In [28]:
```
import pandas as pd
import numpy as np
import pickle
import matplotlib.pyplot as plt
from scipy import stats
import tensorflow as tf
from tensorflow import keras
import seaborn as sns
from pylab import rcParams
from sklearn.model_selection import train_test_split
from keras.models import Model, load_model
from keras.layers import Input, Dense
from keras.callbacks import ModelCheckpoint, TensorBoard
from keras import regularizers
```

In [29]:
```
sns.set(style='whitegrid', palette='muted', font_scale=1.5)

rcParams['figure.figsize'] = 14, 8

RANDOM_SEED = 42
LABELS = ["Normal", "Fraud"]
```

In [30]:
```
df =  pd.read_csv("creditcard.csv")
```

In [31]:
```
count_classes = pd.value_counts(df['Class'], sort = True)
count_classes.plot(kind = 'bar', rot=0)
plt.title("Transaction class distribution")
plt.xticks(range(2), LABELS)
plt.xlabel("Class")
plt.ylabel("Frequency");
```


In [32]:
```
from sklearn.preprocessing import StandardScaler

data = df.drop(['Time'], axis=1)
```

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

In [33]:
```
X_train, X_test = train_test_split(data, test_size=0.2, random_state=RANDOM_SEED)
X_train = X_train[X_train.Class == 0]
X_train = X_train.drop(['Class'], axis=1)

y_test = X_test['Class']
X_test = X_test.drop(['Class'], axis=1)

X_train = X_train.values
X_test = X_test.values
```

In [34]:
```
input_dim = X_train.shape[1]
encoding_dim = 14
```

In [35]:
```
input_layer = Input(shape=(input_dim, ))

encoder = Dense(encoding_dim, activation="tanh",
                activity_regularizer=regularizers.l1(10e-5))(input_layer)
encoder = Dense(int(encoding_dim / 2), activation="relu")(encoder)

decoder = Dense(int(encoding_dim / 2), activation='tanh')(encoder)
decoder = Dense(input_dim, activation='relu')(decoder)

autoencoder = Model(inputs=input_layer, outputs=decoder)
```

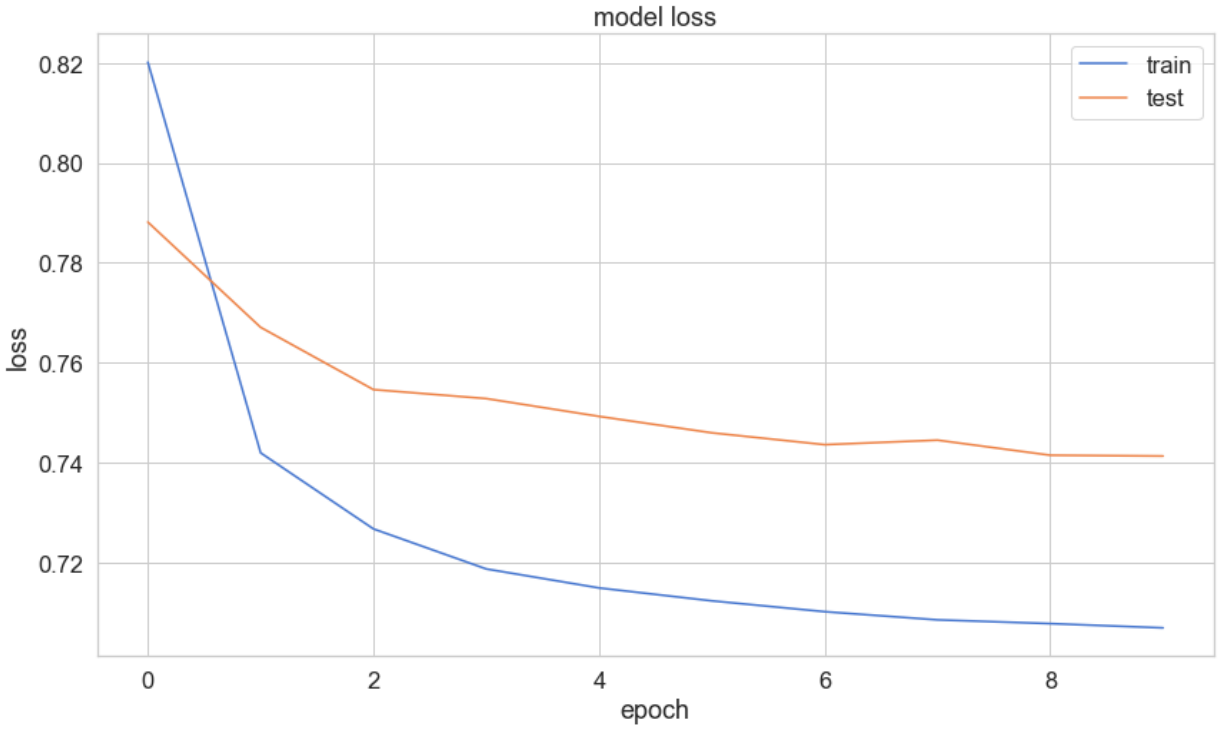
In [36]:
```
nb_epoch = 10
batch_size = 32
early_stop = tf.keras.callbacks. EarlyStopping( monitor= 'val_loss', min_delta=0.000
                                                restore_best_weights=True)

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

checkpointer = ModelCheckpoint(filepath="model.h5",
                               verbose=0,
                               save_best_only=True)
tensorboard = TensorBoard(log_dir='./logs',
                          histogram_freq=0,
                          write_graph=True,
                          write_images=True)

history = autoencoder.fit(X_train, X_train,
                    epochs=nb_epoch,
                    batch_size=batch_size,
                    shuffle=True,
                    validation_data=(X_test, X_test),
                    verbose=1,
                    callbacks=[checkpointer, early_stop]).history
```

```
Epoch 1/10
7108/7108 [==============================] - 25s 3ms/step - loss: 0.8202 - accuracy:
0.5658 - val_loss: 0.7882 - val_accuracy: 0.6362
Epoch 2/10
7108/7108 [==============================] - 18s 2ms/step - loss: 0.7420 - accuracy:
0.6594 - val_loss: 0.7671 - val_accuracy: 0.6709
Epoch 3/10
```

```
7108/7108 [==============================] - 9s 1ms/step - loss: 0.7268 - accuracy:
0.6833 - val_loss: 0.7547 - val_accuracy: 0.6916
Epoch 4/10
7108/7108 [==============================] - 9s 1ms/step - loss: 0.7188 - accuracy:
0.6918 - val_loss: 0.7529 - val_accuracy: 0.6780
Epoch 5/10
7108/7108 [==============================] - 9s 1ms/step - loss: 0.7150 - accuracy:
0.6959 - val_loss: 0.7493 - val_accuracy: 0.6952
Epoch 6/10
7108/7108 [==============================] - 9s 1ms/step - loss: 0.7124 - accuracy:
0.6992 - val_loss: 0.7460 - val_accuracy: 0.7003
Epoch 7/10
7108/7108 [==============================] - 8s 1ms/step - loss: 0.7102 - accuracy:
0.7008 - val_loss: 0.7436 - val_accuracy: 0.7012
Epoch 8/10
7108/7108 [==============================] - 8s 1ms/step - loss: 0.7086 - accuracy:
0.7016 - val_loss: 0.7445 - val_accuracy: 0.6971
Epoch 9/10
7108/7108 [==============================] - 8s 1ms/step - loss: 0.7079 - accuracy:
0.7019 - val_loss: 0.7415 - val_accuracy: 0.7027
Epoch 10/10
7108/7108 [==============================] - 8s 1ms/step - loss: 0.7070 - accuracy:
0.7027 - val_loss: 0.7414 - val_accuracy: 0.7031
```

In [37]:

```
plt.plot(history['loss'])
plt.plot(history['val_loss'])
plt.title('model loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper right');
```




In [38]:

```
predictions = autoencoder.predict(X_test)
```

In [39]:

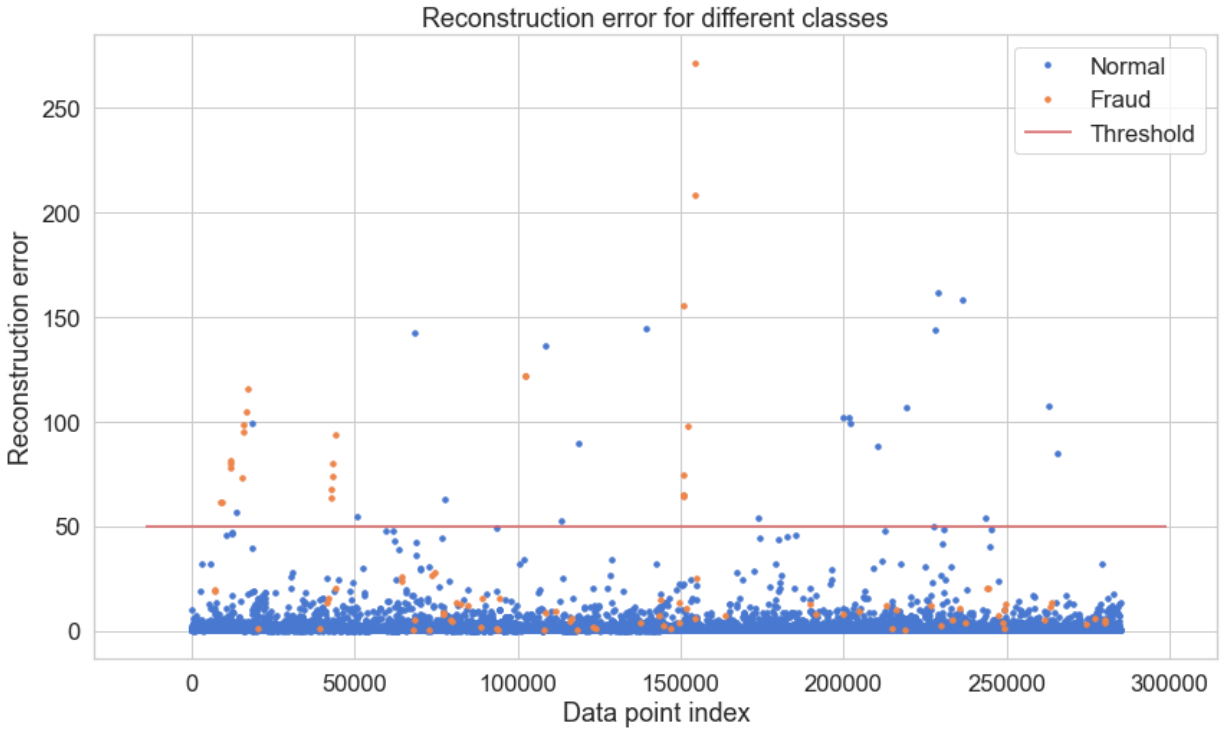
```
mse = np.mean(np.power(X_test - predictions, 2), axis=1)
error_df = pd.DataFrame({'reconstruction_error': mse,
                         'true_class': y_test})
```

In [40]:
```
error_df.describe()
```

Out[40]:

| | reconstruction_error | true_class |
|---|---|---|
| **count** | 56962.000000 | 56962.000000 |
| **mean** | 0.740476 | 0.001720 |
| **std** | 3.474243 | 0.041443 |
| **min** | 0.043415 | 0.000000 |
| **25%** | 0.240167 | 0.000000 |
| **50%** | 0.383599 | 0.000000 |
| **75%** | 0.612013 | 0.000000 |
| **max** | 271.266166 | 1.000000 |

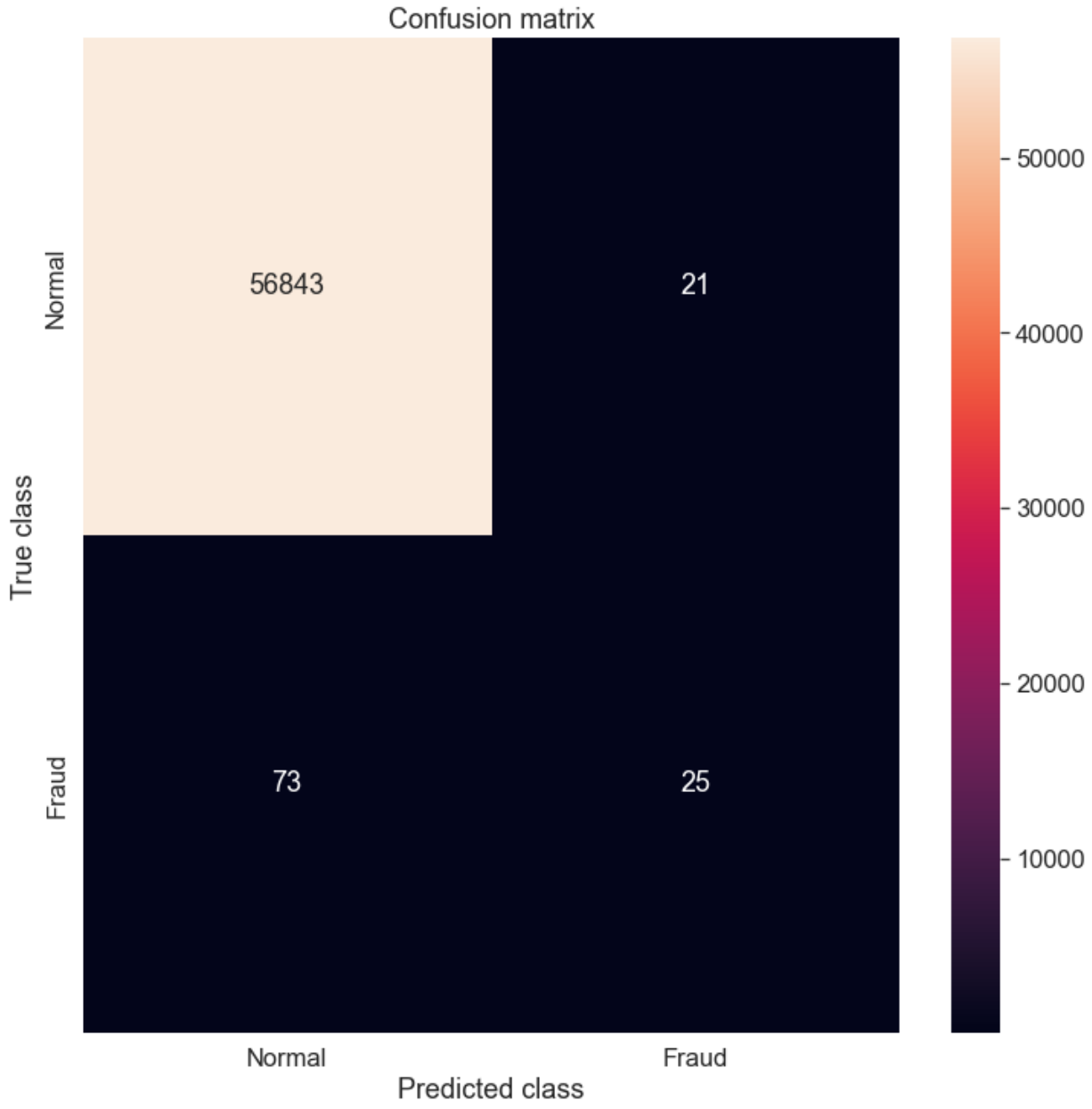
In [41]:
```
threshold = 50
groups = error_df.groupby('true_class')
fig, ax = plt.subplots()

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, ax.get_xlim()[0], ax.get_xlim()[1], colors="r", zorder=100, lab
ax.legend()
plt.title("Reconstruction error for different classes")
plt.ylabel("Reconstruction error")
plt.xlabel("Data point index")
plt.show();
```




In [42]:
```
from sklearn.metrics import confusion_matrix,recall_score,accuracy_score,precision_s
```

In [43]:

```
y_pred = [1 if e > threshold else 0 for e in error_df.reconstruction_error.values]
conf_matrix = confusion_matrix(error_df.true_class, y_pred)

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 [44]:

```
error_df['pred'] = y_pred
```

In [46]:

```
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: 0.9983497770443454
Recall: 0.25510204081632654
Precision: 0.5434782608695652
```