

```
import pandas as pd
import numpy as np
import tensorflow as tf
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
```

```
!pip install tensorflow --user
!pip install keras
!pip install daytime
!pip install torch
```

```
Requirement already satisfied: setuptools in /usr/local/lib/python3.10/dist-packages (from tensorflow) (75.1.0)
Requirement already satisfied: six>=1.12.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (1.16.0)
Requirement already satisfied: termcolor>=1.1.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (2.5.0)
Requirement already satisfied: typing-extensions>=3.6.6 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (4.12.2)
Requirement already satisfied: wrapt>=1.11.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (1.16.0)
Requirement already satisfied: grpcio<2.0,>=1.24.3 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (1.64.1)
Requirement already satisfied: tensorboard<2.18,>=2.17 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (2.17.0)
Requirement already satisfied: keras>=3.2.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (3.4.1)
Requirement already satisfied: tensorflow-io-gcs-filesystem>=0.23.1 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (1.26.4)
Requirement already satisfied: numpy<2.0.0,>=1.23.5 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (1.26.4)
Requirement already satisfied: wheel<1.0,>=0.23.0 in /usr/local/lib/python3.10/dist-packages (from astunparse>=1.6.0->tensorflow) (0.42.0)
Requirement already satisfied: rich in /usr/local/lib/python3.10/dist-packages (from keras>=3.2.0->tensorflow) (13.9.3)
Requirement already satisfied: namex in /usr/local/lib/python3.10/dist-packages (from keras>=3.2.0->tensorflow) (0.0.8)
Requirement already satisfied: optree in /usr/local/lib/python3.10/dist-packages (from keras>=3.2.0->tensorflow) (0.13.0)
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.10/dist-packages (from requests<3,>=2.21.0->ten) (3.0.1)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-packages (from requests<3,>=2.21.0->tensorflow) (3.0.1)
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.10/dist-packages (from requests<3,>=2.21.0->tensorflow) (2.0.7)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.10/dist-packages (from requests<3,>=2.21.0->tensorflow) (2024.7.4)
Requirement already satisfied: markdown>=2.6.8 in /usr/local/lib/python3.10/dist-packages (from tensorboard<2.18,>=2.17->tensorflow) (3.6.0)
Requirement already satisfied: tensorboard-data-server<0.8.0,>=0.7.0 in /usr/local/lib/python3.10/dist-packages (from tensorboard<2.18,>=2.17->tensorflow) (0.17.0)
Requirement already satisfied: werkzeug>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from tensorboard<2.18,>=2.17->tensorflow) (3.0.0)
Requirement already satisfied: MarkupSafe>=2.1.1 in /usr/local/lib/python3.10/dist-packages (from werkzeug>=1.0.1->tensorboard<2.18,>=2.17->tensorflow) (2.1.1)
Requirement already satisfied: markdown-it-py>=2.2.0 in /usr/local/lib/python3.10/dist-packages (from rich->keras>=3.2.0->tensorflow) (3.0.0)
Requirement already satisfied: pygments<3.0.0,>=2.13.0 in /usr/local/lib/python3.10/dist-packages (from rich->keras>=3.2.0->tensorflow) (2.18.0)
Requirement already satisfied: mdurl<=0.1 in /usr/local/lib/python3.10/dist-packages (from markdown-it-py>=2.2.0->rich->keras>=3.2.0->tensorflow) (0.1.2)
Requirement already satisfied: keras in /usr/local/lib/python3.10/dist-packages (3.4.1)
Requirement already satisfied: absl-py in /usr/local/lib/python3.10/dist-packages (from keras) (1.4.0)
Requirement already satisfied: numpy in /usr/local/lib/python3.10/dist-packages (from keras) (1.26.4)
Requirement already satisfied: rich in /usr/local/lib/python3.10/dist-packages (from keras) (13.9.3)
Requirement already satisfied: namex in /usr/local/lib/python3.10/dist-packages (from keras) (0.0.8)
Requirement already satisfied: h5py in /usr/local/lib/python3.10/dist-packages (from keras) (3.12.1)
Requirement already satisfied: optree in /usr/local/lib/python3.10/dist-packages (from keras) (0.13.0)
Requirement already satisfied: ml-dtypes in /usr/local/lib/python3.10/dist-packages (from keras) (0.4.1)
Requirement already satisfied: packaging in /usr/local/lib/python3.10/dist-packages (from keras) (24.1)
Requirement already satisfied: typing-extensions>=4.5.0 in /usr/local/lib/python3.10/dist-packages (from optree->keras) (4.12.2)
Requirement already satisfied: markdown-it-py>=2.2.0 in /usr/local/lib/python3.10/dist-packages (from rich->keras) (3.0.0)
Requirement already satisfied: pygments<3.0.0,>=2.13.0 in /usr/local/lib/python3.10/dist-packages (from rich->keras) (2.18.0)
Requirement already satisfied: mdurl<=0.1 in /usr/local/lib/python3.10/dist-packages (from markdown-it-py>=2.2.0->rich->keras) (0.1.2)
Collecting daytime
  Downloading daytime-0.4.tar.gz (2.4 kB)
  Preparing metadata (setup.py) ... done
Building wheels for collected packages: daytime
  Building wheel for daytime (setup.py) ... done
  Created wheel for daytime: filename=daytime-0.4-py3-none-any.whl size=2402 sha256=5ceca547295d4d502ee6025d6735142ef2302df04785c
  Stored in directory: /root/.cache/pip/wheels/cd/40/c7/fc109bc6716d31e4d5fcd0cd72891253fa46032e71d9aa1b93
Successfully built daytime
Installing collected packages: daytime
Successfully installed daytime-0.4
Requirement already satisfied: torch in /usr/local/lib/python3.10/dist-packages (2.5.0+cu121)
Requirement already satisfied: filelock in /usr/local/lib/python3.10/dist-packages (from torch) (3.16.1)
Requirement already satisfied: typing-extensions>=4.8.0 in /usr/local/lib/python3.10/dist-packages (from torch) (4.12.2)
Requirement already satisfied: networkx in /usr/local/lib/python3.10/dist-packages (from torch) (3.4.2)
Requirement already satisfied: jinja2 in /usr/local/lib/python3.10/dist-packages (from torch) (3.1.4)
Requirement already satisfied: fsspec in /usr/local/lib/python3.10/dist-packages (from torch) (2024.10.0)
Requirement already satisfied: sympy>=1.13.1 in /usr/local/lib/python3.10/dist-packages (from torch) (1.13.1)
Requirement already satisfied: mpmath<1.4,>=1.1.0 in /usr/local/lib/python3.10/dist-packages (from sympy>=1.13.1->torch) (1.3.0)
Requirement already satisfied: MarkupSafe>=2.0 in /usr/local/lib/python3.10/dist-packages (from jinja2->torch) (3.0.2)
```

```
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"]

#dataset = pd.read_csv("E:\Teaching material\Deep learning BE IT 2019 course\creditcard.csv")
dataset = pd.read_csv("creditcard.csv")
#dataset.head
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',

	Time	V1	V2	V3	V4	V5	V6	V7	V8
count	39702.000000	39702.000000	39702.000000	39702.000000	39702.000000	39702.000000	39702.000000	39702.000000	39702.000000
mean	25502.432497	-0.213932	0.051942	0.713780	0.189431	-0.231083	0.100915	-0.112981	0.041731
std	12626.308343	1.831757	1.548373	1.506728	1.399784	1.379431	1.304722	1.244260	1.222742
min	0.000000	-30.552380	-40.978852	-31.103685	-5.172595	-42.147898	-23.496714	-26.548144	-41.484823
25%	14678.500000	-0.967117	-0.525207	0.233522	-0.717668	-0.835646	-0.638647	-0.597725	-0.152448
50%	30598.500000	-0.233669	0.101393	0.818163	0.185015	-0.267851	-0.158696	-0.073409	0.048385
75%	35625.000000	1.160708	0.743741	1.447844	1.071347	0.297049	0.487010	0.433877	0.314947
max	39929.000000	1.960497	16.713389	4.101716	13.143668	34.099309	22.529298	36.677268	20.007208

8 rows × 31 columns

```
#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 True

No. of unique labels 3

Label values [0. 1. nan]

Break down of the Normal and Fraud Transactions

Class

0.0 39597

1.0 104

Name: count, dtype: int64

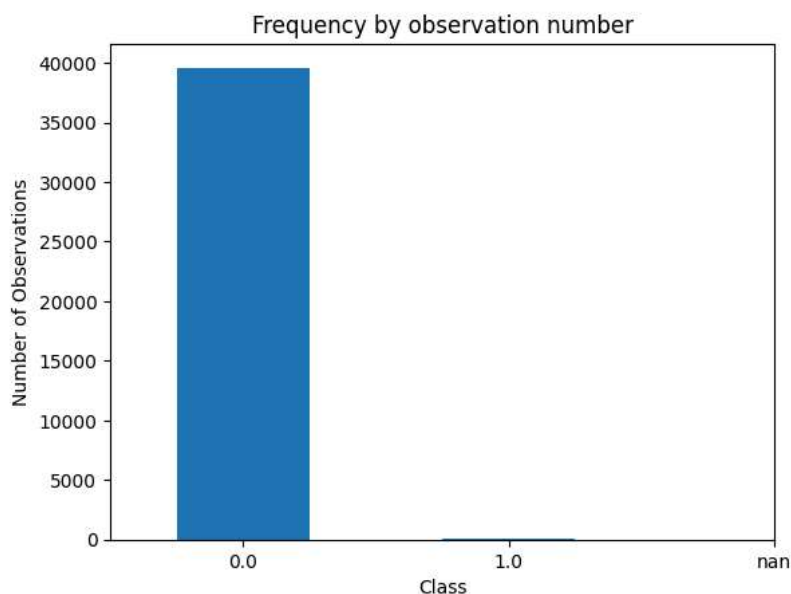
<ipython-input-5-d0648ee76bd2>:10: FutureWarning: pandas.value_counts is deprecated and will be removed in a future version. Use pd.value_counts instead.

print(pd.value_counts(dataset['Class'], sort = True))

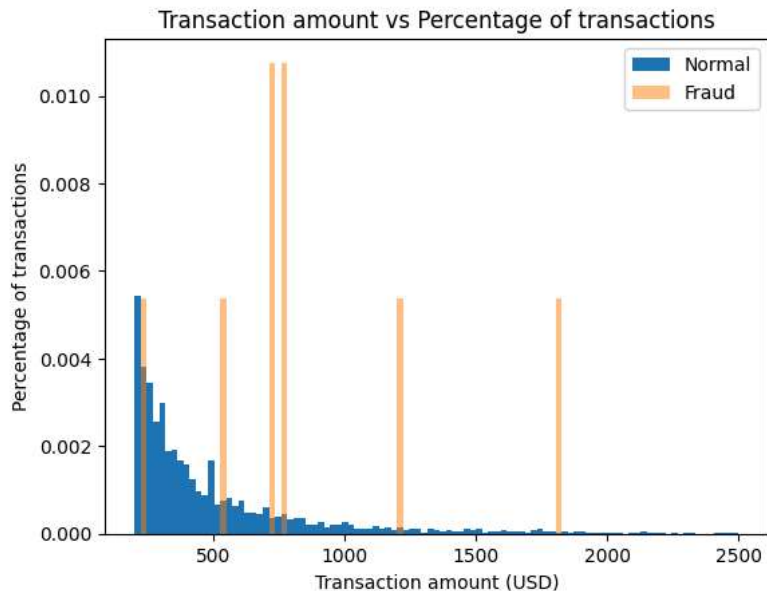
```
#Visualizing the imbalanced dataset
count_classes = pd.value_counts(dataset['Class'], sort = True)
count_classes.plot(kind = 'bar', rot=0)
plt.xticks(range(len(dataset['Class'].unique())), dataset.Class.unique())
plt.title("Frequency by observation number")
plt.xlabel("Class")
plt.ylabel("Number of Observations");
```

⌕ <ipython-input-6-402fd8456ddc>:2: FutureWarning: pandas.value_counts is deprecated and will be removed in a future version. Use pd.value_counts instead.

count_classes = pd.value_counts(dataset['Class'], sort = True)



```
# Save the normal and fraudulent 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)
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()
```



```
'''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.'''
```

⌕ '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.'

```
sc=StandardScaler()
dataset['Time'] = sc.fit_transform(dataset['Time'].values.reshape(-1, 1))
dataset['Amount'] = sc.fit_transform(dataset['Amount'].values.reshape(-1, 1))
```

```
'''The last column in the dataset is our target variable.'''
```

```
raw_data = dataset.values
# The last element contains if the transaction is normal which is represented by a 0 and if fraud then 1
labels = raw_data[:, -1]
# The other data points are the electrocardiogram data
data = raw_data[:, 0:-1]
train_data, test_data, train_labels, test_labels = train_test_split(
    data, labels, test_size=0.2, random_state=2021
)
```

```
'''Normalize the data to have a value between 0 and 1'''
```

```
min_val = tf.reduce_min(train_data)
max_val = tf.reduce_max(train_data)
train_data = (train_data - min_val) / (max_val - min_val)
test_data = (test_data - min_val) / (max_val - min_val)
train_data = tf.cast(train_data, tf.float32)
test_data = tf.cast(test_data, tf.float32)
```

```
'''Use only normal transactions to train the Autoencoder.
```

```
Normal data has a value of 0 in the target variable. Using the target variable to create a normal and fraud dataset.'''
```

```
train_labels = train_labels.astype(bool)
test_labels = test_labels.astype(bool)
```


```
#creating normal and fraud datasets
```

```
normal_train_data = train_data[~train_labels]
normal_test_data = test_data[~test_labels]
```

```

fraud_train_data = train_data[train_labels]
fraud_test_data = test_data[test_labels]
print(" No. of records in Fraud Train Data=",len(fraud_train_data))
print(" No. of records in Normal Train data=",len(normal_train_data))
print(" No. of records in Fraud Test Data=",len(fraud_test_data))
print(" No. of records in Normal Test data=",len(normal_test_data))

```

 No. of records in Fraud Train Data= 79
 No. of records in Normal Train data= 31682
 No. of records in Fraud Test Data= 26
 No. of records in Normal Test data= 7915

```

nb_epoch = 50
batch_size = 64
input_dim = normal_train_data.shape[1] #num of columns, 30
encoding_dim = 14
hidden_dim_1 = int(encoding_dim / 2) #
hidden_dim_2=4
learning_rate = 1e-7

```

```

#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: "functional"

Layer (type)	Output Shape	Param #
input_layer (InputLayer)	(None, 30)	0
dense (Dense)	(None, 14)	434
dropout (Dropout)	(None, 14)	0
dense_1 (Dense)	(None, 7)	105
dense_2 (Dense)	(None, 4)	32
dense_3 (Dense)	(None, 7)	35
dropout_1 (Dropout)	(None, 7)	0
dense_4 (Dense)	(None, 14)	112
dense_5 (Dense)	(None, 30)	450

Total params: 1,168 (4.56 KB)
 Trainable params: 1,168 (4.56 KB)
 Non-trainable params: 0 (0.00 B)

```

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

```

```

p=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)

```

```

-----
ValueError                                Traceback (most recent call last)
<ipython-input-19-9acff5e56152> in <cell line: 3>()
      1 """Define the callbacks for checkpoints and early stopping"""
      2
----> 3 p=tf.keras.callbacks.ModelCheckpoint(filepath="autoencoder_fraud.h5", mode='min', monitor='val_loss', verbose=2,
save_best_only=True)
      4 # define our early stopping
      5 early_stop = tf.keras.callbacks.EarlyStopping(

/usr/local/lib/python3.10/dist-packages/keras/src/callbacks/model_checkpoint.py in __init__(self, filepath, monitor, verbose,
save_best_only, save_weights_only, mode, save_freq, initial_value_threshold)
    189     else:
    190         if not self.filepath.endswith(".keras"):
--> 191             raise ValueError(
    192                 "The filepath provided must end in `.keras` "
    193                 "(Keras model format). Received: "

ValueError: The filepath provided must end in `.keras` (Keras model format). Received: filepath=autoencoder_fraud.h5

```

Next steps: [Explain error](#)

#Compile the Autoencoder

```

autoencoder.compile(metrics=['accuracy'],
                    loss='mean_squared_error',
                    optimizer='adam')

```

#Train the Autoencoder

```

history = autoencoder.fit(normal_train_data, normal_train_data,
                        epochs=nb_epoch,
                        batch_size=batch_size,
                        shuffle=True,
                        validation_data=(test_data, test_data),
                        verbose=1,

                        ).history

```

```

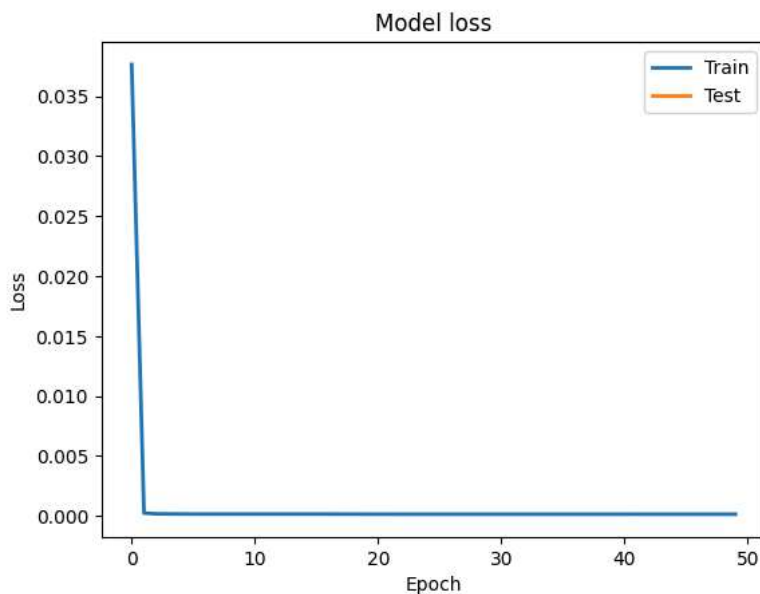
Epoch 22/50
496/496 ━━━━━━━━━━━ 2s 3ms/step - accuracy: 0.2005 - loss: 1.5430e-04 - val_accuracy: 0.1968 - val_loss: nan
Epoch 23/50
496/496 ━━━━━━━━━━━ 3s 4ms/step - accuracy: 0.1950 - loss: 1.5541e-04 - val_accuracy: 0.1985 - val_loss: nan

```

```
Epoch 45/50
496/496 ————— 1s 2ms/step - accuracy: 0.2054 - loss: 1.5571e-04 - val_accuracy: 0.1995 - val_loss: nan
Epoch 46/50
496/496 ————— 1s 2ms/step - accuracy: 0.1994 - loss: 1.5314e-04 - val_accuracy: 0.2118 - val_loss: nan
Epoch 47/50
496/496 ————— 1s 3ms/step - accuracy: 0.2082 - loss: 1.5931e-04 - val_accuracy: 0.2102 - val_loss: nan
Epoch 48/50
496/496 ————— 1s 2ms/step - accuracy: 0.1968 - loss: 1.5416e-04 - val_accuracy: 0.2118 - val_loss: nan
Epoch 49/50
496/496 ————— 1s 2ms/step - accuracy: 0.2022 - loss: 1.5131e-04 - val_accuracy: 0.2118 - val_loss: nan
Epoch 50/50
496/496 ————— 1s 2ms/step - accuracy: 0.2056 - loss: 1.5380e-04 - val_accuracy: 0.1781 - val_loss: nan
```

#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()
```



"""Detect Anomalies on test data

Anomalies are data points where the reconstruction loss is higher

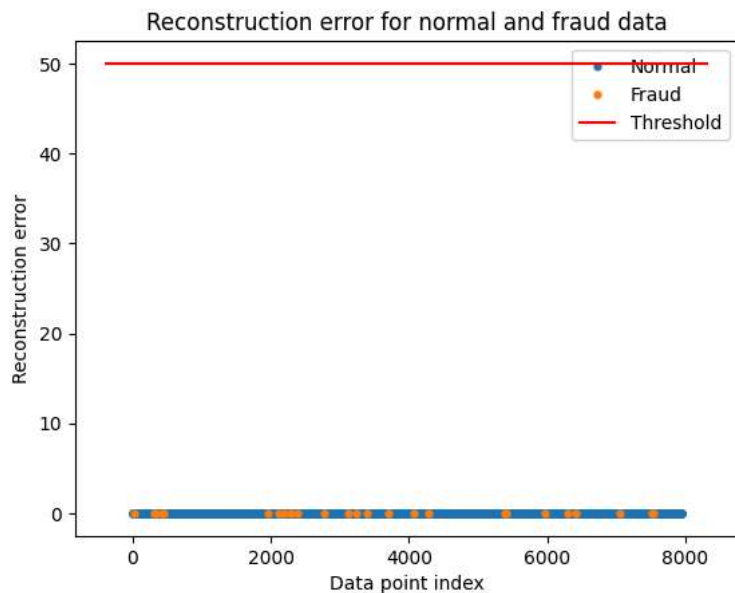
```
To calculate the reconstruction loss on test data,
predict the test data and calculate the mean square error between the test data and the reconstructed test data."""
test_x_predictions = autoencoder.predict(test_data)
mse = np.mean(np.power(test_data - test_x_predictions, 2), axis=1)
error_df = pd.DataFrame({'Reconstruction_error': mse,
                          'True_class': test_labels})
```



```
249/249 ————— 1s 2ms/step
```

#Plotting the test data points and their respective reconstruction error sets a threshold value to visualize
#if the threshold value needs to be adjusted.

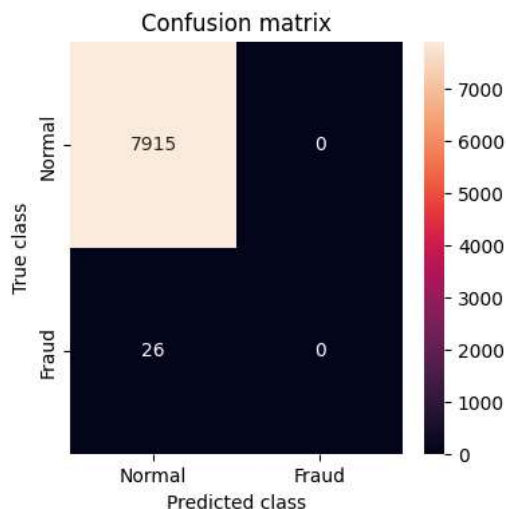
```
threshold_fixed = 50
groups = error_df.groupby('True_class')
fig, ax = plt.subplots()
for name, group in groups:
    ax.plot(group.index, group.Reconstruction_error, marker='o', ms=3.5, linestyle='',
            label= "Fraud" if name == 1 else "Normal")
ax.hlines(threshold_fixed, ax.get_xlim()[0], 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();
```



'''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']))
```



Accuracy: 0.9967258531671074

Recall: 0.0

Precision: 0.0

/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1531: UndefinedMetricWarning: Precision is ill-defined ar
_warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))

'''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.'''



'As our dataset is highly imbalanced, we see a high accuracy but a low recall and precision.\n\nThings to further improve precision and recall would add more relevant features,\ndifferent architecture for autoencoder, different hyperparameters, or a different algorithm '

