Experiment No.4

Title : Use Autoencoder to implement anomaly detection

In [19]:

```
import numpy as np
import pandas as pd
import tensorflow as tf
import matplotlib.pyplot as plt
from sklearn.metrics import accuracy_score
from tensorflow.keras.optimizers import Adam
from sklearn.preprocessing import MinMaxScaler
from tensorflow.keras import Model, Sequential
from tensorflow.keras.layers import Dense, Dropout
from sklearn.model_selection import train_test_split
from tensorflow.keras.losses import MeanSquaredLogarithmicError
```

Load the data

In [2]:

```
# Download the dataset
PATH_TO_DATA = 'http://storage.googleapis.com/download.tensorflow.org/data/ecg.csv'
data = pd.read_csv(PATH_TO_DATA, header=None)
print(data.shape)
data.head()
```

```
(4998, 141)
```

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	-1.250
1	-1.100878	-3.996840	-4.285843	-4.506579	-4.022377	-3.234368	-1.566126	-0.992258	-0.754
2	-0.567088	-2.593450	-3.874230	-4.584095	-4.187449	-3.151462	-1.742940	-1.490659	-1.183
3	0.490473	-1.914407	-3.616364	-4.318823	-4.268016	-3.881110	-2.993280	-1.671131	-1.333
4	0.800232	-0.874252	-2.384761	-3.973292	-4.338224	-3.802422	-2.534510	-1.783423	-1.594

5 rows × 141 columns

Split the data for training and testing

In [3]:

```
# last column is the target
# 0 = anomaly, 1 = normal
TARGET = 140

features = data.drop(TARGET, axis=1)
target = data[TARGET]

x_train, x_test, y_train, y_test = train_test_split(
    features, target, test_size=0.2, stratify=target
)
```

Scale the data using MinMaxScaler

In [4]:

```
# use case is novelty detection so use only the normal data
# for training
train_index = y_train[y_train == 1].index
train_data = x_train.loc[train_index]
```

In [20]:

```
min_max_scaler = MinMaxScaler(feature_range=(0, 1))
x_train_scaled = min_max_scaler.fit_transform(train_data.copy())
x_test_scaled = min_max_scaler.transform(x_test.copy())
```

Build an AutoEncoder model

In [6]:

```
# create a model by subclassing Model class in tensorflow
class AutoEncoder(Model):
  """
  Parameters
  ----------
  output_units: int
    Number of output units

  code_size: int
    Number of units in bottle neck
  """

  def __init__(self, output_units, code_size=8):
    super().__init__()
    self.encoder = Sequential([
      Dense(64, activation='relu'),
      Dropout(0.1),
      Dense(32, activation='relu'),
      Dropout(0.1),
      Dense(16, activation='relu'),
      Dropout(0.1),
      Dense(code_size, activation='relu')
    ])
    self.decoder = Sequential([
      Dense(16, activation='relu'),
      Dropout(0.1),
      Dense(32, activation='relu'),
      Dropout(0.1),
      Dense(64, activation='relu'),
      Dropout(0.1),
      Dense(output_units, activation='sigmoid')
    ])

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

In [7]:

```
model = AutoEncoder(output_units=x_train_scaled.shape[1])
# configurations of model
model.compile(loss='msle', metrics=['mse'], optimizer='adam')

history = model.fit(
    x_train_scaled,
    x_train_scaled,
    epochs=20,
    batch_size=512,
    validation_data=(x_test_scaled, x_test_scaled)
)
```

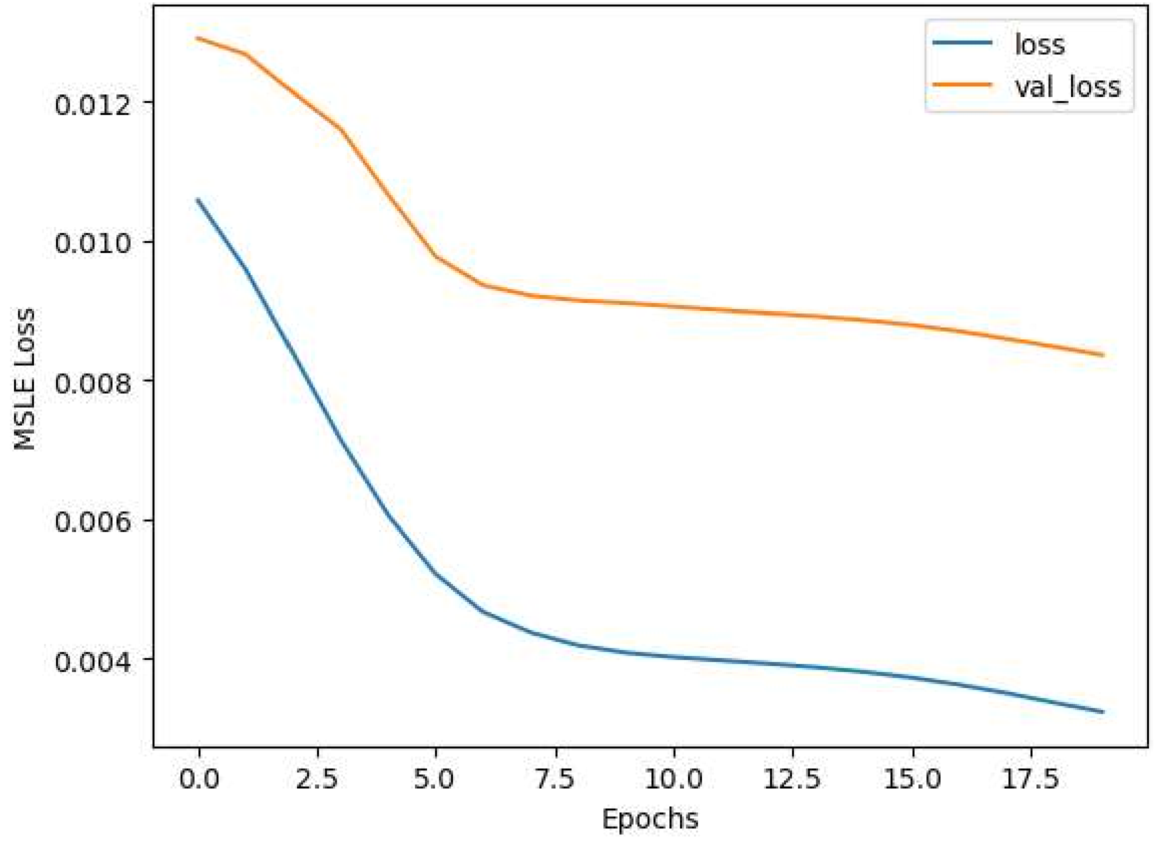
```
Epoch 1/20
5/5 ━━━━━━━━━━━━━━━━━━━━ 2s 38ms/step - loss: 0.0107 - mse: 0.0241 - val_los
s: 0.0129 - val_mse: 0.0298
Epoch 2/20
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 0.0097 - mse: 0.0218 - val_los
s: 0.0127 - val_mse: 0.0292
Epoch 3/20
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 0.0085 - mse: 0.0192 - val_los
s: 0.0121 - val_mse: 0.0279
Epoch 4/20
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 0.0073 - mse: 0.0165 - val_los
s: 0.0116 - val_mse: 0.0267
Epoch 5/20
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step - loss: 0.0062 - mse: 0.0140 - val_los
s: 0.0107 - val_mse: 0.0246
Epoch 6/20
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 0.0053 - mse: 0.0119 - val_los
s: 0.0098 - val_mse: 0.0226
Epoch 7/20
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 0.0047 - mse: 0.0106 - val_los
s: 0.0094 - val_mse: 0.0217
Epoch 8/20
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step - loss: 0.0044 - mse: 0.0100 - val_los
s: 0.0092 - val_mse: 0.0214
Epoch 9/20
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step - loss: 0.0042 - mse: 0.0093 - val_los
s: 0.0091 - val_mse: 0.0212
Epoch 10/20
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step - loss: 0.0042 - mse: 0.0094 - val_los
s: 0.0091 - val_mse: 0.0211
Epoch 11/20
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 0.0040 - mse: 0.0090 - val_los
s: 0.0091 - val_mse: 0.0210
Epoch 12/20
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 0.0039 - mse: 0.0087 - val_los
s: 0.0090 - val_mse: 0.0209
Epoch 13/20
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step - loss: 0.0040 - mse: 0.0088 - val_los
s: 0.0090 - val_mse: 0.0208
Epoch 14/20
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 0.0038 - mse: 0.0085 - val_los
s: 0.0089 - val_mse: 0.0207
Epoch 15/20
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step - loss: 0.0038 - mse: 0.0085 - val_los
s: 0.0089 - val_mse: 0.0206
Epoch 16/20
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step - loss: 0.0037 - mse: 0.0084 - val_los
```

```
s: 0.0088 - val_mse: 0.0204
Epoch 17/20
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 0.0036 - mse: 0.0081 - val_los
s: 0.0087 - val_mse: 0.0202
Epoch 18/20
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step - loss: 0.0035 - mse: 0.0079 - val_los
s: 0.0086 - val_mse: 0.0200
Epoch 19/20
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step - loss: 0.0033 - mse: 0.0075 - val_los
s: 0.0085 - val_mse: 0.0197
Epoch 20/20
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step - loss: 0.0033 - mse: 0.0073 - val_los
s: 0.0084 - val_mse: 0.0194
```

Plot history

In [8]:

```
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.xlabel('Epochs')
plt.ylabel('MSLE Loss')
plt.legend(['loss', 'val_loss'])
plt.show()
```



Find threshold

In [9]:

```
def find_threshold(model, x_train_scaled):
  reconstructions = model.predict(x_train_scaled)
  # provides losses of individual instances
  reconstruction_errors = tf.keras.losses.msle(reconstructions, x_train_scaled)

  # threshold for anomaly scores
  threshold = np.mean(reconstruction_errors.numpy()) \
      + np.std(reconstruction_errors.numpy())
  return threshold

def find_threshold_method_two(model, x_train_scaled):
  # another method to find threshold
  reconstructions = model.predict(x_train_scaled)
  # provides losses of individual instances
  reconstruction_errors = tf.keras.losses.msle(reconstructions, x_train_scaled)

  threshold_2 = np.percentile(reconstruction_errors, 95)
  return threshold_2

def get_predictions(model, x_test_scaled, threshold):
  predictions = model.predict(x_test_scaled)
  # provides losses of individual instances
  errors = tf.keras.losses.msle(predictions, x_test_scaled)
  # 0 = anomaly, 1 = normal
  anomaly_mask = pd.Series(errors) > threshold
  preds = anomaly_mask.map(lambda x: 0.0 if x == True else 1.0)
  return preds
```

In [10]:

```
threshold = find_threshold(model, x_train_scaled)
print(f"Threshold method one: {threshold}")

threshold_2 = find_threshold_method_two(model, x_train_scaled)
print(f"Threshold method two: {threshold_2}")
```

```
73/73 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step
Threshold method one: 0.007880838475657386
73/73 ━━━━━━━━━━━━━━━━━━━━ 0s 653us/step
Threshold method two: 0.011669582318263243
```

In [11]:

```
preds = get_predictions(model, x_test_scaled, threshold)
accuracy_score(preds, y_test)
```

```
32/32 ━━━━━━━━━━━━━━━━━━━━ 0s 678us/step
```

Out[11]:

```
0.953
```

Tuning AutoEncoder using keras tuner

In [12]:

```
!pip install -U keras-tuner
```

```
Collecting keras-tuner
  Obtaining dependency information for keras-tuner from https://files.pyth
onhosted.org/packages/db/5d/945296512980b0827e93418514c8be9236baa6f0a1e8ca
8be3a2026665b0/keras_tuner-1.4.7-py3-none-any.whl.metadata (https://files.
pythonhosted.org/packages/db/5d/945296512980b0827e93418514c8be9236baa6f0a1
e8ca8be3a2026665b0/keras_tuner-1.4.7-py3-none-any.whl.metadata)
  Downloading keras_tuner-1.4.7-py3-none-any.whl.metadata (5.4 kB)
Requirement already satisfied: keras in c:\users\admin\anaconda3\lib\site-
packages (from keras-tuner) (3.4.1)
Requirement already satisfied: packaging in c:\users\admin\anaconda3\lib\s
ite-packages (from keras-tuner) (23.1)
Requirement already satisfied: requests in c:\users\admin\anaconda3\lib\si
te-packages (from keras-tuner) (2.31.0)
Collecting kt-legacy (from keras-tuner)
  Obtaining dependency information for kt-legacy from https://files.python
hosted.org/packages/16/53/aca9f36da2516db008017db85a1f3cafaee0efc5fc7a25d9
4c909651792f/kt_legacy-1.0.5-py3-none-any.whl.metadata (https://files.pyth
onhosted.org/packages/16/53/aca9f36da2516db008017db85a1f3cafaee0efc5fc7a25
d94c909651792f/kt_legacy-1.0.5-py3-none-any.whl.metadata)
  Downloading kt_legacy-1.0.5-py3-none-any.whl.metadata (221 bytes)
Requirement already satisfied: absl-py in c:\users\admin\anaconda3\lib\sit
e-packages (from keras->keras-tuner) (2.1.0)
Requirement already satisfied: numpy in c:\users\admin\anaconda3\lib\site-
packages (from keras->keras-tuner) (1.24.3)
Requirement already satisfied: rich in c:\users\admin\anaconda3\lib\site-p
ackages (from keras->keras-tuner) (13.7.1)
Requirement already satisfied: namex in c:\users\admin\anaconda3\lib\site-
packages (from keras->keras-tuner) (0.0.8)
Requirement already satisfied: h5py in c:\users\admin\anaconda3\lib\site-p
ackages (from keras->keras-tuner) (3.11.0)
Requirement already satisfied: optree in c:\users\admin\anaconda3\lib\site
-packages (from keras->keras-tuner) (0.12.1)
Requirement already satisfied: ml-dtypes in c:\users\admin\anaconda3\lib\s
ite-packages (from keras->keras-tuner) (0.4.0)
Requirement already satisfied: charset-normalizer<4,>=2 in c:\users\admin
\anaconda3\lib\site-packages (from requests->keras-tuner) (2.0.4)
Requirement already satisfied: idna<4,>=2.5 in c:\users\admin\anaconda3\li
b\site-packages (from requests->keras-tuner) (3.4)
Requirement already satisfied: urllib3<3,>=1.21.1 in c:\users\admin\anacon
da3\lib\site-packages (from requests->keras-tuner) (1.26.16)
Requirement already satisfied: certifi>=2017.4.17 in c:\users\admin\anacon
da3\lib\site-packages (from requests->keras-tuner) (2023.7.22)
Requirement already satisfied: typing-extensions>=4.5.0 in c:\users\admin
\anaconda3\lib\site-packages (from optree->keras->keras-tuner) (4.12.2)
Requirement already satisfied: markdown-it-py>=2.2.0 in c:\users\admin\ana
conda3\lib\site-packages (from rich->keras->keras-tuner) (2.2.0)
Requirement already satisfied: pygments<3.0.0,>=2.13.0 in c:\users\admin\a
naconda3\lib\site-packages (from rich->keras->keras-tuner) (2.15.1)
Requirement already satisfied: mdurl~=0.1 in c:\users\admin\anaconda3\lib
\site-packages (from markdown-it-py>=2.2.0->rich->keras->keras-tuner) (0.
1.0)
Downloading keras_tuner-1.4.7-py3-none-any.whl (129 kB)
   ---------------------------------------- 0.0/129.1 kB ? eta -:--:--
   --- ------------------------------------ 10.2/129.1 kB ? eta -:--:--
   ---------------------------------------- 129.1/129.1 kB 1.9 MB/s eta 0:
00:00
Downloading kt_legacy-1.0.5-py3-none-any.whl (9.6 kB)
```

```
Installing collected packages: kt-legacy, keras-tuner
Successfully installed keras-tuner-1.4.7 kt-legacy-1.0.5
```

In [13]:

```
import kerastuner as kt

class AutoEncoderTuner(Model):

  def __init__(self, hp, output_units, code_size=8):
    super().__init__()
    dense_1_units = hp.Int('dense_1_units', min_value=16, max_value=72, step=4)
    dense_2_units = hp.Int('dense_2_units', min_value=16, max_value=72, step=4)
    dense_3_units = hp.Int('dense_3_units', min_value=16, max_value=72, step=4)
    dense_4_units = hp.Int('dense_4_units', min_value=16, max_value=72, step=4)
    dense_5_units = hp.Int('dense_5_units', min_value=16, max_value=72, step=4)
    dense_6_units = hp.Int('dense_6_units', min_value=16, max_value=72, step=4)

    self.encoder = Sequential([
      Dense(dense_1_units, activation='relu'),
      Dropout(0.1),
      Dense(dense_2_units, activation='relu'),
      Dropout(0.1),
      Dense(dense_3_units, activation='relu'),
      Dropout(0.1),
      Dense(code_size, activation='relu')
    ])
    self.decoder = Sequential([
      Dense(dense_4_units, activation='relu'),
      Dropout(0.1),
      Dense(dense_5_units, activation='relu'),
      Dropout(0.1),
      Dense(dense_6_units, activation='relu'),
      Dropout(0.1),
      Dense(output_units, activation='sigmoid')
    ])

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


def build_model(hp):
  model = AutoEncoderTuner(hp, 140)
  hp_learning_rate = hp.Choice('learning_rate', values=[1e-2, 1e-3, 1e-4])
  model.compile(
      loss='msle',
      optimizer=Adam(learning_rate=hp_learning_rate),
  )
  return model
```

```
C:\Users\ADMIN\AppData\Local\Temp\ipykernel_8352\3309659638.py:1: Deprecatio
nWarning: `import kerastuner` is deprecated, please use `import keras_tuner
`.
  import kerastuner as kt
```

In [14]:

```
tuner = kt.Hyperband(
    build_model,
    objective='val_loss',
    max_epochs=20,
    factor=3,
    directory='autoencoder',
    project_name='tuning_autoencoder6'
)

tuner.search(
    x_train_scaled,
    x_train_scaled,
    epochs=20,
    batch_size=512,
    validation_data=(x_test_scaled, x_test_scaled)
)
```

```
Trial 30 Complete [00h 00m 04s]
val_loss: 0.008870153687894344

Best val_loss So Far: 0.007670378312468529
Total elapsed time: 00h 01m 17s
```

In [15]:

```
hparams = [f'dense_{i}_units' for i in range(1,7)] + ['learning_rate']
best_hyperparams = tuner.get_best_hyperparameters()
for hps in hparams:
  print(f"{hps}: {best_hyperparams[0][hps]}")
```

```
dense_1_units: 64
dense_2_units: 48
dense_3_units: 28
dense_4_units: 48
dense_5_units: 48
dense_6_units: 68
learning_rate: 0.01
```

In [16]:

```
best_model = tuner.get_best_models()[0]
best_model.compile(loss='msle', optimizer=Adam(0.001))

best_model.fit(
    x_train_scaled,
    x_train_scaled,
    epochs=20,
    batch_size=512,
    validation_data=(x_test_scaled, x_test_scaled)
)
```

```
Epoch 1/20
5/5 ━━━━━━━━━━━━━━━━━━━━ 1s 32ms/step - loss: 0.0022 - val_loss: 0.0076
Epoch 2/20
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 0.0022 - val_loss: 0.0076
Epoch 3/20
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step - loss: 0.0021 - val_loss: 0.0075
Epoch 4/20
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 0.0021 - val_loss: 0.0075
Epoch 5/20
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 0.0021 - val_loss: 0.0074
Epoch 6/20
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 0.0020 - val_loss: 0.0073
Epoch 7/20
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 0.0020 - val_loss: 0.0072
Epoch 8/20
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - loss: 0.0020 - val_loss: 0.0071
Epoch 9/20
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - loss: 0.0020 - val_loss: 0.0071
Epoch 10/20
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step - loss: 0.0019 - val_loss: 0.0070
Epoch 11/20
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 0.0019 - val_loss: 0.0069
Epoch 12/20
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 0.0018 - val_loss: 0.0069
Epoch 13/20
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 0.0018 - val_loss: 0.0068
Epoch 14/20
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 0.0018 - val_loss: 0.0067
Epoch 15/20
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step - loss: 0.0018 - val_loss: 0.0066
Epoch 16/20
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 0.0018 - val_loss: 0.0066
Epoch 17/20
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 0.0018 - val_loss: 0.0066
Epoch 18/20
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 0.0017 - val_loss: 0.0065
Epoch 19/20
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 0.0017 - val_loss: 0.0065
Epoch 20/20
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 0.0017 - val_loss: 0.0064
```

Out[16]:

```
<keras.src.callbacks.history.History at 0x25a4eb62290>
```

In [17]:

```
threshold_ = find_threshold(best_model, x_train_scaled)
preds_ = get_predictions(best_model, x_test_scaled, threshold_)
accuracy_score(preds_, y_test)
```

```
73/73 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step
32/32 ━━━━━━━━━━━━━━━━━━━━ 0s 710us/step
```

Out[17]:

```
0.968
```