.4398

Epoch 16: val\_loss improved from 0.67683 to 0.67487, saving model to autoencoder\_fraud.h5

3110/3110 [==============================] - 8s 2ms/step - loss: 0.6441 - accuracy: 0.4398 - val\_loss: 0.6749 - val\_accuracy: 0.4349

Epoch 17/50

3106/3110 [============================>.] - ETA: 0s - loss: 0.6426 - accuracy: 0.4412

Epoch 17: val\_loss improved from 0.67487 to 0.67270, saving model to autoencoder\_fraud.h5

3110/3110 [==============================] - 8s 2ms/step - loss: 0.6425 - accuracy: 0.4412 - val\_loss: 0.6727 - val\_accuracy: 0.4395

Epoch 18/50

3090/3110 [============================>.] - ETA: 0s - loss: 0.6406 - accuracy: 0.4403

Epoch 18: val\_loss improved from 0.67270 to 0.67190, saving model to autoencoder\_fraud.h5

3110/3110 [==============================] - 6s 2ms/step - loss: 0.6406 - accuracy: 0.4402 - val\_loss: 0.6719 - val\_accuracy: 0.4403

Epoch 19/50

3098/3110 [============================>.] - ETA: 0s - loss: 0.6380 - accuracy: 0.4401

Epoch 19: val\_loss improved from 0.67190 to 0.67080, saving model to autoencoder\_fraud.h5

3110/3110 [==============================] - 5s 2ms/step - loss: 0.6390 - accuracy: 0.4402 - val\_loss: 0.6708 - val\_accuracy: 0.4440

Epoch 20/50

3077/3110 [============================>.] - ETA: 0s - loss: 0.6386 - accuracy: 0.4404

Epoch 20: val\_loss improved from 0.67080 to 0.66821, saving model to autoencoder\_fraud.h5

3110/3110 [==============================] - 5s 2ms/step - loss: 0.6377 - accuracy: 0.4403 - val\_loss: 0.6682 - val\_accuracy: 0.4492

Epoch 21/50

3075/3110 [============================>.] - ETA: 0s - loss: 0.6370 - accuracy: 0.4409

Epoch 21: val\_loss did not improve from 0.66821

3110/3110 [==============================] - 5s 2ms/step - loss: 0.6366 - accuracy: 0.4412 - val\_loss: 0.6682 - val\_accuracy: 0.4407

Epoch 22/50

3100/3110 [============================>.] - ETA: 0s - loss: 0.6361 - accuracy: 0.4425

Epoch 22: val\_loss improved from 0.66821 to 0.66651, saving model to autoencoder\_fraud.h5

3110/3110 [==============================] - 6s 2ms/step - loss: 0.6358 - accuracy: 0.4424 - val\_loss: 0.6665 - val\_accuracy: 0.4408

Epoch 23/50

3101/3110 [============================>.] - ETA: 0s - loss: 0.6350 - accuracy: 0.4438

Epoch 23: val\_loss improved from 0.66651 to 0.66529, saving model to autoencoder\_fraud.h5

3110/3110 [==============================] - 8s 3ms/step - loss: 0.6349 - accuracy: 0.4439 - val\_loss: 0.6653 - val\_accuracy: 0.4497

Epoch 24/50

3099/3110 [============================>.] - ETA: 0s - loss: 0.6341 - accuracy: 0.4440

Epoch 24: val\_loss improved from 0.66529 to 0.66453, saving model to autoencoder\_fraud.h5

3110/3110 [==============================] - 8s 3ms/step - loss: 0.6340 - accuracy: 0.4440 - val\_loss: 0.6645 - val\_accuracy: 0.4498

Epoch 25/50

3095/3110 [============================>.] - ETA: 0s - loss: 0.6330 - accuracy: 0.4465

Epoch 25: val\_loss improved from 0.66453 to 0.66373, saving model to autoencoder\_fraud.h5

3110/3110 [==============================] - 8s 3ms/step - loss: 0.6331 - accuracy: 0.4465 - val\_loss: 0.6637 - val\_accuracy: 0.4533

Epoch 26/50

3096/3110 [============================>.] - ETA: 0s - loss: 0.6327 - accuracy: 0.4486

Epoch 26: val\_loss improved from 0.66373 to 0.66291, saving model to autoencoder\_fraud.h5

3110/3110 [==============================] - 8s 2ms/step - loss: 0.6326 - accuracy: 0.4485 - val\_loss: 0.6629 - val\_accuracy: 0.4578

Epoch 27/50

3098/3110 [============================>.] - ETA: 0s - loss: 0.6326 - accuracy: 0.4504

Epoch 27: val\_loss did not improve from 0.66291

3110/3110 [==============================] - 5s 2ms/step - loss: 0.6319 - accuracy: 0.4504 - val\_loss: 0.6633 - val\_accuracy: 0.4493

Epoch 28/50

3089/3110 [============================>.] - ETA: 0s - loss: 0.6313 - accuracy: 0.4508

Epoch 28: val\_loss improved from 0.66291 to 0.66232, saving model to autoencoder\_fraud.h5

3110/3110 [==============================] - 5s 2ms/step - loss: 0.6315 - accuracy: 0.4508 - val\_loss: 0.6623 - val\_accuracy: 0.4508

Epoch 29/50

3074/3110 [============================>.] - ETA: 0s - loss: 0.6310 - accuracy: 0.4526

Epoch 29: val\_loss improved from 0.66232 to 0.66112, saving model to autoencoder\_fraud.h5

3110/3110 [==============================] - 5s 2ms/step - loss: 0.6308 - accuracy: 0.4527 - val\_loss: 0.6611 - val\_accuracy: 0.4585

Epoch 30/50

3105/3110 [============================>.] - ETA: 0s - loss: 0.6301 - accuracy: 0.4540

Epoch 30: val\_loss did not improve from 0.66112

3110/3110 [==============================] - 8s 3ms/step - loss: 0.6300 - accuracy: 0.4540 - val\_loss: 0.6614 - val\_accuracy: 0.4561

Epoch 31/50

3106/3110 [============================>.] - ETA: 0s - loss: 0.6295 - accuracy: 0.4555

Epoch 31: val\_loss improved from 0.66112 to 0.66017, saving model to autoencoder\_fraud.h5

3110/3110 [==============================] - 8s 3ms/step - loss: 0.6294 - accuracy: 0.4554 - val\_loss: 0.6602 - val\_accuracy: 0.4584

Epoch 32/50

3106/3110 [============================>.] - ETA: 0s - loss: 0.6291 - accuracy: 0.4547

Epoch 32: val\_loss improved from 0.66017 to 0.65953, saving model to autoencoder\_fraud.h5

3110/3110 [==============================] - 8s 3ms/step - loss: 0.6289 - accuracy: 0.4548 - val\_loss: 0.6595 - val\_accuracy: 0.4571

Epoch 33/50

3084/3110 [============================>.] - ETA: 0s - loss: 0.6279 - accuracy: 0.4572

Epoch 33: val\_loss improved from 0.65953 to 0.65854, saving model to autoencoder\_fraud.h5

3110/3110 [==============================] - 8s 3ms/step - loss: 0.6284 - accuracy: 0.4572 - val\_loss: 0.6585 - val\_accuracy: 0.4574

Epoch 34/50

3100/3110 [============================>.] - ETA: 0s - loss: 0.6280 - accuracy: 0.4569

Epoch 34: val\_loss improved from 0.65854 to 0.65738, saving model to autoencoder\_fraud.h5

3110/3110 [==============================] - 8s 3ms/step - loss: 0.6276 - accuracy: 0.4570 - val\_loss: 0.6574 - val\_accuracy: 0.4651

Epoch 35/50

3092/3110 [============================>.] - ETA: 0s - loss: 0.6274 - accuracy: 0.4586

Epoch 35: val\_loss did not improve from 0.65738

3110/3110 [==============================] - 8s 3ms/step - loss: 0.6273 - accuracy: 0.4586 - val\_loss: 0.6581 - val\_accuracy: 0.4631

Epoch 36/50

3106/3110 [============================>.] - ETA: 0s - loss: 0.6266 - accuracy: 0.4591

Epoch 36: val\_loss improved from 0.65738 to 0.65654, saving model to autoencoder\_fraud.h5

3110/3110 [==============================] - 8s 2ms/step - loss: 0.6266 - accuracy: 0.4591 - val\_loss: 0.6565 - val\_accuracy: 0.4607

Epoch 37/50

3090/3110 [============================>.] - ETA: 0s - loss: 0.6262 - accuracy: 0.4609

Epoch 37: val\_loss improved from 0.65654 to 0.65646, saving model to autoencoder\_fraud.h5

3110/3110 [==============================] - 5s 2ms/step - loss: 0.6266 - accuracy: 0.4610 - val\_loss: 0.6565 - val\_accuracy: 0.4586

Epoch 38/50

3073/3110 [============================>.] - ETA: 0s - loss: 0.6268 - accuracy: 0.4611

Epoch 38: val\_loss did not improve from 0.65646

3110/3110 [==============================] - 5s 1ms/step - loss: 0.6259 - accuracy: 0.4609 - val\_loss: 0.6574 - val\_accuracy: 0.4657

Epoch 39/50

3086/3110 [============================>.] - ETA: 0s - loss: 0.6256 - accuracy: 0.4611

Epoch 39: val\_loss did not improve from 0.65646

3110/3110 [==============================] - 5s 2ms/step - loss: 0.6255 - accuracy: 0.4612 - val\_loss: 0.6565 - val\_accuracy: 0.4626

Epoch 40/50

3078/3110 [============================>.] - ETA: 0s - loss: 0.6251 - accuracy: 0.4618

Epoch 40: val\_loss improved from 0.65646 to 0.65481, saving model to autoencoder\_fraud.h5

3110/3110 [==============================] - 5s 2ms/step - loss: 0.6253 - accuracy: 0.4619 - val\_loss: 0.6548 - val\_accuracy: 0.4683

Epoch 41/50

3093/3110 [============================>.] - ETA: 0s - loss: 0.6257 - accuracy: 0.4623

Epoch 41: val\_loss did not improve from 0.65481

3110/3110 [==============================] - 8s 2ms/step - loss: 0.6250 - accuracy: 0.4624 - val\_loss: 0.6553 - val\_accuracy: 0.4596

Epoch 42/50

3103/3110 [============================>.] - ETA: 0s - loss: 0.6250 - accuracy: 0.4614

Epoch 42: val\_loss did not improve from 0.65481

3110/3110 [==============================] - 7s 2ms/step - loss: 0.6248 - accuracy: 0.4614 - val\_loss: 0.6549 - val\_accuracy: 0.4678

Epoch 43/50

3110/3110 [==============================] - ETA: 0s - loss: 0.6247 - accuracy: 0.4635

Epoch 43: val\_loss did not improve from 0.65481

3110/3110 [==============================] - 8s 2ms/step - loss: 0.6247 - accuracy: 0.4635 - val\_loss: 0.6561 - val\_accuracy: 0.4611

Epoch 44/50

3073/3110 [============================>.] - ETA: 0s - loss: 0.6242 - accuracy: 0.4636

Epoch 44: val\_loss improved from 0.65481 to 0.65384, saving model to autoencoder\_fraud.h5

3110/3110 [==============================] - 7s 2ms/step - loss: 0.6244 - accuracy: 0.4638 - val\_loss: 0.6538 - val\_accuracy: 0.4641

Epoch 45/50

3092/3110 [============================>.] - ETA: 0s - loss: 0.6241 - accuracy: 0.4639

Epoch 45: val\_loss did not improve from 0.65384

3110/3110 [==============================] - 5s 2ms/step - loss: 0.6242 - accuracy: 0.4639 - val\_loss: 0.6569 - val\_accuracy: 0.4725

Epoch 46/50

3077/3110 [============================>.] - ETA: 0s - loss: 0.6244 - accuracy: 0.4633

Epoch 46: val\_loss did not improve from 0.65384

3110/3110 [==============================] - 5s 2ms/step - loss: 0.6239 - accuracy: 0.4633 - val\_loss: 0.6541 - val\_accuracy: 0.4664

Epoch 47/50

3071/3110 [============================>.] - ETA: 0s - loss: 0.6246 - accuracy: 0.4646

Epoch 47: val\_loss did not improve from 0.65384

3110/3110 [==============================] - 5s 1ms/step - loss: 0.6235 - accuracy: 0.4645 - val\_loss: 0.6543 - val\_accuracy: 0.4665

Epoch 48/50

3095/3110 [============================>.] - ETA: 0s - loss: 0.6222 - accuracy: 0.4640

Epoch 48: val\_loss did not improve from 0.65384

3110/3110 [==============================] - 6s 2ms/step - loss: 0.6232 - accuracy: 0.4641 - val\_loss: 0.6542 - val\_accuracy: 0.4833

Epoch 49/50

3091/3110 [============================>.] - ETA: 0s - loss: 0.6222 - accuracy: 0.4652

Epoch 49: val\_loss did not improve from 0.65384

3110/3110 [==============================] - 8s 2ms/step - loss: 0.6231 - accuracy: 0.4652 - val\_loss: 0.6542 - val\_accuracy: 0.4655

Epoch 50/50

3098/3110 [============================>.] - ETA: 0s - loss: 0.6231 - accuracy: 0.4647

Epoch 50: val\_loss improved from 0.65384 to 0.65340, saving model to autoencoder\_fraud.h5

3110/3110 [==============================] - 8s 2ms/step - loss: 0.6228 - accuracy: 0.4647 - val\_loss: 0.6534 - val\_accuracy: 0.4692

In [16]:

*#Plot training and test loss*

​

plt.plot(history['loss'], linewidth**=**2, label**=**'Train')

plt.plot(history['val\_loss'], linewidth**=**2, label**=**'Test')

plt.legend(loc**=**'upper right')

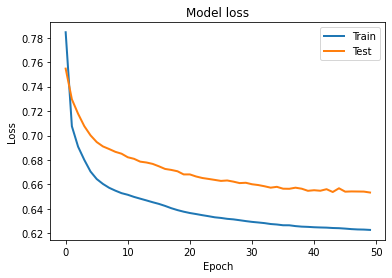
plt.title('Model loss')

plt.ylabel('Loss')

plt.xlabel('Epoch')

*#plt.ylim(ymin=0.70,ymax=1)*

plt.show()



In [17]:

"""Detect Anomalies on test data

​

Anomalies are data points where the reconstruction loss is higher

​

To calculate the reconstruction loss on test data,

predict the test data and calculate the mean square error between the test data and the reconstructed test data."""

​

test\_x\_predictions **=** autoencoder.predict(test\_x)

mse **=** np.mean(np.power(test\_x **-** test\_x\_predictions, 2), axis**=**1)

error\_df **=** pd.DataFrame({'Reconstruction\_error': mse,

'True\_class': test\_y})

error\_df.describe()

2671/2671 [==============================] - 4s 1ms/step

Out[17]:

|  | **Reconstruction\_error** | **True\_class** |
| --- | --- | --- |
| **count** | 85443.000000 | 85443.000000 |
| **mean** | 0.653400 | 0.001662 |
| **std** | 3.513865 | 0.040733 |
| **min** | 0.035542 | 0.000000 |
| **25%** | 0.166302 | 0.000000 |
| **50%** | 0.273823 | 0.000000 |
| **75%** | 0.490919 | 0.000000 |
| **max** | 308.161388 | 1.000000 |

In [18]:

*#Plotting the test data points and their respective reconstruction error sets a threshold value to visualize*

*#if the threshold value needs to be adjusted.*

​

threshold\_fixed **=** 50

groups **=** error\_df.groupby('True\_class')

fig, ax **=** plt.subplots()

**for** name, group **in** groups:

ax.plot(group.index, group.Reconstruction\_error, marker**=**'o', ms**=**3.5, linestyle**=**'',

label**=** "Fraud" **if** name **==** 1 **else** "Normal")

ax.hlines(threshold\_fixed, ax.get\_xlim()[0], ax.get\_xlim()[1], colors**=**"r", zorder**=**100, label**=**'Threshold')

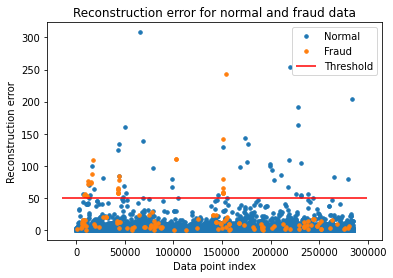
ax.legend()

plt.title("Reconstruction error for normal and fraud data")

plt.ylabel("Reconstruction error")

plt.xlabel("Data point index")

plt.show();



In [19]:

'''Detect anomalies as points where the reconstruction loss is greater than a fixed threshold.

Here we see that a value of 52 for the threshold will be good.

​

Evaluating the performance of the anomaly detection'''

​

threshold\_fixed **=**52

pred\_y **=** [1 **if** e **>** threshold\_fixed **else** 0 **for** e **in** error\_df.Reconstruction\_error.values]

error\_df['pred'] **=**pred\_y

conf\_matrix **=** confusion\_matrix(error\_df.True\_class, pred\_y)

plt.figure(figsize**=**(4, 4))

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()

*# print Accuracy, precision and recall*

print(" Accuracy: ",accuracy\_score(error\_df['True\_class'], error\_df['pred']))

print(" Recall: ",recall\_score(error\_df['True\_class'], error\_df['pred']))

print(" Precision: ",precision\_score(error\_df['True\_class'], error\_df['pred']))

**---------------------------------------------------------------------------**

**NameError** Traceback (most recent call last)

Input **In [19]**, in <cell line: 11>**()**

9 conf\_matrix = confusion\_matrix(error\_df.True\_class, pred\_y)

10 plt.figure(figsize=(4, 4))

**---> 11** sns.heatmap(conf\_matrix, xticklabels=LABELS, yticklabels=LABELS, annot=**True**, fmt="d");

12 plt.title("Confusion matrix")

13 plt.ylabel('True class')

**NameError**: name 'sns' is not defined

<Figure size 288x288 with 0 Axes>

In [ ]:

'''As our dataset is highly imbalanced, we see a high accuracy but a low recall and precision.

​

Things to further improve precision and recall would add more relevant features,

different architecture for autoencoder, different hyperparameters, or a different algorithm.'''