In [1]:
```
import matplotlib.pyplot as plt
import pandas as pd
import tensorflow as tf
import seaborn as sns
from tensorflow.keras.models import Model
from sklearn.metrics import accuracy_score, precision_score, recall_score
from sklearn.model_selection import train_test_split
from keras import Sequential
sns.set
import numpy as np
from tensorflow.keras import layers, losses
```

In [2]:
```
df = pd.read_csv('http://storage.googleapis.com/download.tensorflow.org/data/ecg.

raw_data = df.values

df.head(10)
```

Out[2]:

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|---|
| **0** | -0.112522 | -2.827204 | -3.773897 | -4.349751 | -4.376041 | -3.474986 | -2.181408 | -1.818286 | -1.250522 |
| **1** | -1.100878 | -3.996840 | -4.285843 | -4.506579 | -4.022377 | -3.234368 | -1.566126 | -0.992258 | -0.754680 |
| **2** | -0.567088 | -2.593450 | -3.874230 | -4.584095 | -4.187449 | -3.151462 | -1.742940 | -1.490659 | -1.183580 |
| **3** | 0.490473 | -1.914407 | -3.616364 | -4.318823 | -4.268016 | -3.881110 | -2.993280 | -1.671131 | -1.333884 |
| **4** | 0.800232 | -0.874252 | -2.384761 | -3.973292 | -4.338224 | -3.802422 | -2.534510 | -1.783423 | -1.594450 |
| **5** | -1.507674 | -3.574550 | -4.478011 | -4.408275 | -3.321242 | -2.105171 | -1.481048 | -1.301362 | -0.498240 |
| **6** | -0.297161 | -2.766635 | -4.102185 | -4.589669 | -4.219357 | -3.650443 | -2.300518 | -1.293917 | -1.065658 |
| **7** | 0.446769 | -1.507397 | -3.187468 | -4.507462 | -4.604201 | -3.636115 | -2.311604 | -1.597727 | -1.362450 |
| **8** | 0.087631 | -1.753490 | -3.304473 | -4.704657 | -4.686415 | -3.611817 | -2.267268 | -1.570893 | -1.417790 |
| **9** | -0.832281 | -1.700368 | -2.257301 | -2.853671 | -2.853301 | -2.701487 | -2.285726 | -1.555512 | -1.266622 |

10 rows × 141 columns

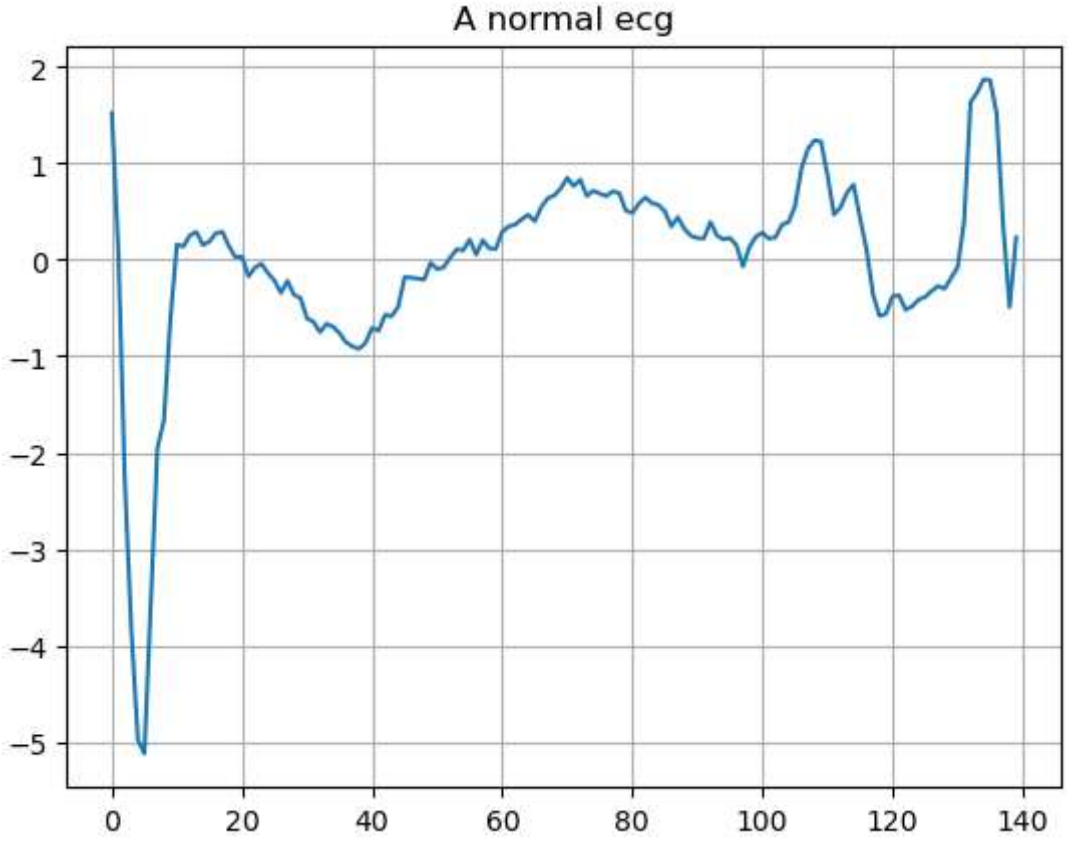
In [3]:
```
labels = raw_data[:, -1]

data = raw_data[:, 0:-1]

pd.Series(labels).value_counts()
train_data,test_data, train_labels,test_labels = train_test_split(data, labels, t
```

In [4]:
```
train_labels = train_labels.astype(bool)
test_labels = test_labels.astype(bool)
normal_train_data = train_data[train_labels]
normal_test_data = test_data[test_labels]
anamalous_train_data = train_data[~train_labels]
anamalous_test_data = test_data[~test_labels]
```

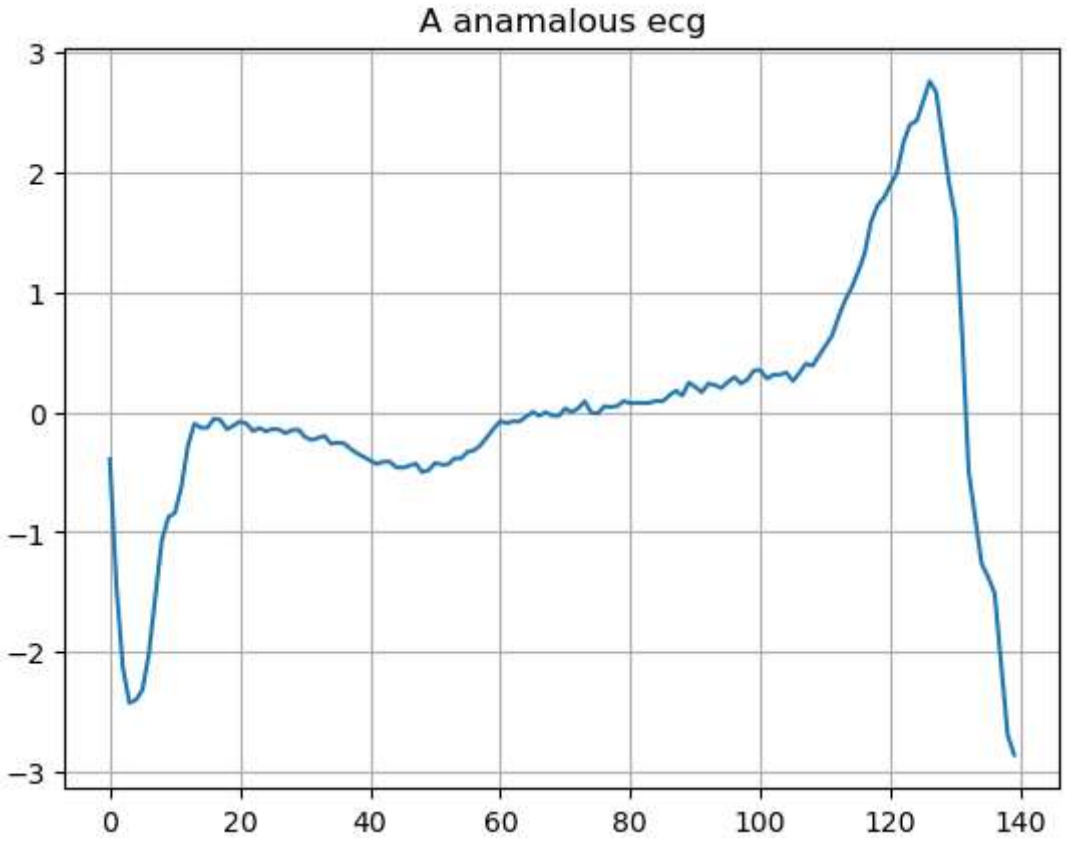
In [5]:
```
~train_labels
```

Out[5]:
```
array([False, False, False, ..., False, False, False])
```

In [6]:
```
plt.grid()
plt.plot(np.arange(140),normal_train_data[0])
plt.title('A normal ecg')
plt.show()
```



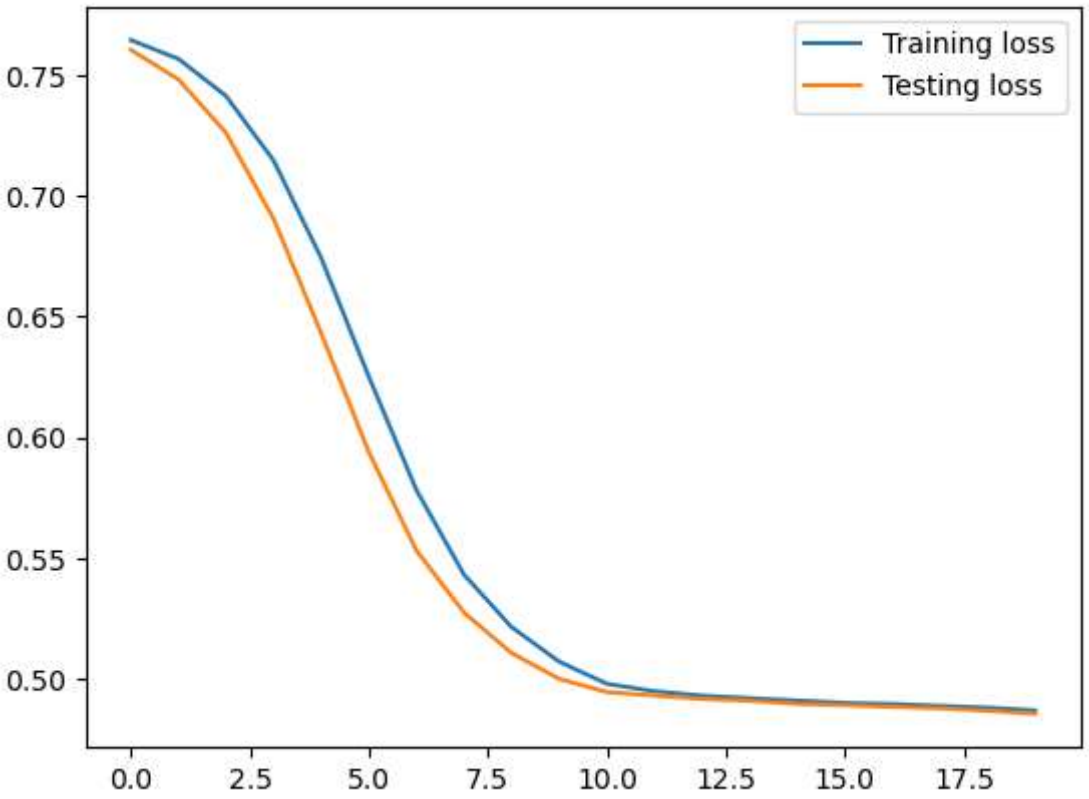
In [7]:
```
plt.grid()
plt.plot(np.arange(140),anamalous_train_data[0])
plt.title('A anamalous ecg')
plt.show()
```



In [8]:
```
class AnomalyDetector(Model):
  def __init__(self):
    super(AnomalyDetector,self).__init__()
    self.encoder=Sequential([
        layers.Dense(32,activation='relu'),
        layers.Dense(16,activation='relu'),
        layers.Dense(8,activation='relu')

    ])
    self.decoder=tf.keras.Sequential([
        layers.Dense(32,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

autoencoder=AnomalyDetector()
autoencoder.compile(optimizer='adam',loss='mae')
history=autoencoder.fit(normal_train_data,normal_train_data,
                        epochs=20,
                        batch_size=512,
                        validation_data=(normal_test_data,normal_test_data),
                        shuffle=True)
plt.plot(history.history['loss'],label='Training loss')
plt.plot(history.history['val_loss'],label='Testing loss')
plt.legend()
plt.show()
```

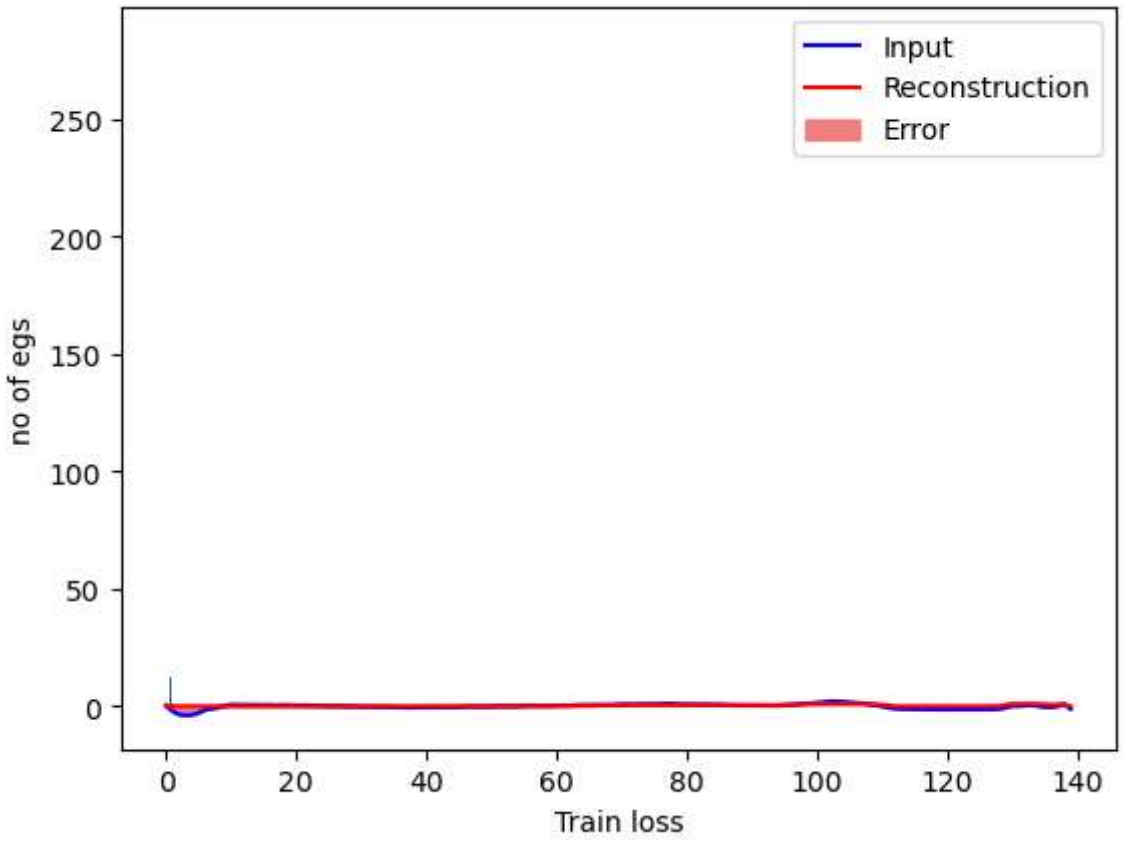
```
Epoch 1/20
5/5 [==============================] - 4s 51ms/step - loss: 0.7644 - val_loss:
0.7603
Epoch 2/20
5/5 [==============================] - 0s 13ms/step - loss: 0.7566 - val_loss:
0.7481
Epoch 3/20
5/5 [==============================] - 0s 11ms/step - loss: 0.7413 - val_loss:
0.7260
Epoch 4/20
5/5 [==============================] - 0s 13ms/step - loss: 0.7145 - val_loss:
0.6904
Epoch 5/20
5/5 [==============================] - 0s 13ms/step - loss: 0.6742 - val_loss:
0.6430
Epoch 6/20
5/5 [==============================] - 0s 14ms/step - loss: 0.6250 - val_loss:
0.5938
Epoch 7/20
5/5 [==============================] - 0s 14ms/step - loss: 0.5781 - val_loss:
0.5531
Epoch 8/20
5/5 [==============================] - 0s 14ms/step - loss: 0.5431 - val_loss:
```

```
0.5275
Epoch 9/20
5/5 [==============================] - 0s 13ms/step - loss: 0.5214 - val_loss:
0.5107
Epoch 10/20
5/5 [==============================] - 0s 13ms/step - loss: 0.5071 - val_loss:
0.5000
Epoch 11/20
5/5 [==============================] - 0s 13ms/step - loss: 0.4980 - val_loss:
0.4946
Epoch 12/20
5/5 [==============================] - 0s 11ms/step - loss: 0.4949 - val_loss:
0.4931
Epoch 13/20
5/5 [==============================] - 0s 19ms/step - loss: 0.4931 - val_loss:
0.4917
Epoch 14/20
5/5 [==============================] - 0s 12ms/step - loss: 0.4920 - val_loss:
0.4910
Epoch 15/20
5/5 [==============================] - 0s 13ms/step - loss: 0.4910 - val_loss:
0.4897
Epoch 16/20
5/5 [==============================] - 0s 13ms/step - loss: 0.4900 - val_loss:
0.4891
Epoch 17/20
5/5 [==============================] - 0s 12ms/step - loss: 0.4896 - val_loss:
0.4884
Epoch 18/20
5/5 [==============================] - 0s 13ms/step - loss: 0.4888 - val_loss:
0.4879
Epoch 19/20
5/5 [==============================] - 0s 11ms/step - loss: 0.4881 - val_loss:
0.4868
Epoch 20/20
5/5 [==============================] - 0s 11ms/step - loss: 0.4870 - val_loss:
0.4856
```


Training loss
Testing loss
0.75
0.70
0.65
0.60
0.55
0.50
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5

In [9]:
```
encoded_image=autoencoder.encoder(anamalous_test_data).numpy()
decoded_image=autoencoder.decoder(encoded_image).numpy()

plt.plot(normal_test_data[0],'b')
plt.plot(decoded_image[0],'r')
plt.fill_between(np.arange(140),decoded_image[0],normal_test_data[0],color='light
plt.legend(labels=['Input','Reconstruction','Error'])

reconstructions=autoencoder.predict(normal_train_data)
train_loss=tf.keras.losses.mae(reconstructions,normal_train_data)
plt.hist(train_loss[None,:],bins=50)
plt.xlabel('Train loss')
plt.ylabel('no of egs')
plt.show()
```

```
74/74 [==============================] - 0s 2ms/step
```

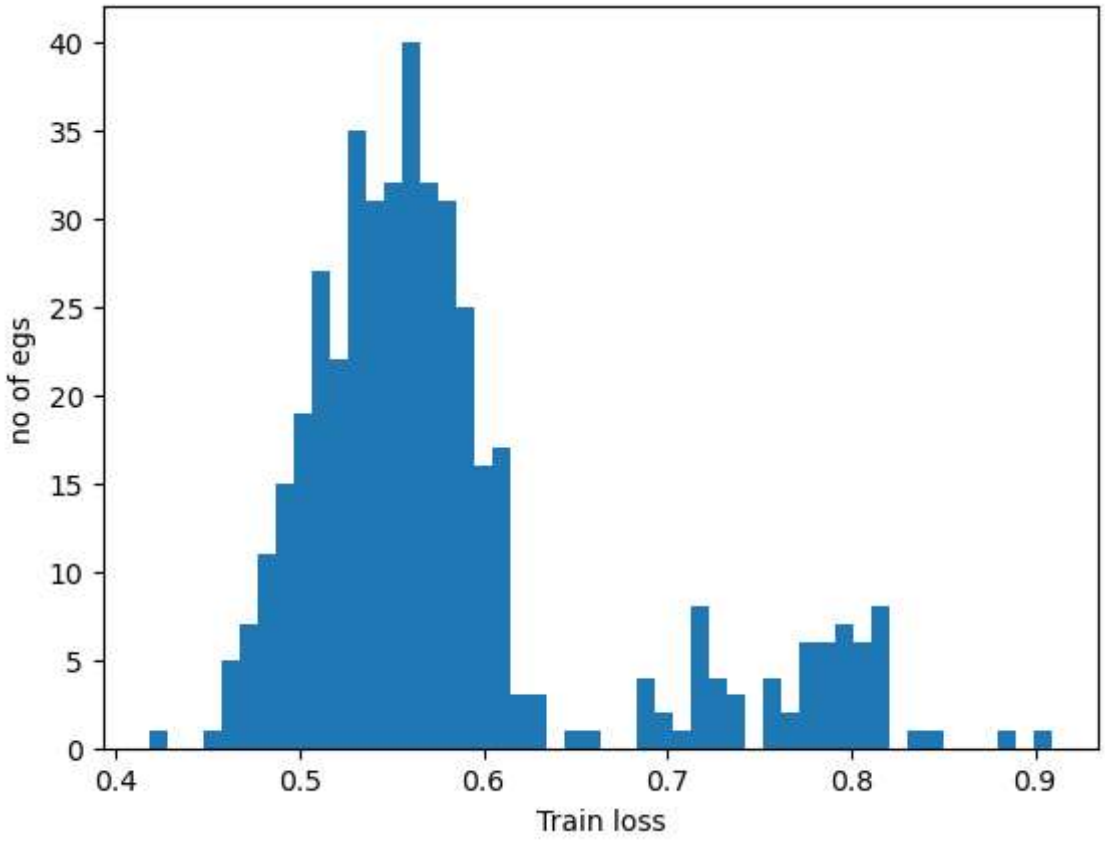



In [10]:
```
threshold=np.mean(train_loss)+np.std(train_loss)
print("threshold:",threshold)
```

```
threshold: 0.5604083278695905
```

In [11]:
```
reconstructions=autoencoder.predict(anamalous_test_data)
test_loss=tf.keras.losses.mae(reconstructions,anamalous_test_data)

plt.hist(test_loss[None,:],bins=50)
plt.xlabel('Train loss')
plt.ylabel('no of egs')
plt.show()
```

```
14/14 [==============================] - 0s 2ms/step
```



In [12]:
```
def predict(model,data,threshold):
  reconstructions=model(data)
  loss=tf.keras.losses.mae(reconstructions,data)
  return tf.math.less(loss,threshold)

def print_stats(predictions,labels):
  print("accuracy={}".format(accuracy_score(labels,preds)))
  print("precision={}".format(precision_score(labels,preds)))
  print("recall={}".format(recall_score(labels,preds)))

preds=predict(autoencoder,test_data,threshold)
print_stats(preds,test_labels)
```

```
accuracy=0.708
precision=0.6835616438356165
recall=0.8910714285714286
```

In [ ]:
```
```