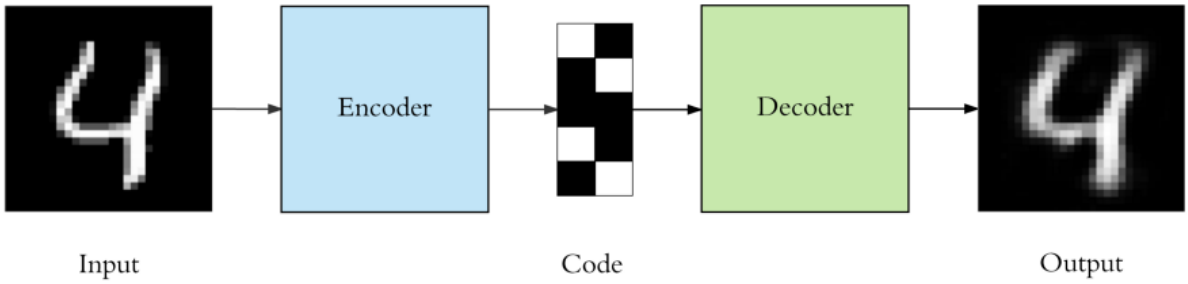
Erkennung von EKG Anomalien mit Hilfe eines Autoencoders



Ein Autoencoder ist eine spezielle Form von Neuralem Netzwerk. Beispielsweise wird ein Bild einer handgeschriebenen Zahl vom Autoencoder zunächst in ein Bild mit weniger Dimensionen codiert. Anschließend wird das reduzierte Bild wieder decodiert, wobei versucht wird, den Rekonstruktionsfehler zu minimieren.

Dieses Verfahren kann genutzt werden, um Anomalien in EKGs zu erkennen. Dazu wird das Modell nur mit Daten normaler EKGs trainiert, jedoch mit dem kompletten Datensatz getestet. Daraus ergibt sich bei der Rekonstruktion abnormaler EKGs ein größerer Rekonstruktionsfehler. Übersteigt die Größe des Fehlers eine definierte Schwelle, wird das EKG als abnormal kategorisiert.

In [1]:

```
import tensorflow as tf
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd

from tensorflow.keras import layers, losses
from sklearn.model_selection import train_test_split
from tensorflow.keras.models import Model
```

In [2]:

```
df = pd.read_csv('Input/ecg.csv', header=None)
df.head()
```

Out[2]:

	0	1	2	3	4	5	6	7	
0	-0.112522	-2.827204	-3.773897	-4.349751	-4.376041	-3.474986	-2.181408	-1.818286	-1.25
1	-1.100878	-3.996840	-4.285843	-4.506579	-4.022377	-3.234368	-1.566126	-0.992258	-0.75
2	-0.567088	-2.593450	-3.874230	-4.584095	-4.187449	-3.151462	-1.742940	-1.490659	-1.18
3	0.490473	-1.914407	-3.616364	-4.318823	-4.268016	-3.881110	-2.993280	-1.671131	-1.33
4	0.800232	-0.874252	-2.384761	-3.973292	-4.338224	-3.802422	-2.534510	-1.783423	-1.59

5 rows × 141 columns

Der Datensatz verfügt über 140 Spalten. Die letzte Spalte enthält die Werte 0 und 1, was für abnormales bzw. normales EKG steht und dient somit als Label.

In [3]:

```
# Separieren von Daten und Labels
data = df.iloc[:,:-1].values
labels = df.iloc[:,-1].values
labels
```

Out[3]:

```
array([1., 1., 1., ..., 0., 0., 0.])
```

In [4]:

```
train_data, test_data, train_labels, test_labels = train_test_split(data, labels, test_si
```

Normalisierung im Bereich [0-1]

Wir verwenden dafür folgende Formel:

$$x_{\text{norm}} = \frac{x - \min(x)}{\max(x) - \min(x)}$$

In [5]:

```
# Maximum und Minimum des Trainingssets berechnen
min = tf.reduce_min(train_data)
max = tf.reduce_max(train_data)

# Verwenden der Formel (data - min)/(max - min)
train_data = (train_data - min)/(max - min)
test_data = (test_data - min)/(max - min)

# Konvertieren der Daten zu float
train_data = tf.cast(train_data, dtype=tf.float32)
test_data = tf.cast(test_data, dtype=tf.float32)
```

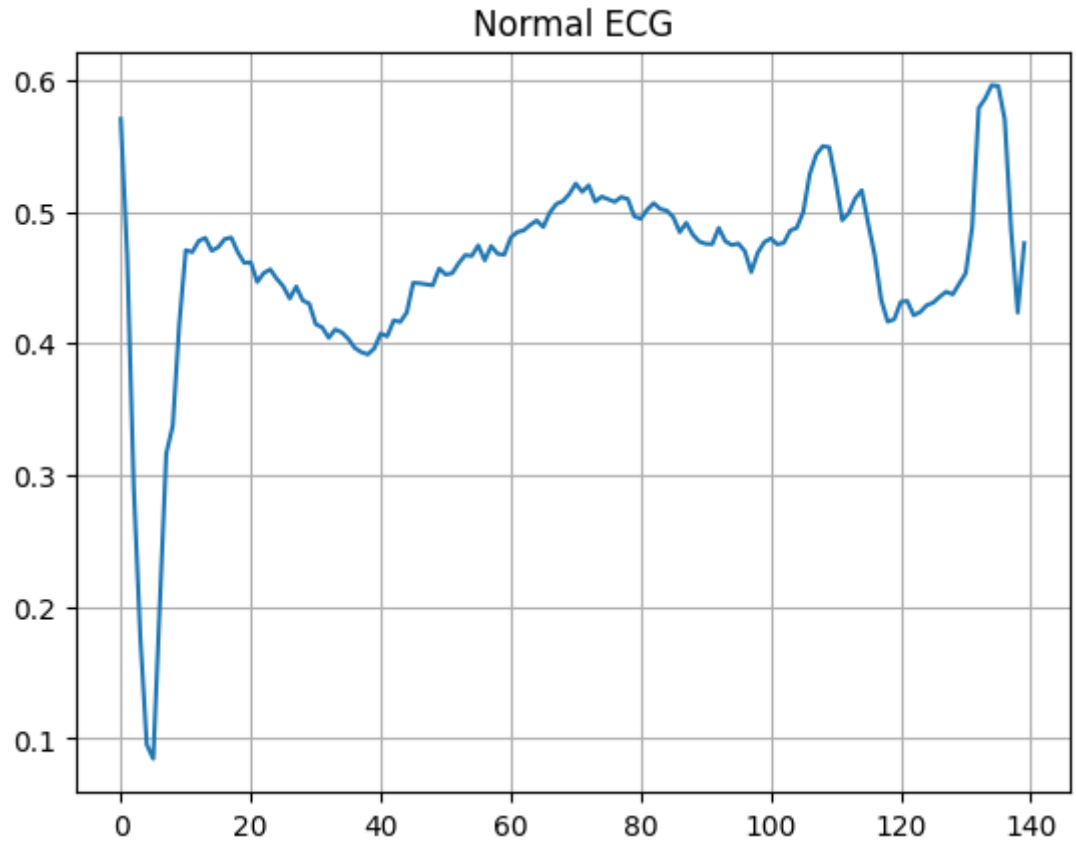
In [6]:

```
# Da die Labels 0 oder 1 sind, können sie zu Booleschen Werten konvertiert werden (true,
train_labels = train_labels.astype(bool)
test_labels = test_labels.astype(bool)

# Separieren der Daten von normalen EKGs von Daten abnormaler EKGs
#Normale EKG-Daten
n_train_data = train_data[train_labels]
n_test_data = test_data[test_labels]

#Abnormale EkG-Daten
an_train_data = train_data[~train_labels]
an_test_data = test_data[~test_labels]

print(n_train_data)
```

```
tf.Tensor(
[[0.57030463 0.46561658 0.29058117 ... 0.48504233 0.4233502  0.47598344]
 [0.48320588 0.28246963 0.16471253 ... 0.567567   0.4677294  0.2692329 ]
 [0.48144642 0.35151404 0.25972766 ... 0.5479421  0.5077544  0.54298663]
 ...
 [0.41039047 0.24164985 0.13120876 ... 0.5277313  0.5654091  0.5023885 ]
 [0.5397748  0.4140786  0.28101394 ... 0.51266515 0.43706053 0.4426865 ]
 [0.29639772 0.15988176 0.18883787 ... 0.53766966 0.545786   0.40826708]],
shape=(2359, 140), dtype=float32)
```

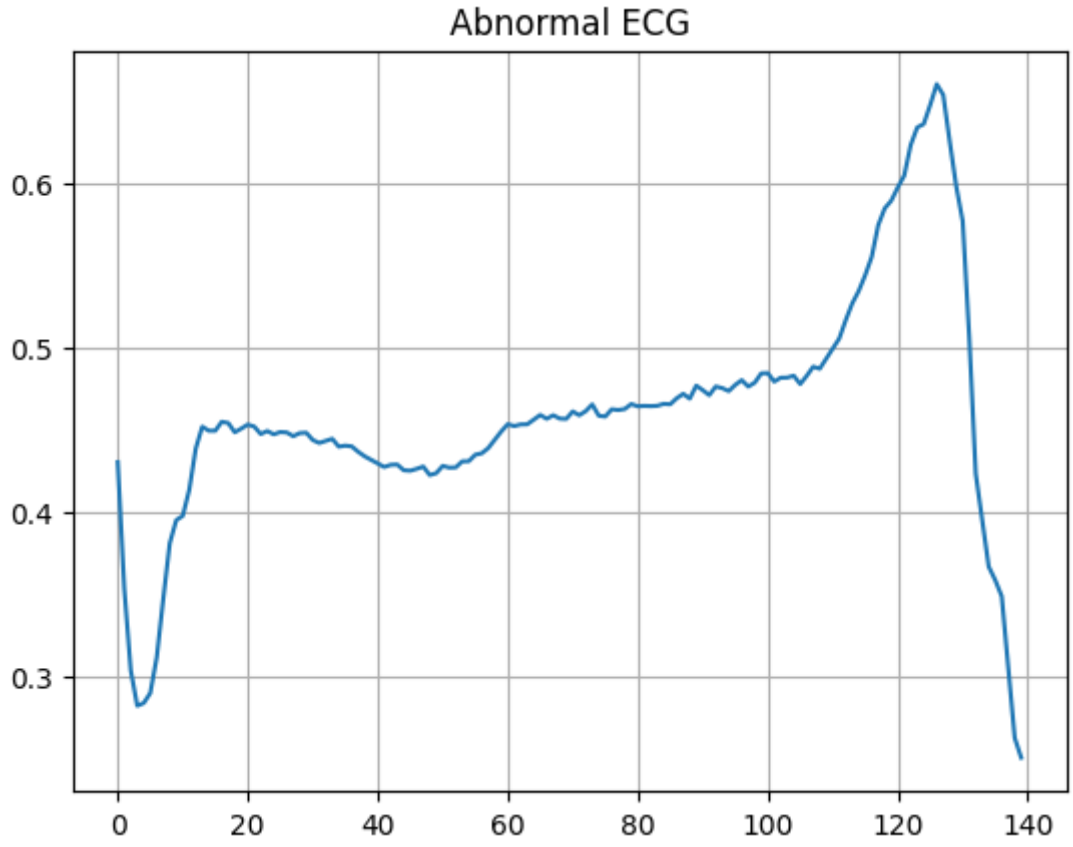
In [7]:

```
# Plot eines normalen EKG
plt.plot(np.arange(140), n_train_data[0])
plt.grid()
plt.title('Normal ECG')
plt.show()
```



In [8]:

```
# Plot eines abnormalen EKG
plt.plot(np.arange(140), an_train_data[0])
plt.grid()
plt.title('Abnormal ECG')
plt.show()
```



In [9]:

```
# Definition des Modells
# Das Modell besteht aus 2 Teilen : 1. Encoder and 2. Decoder

class detector(Model):
  def __init__(self):
    super(detector, self).__init__()
    self.encoder = tf.keras.Sequential([
                                        layers.Dense(32, activation='relu'),
                                        layers.Dense(16, activation='relu'),
                                        layers.Dense(8, activation='relu')
    ])
    self.decoder = tf.keras.Sequential([
                                        layers.Dense(16, activation='relu'),
                                        layers.Dense(32, activation='relu'),
                                        layers.Dense(140, activation='sigmoid')
    ])

  def call(self, x):
    encoded = self.encoder(x)
    decoded = self.decoder(encoded)
    return decoded
```

In [10]:

```
# Kompilieren und Training
autoencoder = detector()
autoencoder.compile(optimizer='adam', loss='mae')
autoencoder.fit(n_train_data, n_train_data, epochs = 20, batch_size=512, validation_data=
```

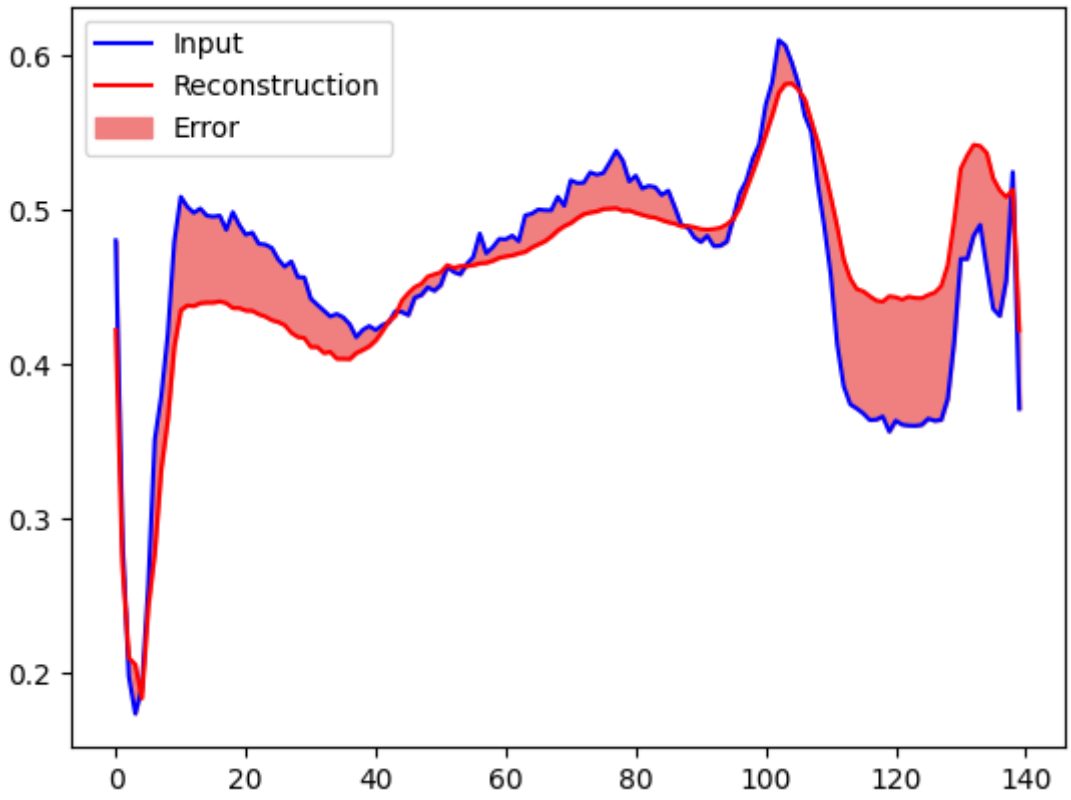
```
Epoch 1/20
5/5 [==============================] - 1s 49ms/step - loss: 0.0591 - val_l
oss: 0.0564
Epoch 2/20
5/5 [==============================] - 0s 5ms/step - loss: 0.0558 - val_lo
ss: 0.0543
Epoch 3/20
5/5 [==============================] - 0s 5ms/step - loss: 0.0534 - val_lo
ss: 0.0514
Epoch 4/20
5/5 [==============================] - 0s 5ms/step - loss: 0.0500 - val_lo
ss: 0.0472
Epoch 5/20
5/5 [==============================] - 0s 5ms/step - loss: 0.0456 - val_lo
ss: 0.0426
Epoch 6/20
5/5 [==============================] - 0s 5ms/step - loss: 0.0411 - val_lo
ss: 0.0383
Epoch 7/20
5/5 [==============================] - 0s 5ms/step - loss: 0.0370 - val_lo
ss: 0.0344
Epoch 8/20
5/5 [==============================] - 0s 5ms/step - loss: 0.0332 - val_lo
ss: 0.0312
Epoch 9/20
5/5 [==============================] - 0s 5ms/step - loss: 0.0304 - val_lo
ss: 0.0288
Epoch 10/20
5/5 [==============================] - 0s 5ms/step - loss: 0.0282 - val_lo
ss: 0.0269
Epoch 11/20
5/5 [==============================] - 0s 5ms/step - loss: 0.0265 - val_lo
ss: 0.0254
Epoch 12/20
5/5 [==============================] - 0s 5ms/step - loss: 0.0251 - val_lo
ss: 0.0241
Epoch 13/20
5/5 [==============================] - 0s 5ms/step - loss: 0.0239 - val_lo
ss: 0.0231
Epoch 14/20
5/5 [==============================] - 0s 5ms/step - loss: 0.0231 - val_lo
ss: 0.0224
Epoch 15/20
5/5 [==============================] - 0s 5ms/step - loss: 0.0226 - val_lo
ss: 0.0220
Epoch 16/20
5/5 [==============================] - 0s 5ms/step - loss: 0.0221 - val_lo
ss: 0.0215
Epoch 17/20
5/5 [==============================] - 0s 5ms/step - loss: 0.0217 - val_lo
ss: 0.0211
Epoch 18/20
5/5 [==============================] - 0s 5ms/step - loss: 0.0212 - val_lo
ss: 0.0207
Epoch 19/20
5/5 [==============================] - 0s 5ms/step - loss: 0.0209 - val_lo
ss: 0.0205
Epoch 20/20
5/5 [==============================] - 0s 5ms/step - loss: 0.0207 - val_lo
ss: 0.0203
```

Out[10]:

```
<keras.callbacks.History at 0x1c917dfbdf0>
```

In [11]:

```
# Definition einer Funktion zum Plot des originalen EKG und seiner Rekonstruktion inklusi
def plot(data, n):
  enc_img = autoencoder.encoder(data)
  dec_img = autoencoder.decoder(enc_img)
  plt.plot(data[n], 'b')
  plt.plot(dec_img[n], 'r')
  plt.fill_between(np.arange(140), data[n], dec_img[n], color = 'lightcoral')
  plt.legend(labels=['Input', 'Reconstruction', 'Error'])
  plt.show()

plot(n_test_data, 0)
plot(an_test_data, 0)
```



In [12]:

```
# Festlegen des Schwellenwerts für den Fehler
reconstructed = autoencoder(n_train_data)
train_loss = losses.mae(reconstructed, n_train_data)
t = np.mean(train_loss) + np.std(train_loss)  # Schwelle als Standartabweichung vom Mitte
def prediction(model, data, threshold):
  rec = model(data)
  loss = losses.mae(rec, data)
  return tf.math.less(loss, threshold)
print(t)
```

```
0.032914322
```

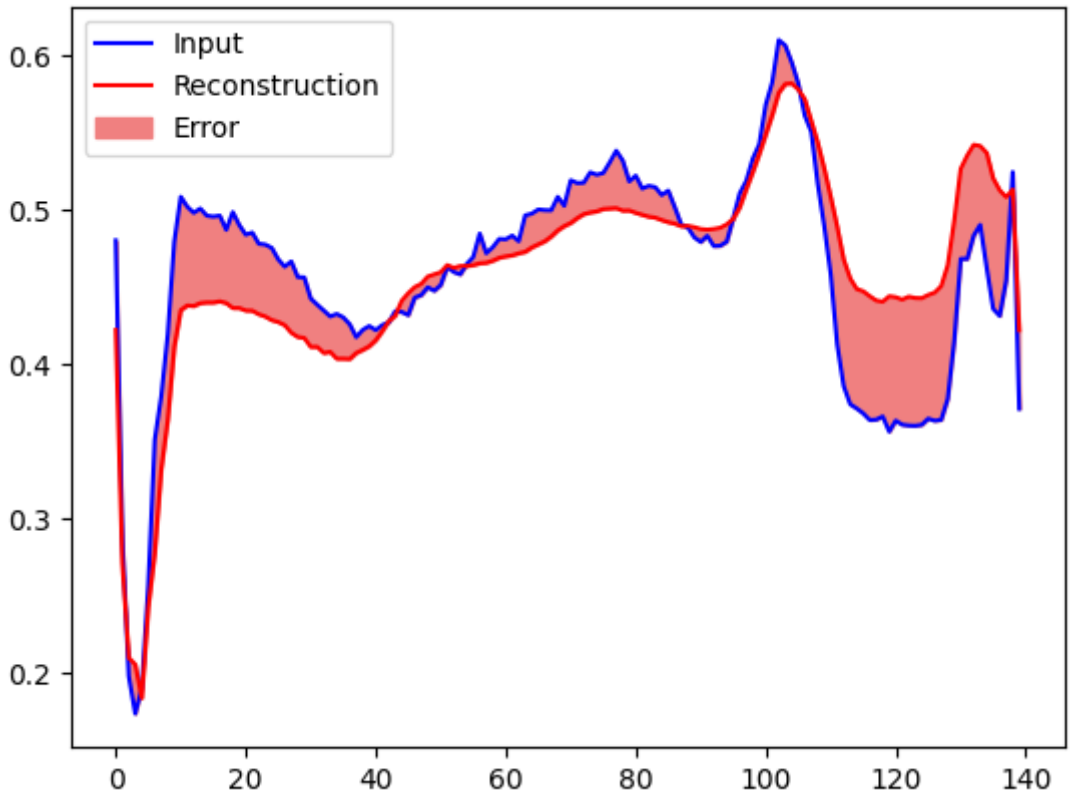
In [13]:

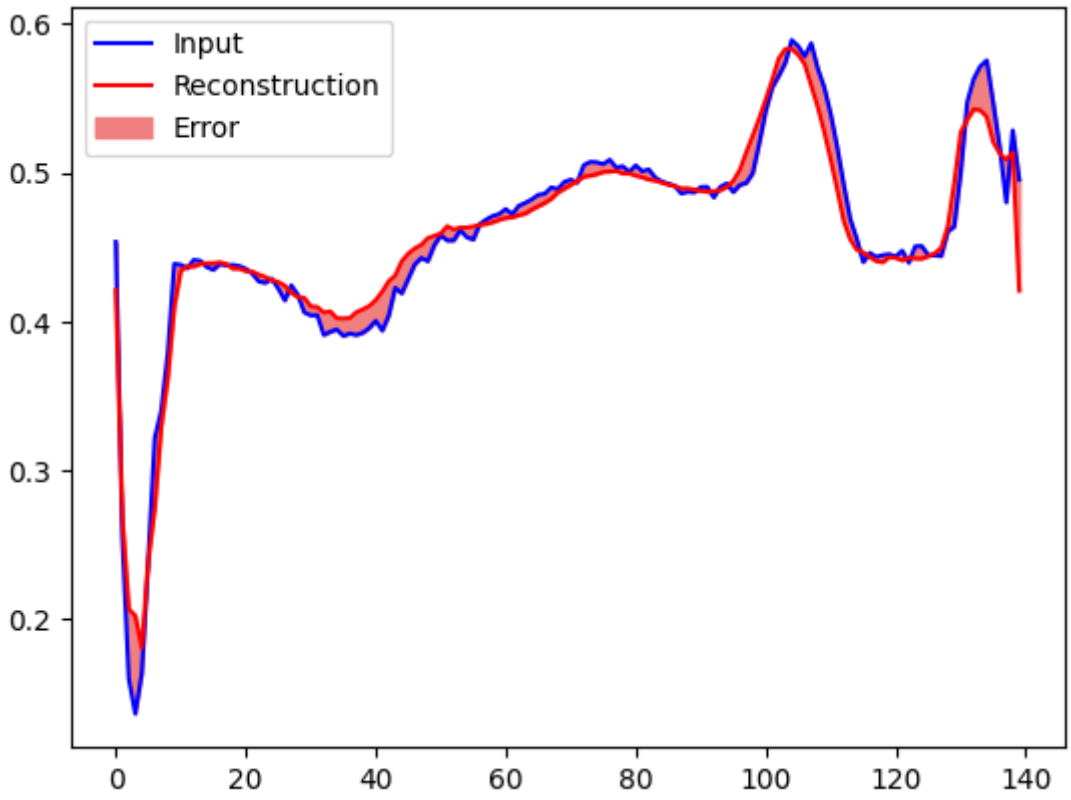
```
pred = prediction(autoencoder, n_test_data, t)
print(pred)
```

```
tf.Tensor(
[False  True  True False  True  True  True False  True  True  True  True
  True  True  True  True  True  True  True  True False  True  True  True
  True  True  True  True  True  True  True  True  True  True False  True
  True  True  True False  True  True  True  True  True  True  True  True
  True  True  True  True False  True  True  True  True  True  True  True
  True  True  True  True False  True  True  True  True  True  True  True
  True  True  True  True  True  True  True False  True  True  True  True
 False  True  True  True  True  True  True  True  True  True  True  True
  True  True  True  True  True  True False  True  True  True  True  True
  True  True  True  True  True  True  True  True  True  True  True  True
  True False  True  True  True  True  True  True  True  True  True  True
  True  True  True  True  True  True  True False  True  True  True  True
  True  True  True  True  True  True  True  True  True  True  True  True
  True  True  True  True  True  True  True  True  True  True  True  True
  True  True  True  True  True  True  True  True False  True  True  True
  True  True  True  True  True  True  True  True  True  True  True  True
  True  True  True  True  True  True  True  True  True  True  True  True
  True  True  True  True  True  True False False  True  True  True  True
  True  True  True  True False  True  True  True  True  True  True  True
  True  True  True  True  True  True  True  True  True  True False False
  True  True  True  True  True  True  True  True  True  True  True  True
  True  True False  True  True  True  True  True  True  True  True  True
  True  True  True  True  True False  True  True  True  True  True  True
  True  True  True  True  True  True  True  True  True  True  True  True
  True  True False  True  True  True  True  True  True  True  True  True
  True  True  True False  True  True  True  True  True  True False  True
  True  True  True  True  True  True  True  True  True  True  True  True
  True  True  True  True  True  True False  True  True False  True  True
  True  True  True  True  True  True  True  True  True  True  True  True
  True  True  True  True  True  True  True  True  True  True  True  True
  True False  True False  True  True  True  True  True  True  True  True
  True  True  True  True  True  True  True  True  True  True  True  True
  True  True  True  True  True  True  True  True False  True  True  True
  True  True  True  True  True  True  True  True  True  True  True False
 False  True  True  True False  True  True  True  True  True  True  True
  True  True  True False False False  True False False  True False  True
 False False  True  True  True  True  True  True  True  True  True  True
  True  True  True  True  True  True  True  True  True  True  True  True
  True  True  True False  True  True  True  True  True  True  True  True
  True  True  True  True  True  True  True  True  True  True  True  True
  True False  True False  True  True  True False False False  True False
  True  True  True  True  True  True  True  True  True  True  True  True
  True  True  True  True  True  True  True  True  True  True  True  True
  True False  True  True  True False  True  True], shape=(560,), dtype=boo
l)
```

In [14]:

```
# Darstellung weiterer Ergebnisse
plot(n_test_data, 0)
plot(n_test_data, 1)
plot(n_test_data, 3)
```





Quellen: https://www.kaggle.com/code/devavratatripathy/ecg-anomaly-detection-using-autoencoders (https://www.kaggle.com/code/devavratatripathy/ecg-anomaly-detection-using-autoencoders)