```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import matplotlib as mpl
import tensorflow as tf
from tensorflow.keras.models import Model
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import MinMaxScaler, StandardScaler
mpl.rcParams['figure.figsize'] = (10, 5)
mpl.rcParams['axes.grid'] = False
```

```
!cat "/content/ECG5000_TRAIN.txt" "/content/ECG5000_TEST.txt" > ecg_final.txt
df = pd.read_csv("/content/ecg_final.txt", sep='  ', header=None)
df.shape
```

```
<ipython-input-8-56c4ed0545ed>:2: ParserWarning: Falling back to the 'python' engine because the 'c' engine does not support
  df = pd.read_csv("/content/ecg_final.txt", sep='  ', header=None)
(5000, 141)
```

```
df = df.add_prefix('c')
df['c0'].value_counts()
```

```
1.0    2919
2.0    1767
4.0     194
3.0      96
5.0      24
Name: c0, dtype: int64
```

```
df.head()
```

	c0	c1	c2	c3	c4	c5	c6	c7	c8
0	1.0	-0.112522	-2.827204	-3.773897	-4.349751	-4.376041	-3.474986	-2.181408	-1.818286
1	1.0	-1.100878	-3.996840	-4.285843	-4.506579	-4.022377	-3.234368	-1.566126	-0.992258
2	1.0	-0.567088	-2.593450	-3.874230	-4.584095	-4.187449	-3.151462	-1.742940	-1.490659
3	1.0	0.490473	-1.914407	-3.616364	-4.318823	-4.268016	-3.881110	-2.993280	-1.671131
4	1.0	0.800232	-0.874252	-2.384761	-3.973292	-4.338224	-3.802422	-2.534510	-1.783423

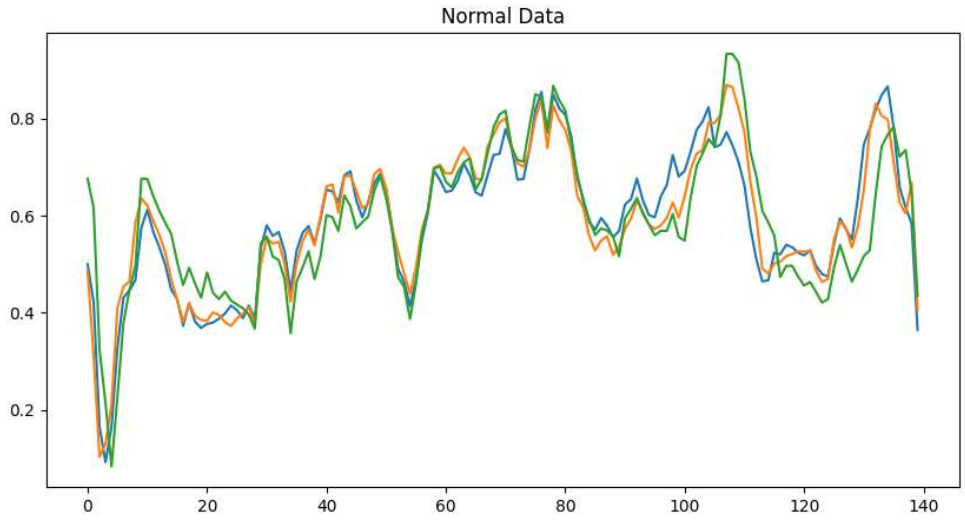
5 rows × 141 columns

```
x_train, x_test, y_train, y_test = train_test_split(df.values, df.values[:,0:1], test_size=0.2, random_state=111)
```

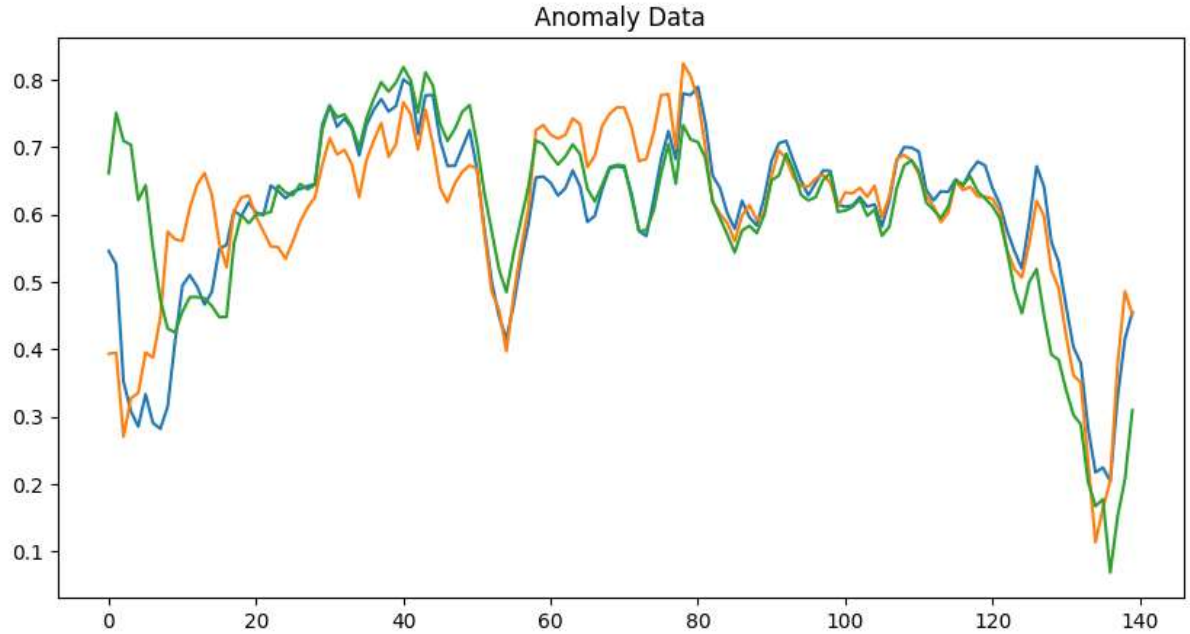
```
scaler = MinMaxScaler()
data_scaled = scaler.fit(x_train)
train_data_scaled = data_scaled.transform(x_train)
test_data_scaled = data_scaled.transform(x_test)
```

```
normal_train_data = pd.DataFrame(train_data_scaled).add_prefix('c').query('c0 == 0').values[:,1:]
anomaly_train_data = pd.DataFrame(train_data_scaled).add_prefix('c').query('c0 > 0').values[:, 1:]
normal_test_data = pd.DataFrame(test_data_scaled).add_prefix('c').query('c0 == 0').values[:,1:]
anomaly_test_data = pd.DataFrame(test_data_scaled).add_prefix('c').query('c0 > 0').values[:, 1:]
```

```
plt.plot(normal_train_data[0])
plt.plot(normal_train_data[1])
plt.plot(normal_train_data[2])
plt.title("Normal Data")
plt.show()
```




```
plt.plot(anomaly_train_data[0])
plt.plot(anomaly_train_data[1])
plt.plot(anomaly_train_data[2])
plt.title("Anomaly Data")
plt.show()
```




```
model = tf.keras.Sequential()
model.add(tf.keras.layers.Dense(64, activation="relu"))
model.add(tf.keras.layers.Dense(32, activation="relu"))
model.add(tf.keras.layers.Dense(16, activation="relu"))
model.add(tf.keras.layers.Dense(8, activation="relu"))
model.add(tf.keras.layers.Dense(16, activation="relu"))
model.add(tf.keras.layers.Dense(32, activation="relu"))
model.add(tf.keras.layers.Dense(64, activation="relu"))
model.add(tf.keras.layers.Dense(140, activation="sigmoid"))
```

```
class AutoEncoder(Model):
  def __init__(self):
    super(AutoEncoder, self).__init__()
```

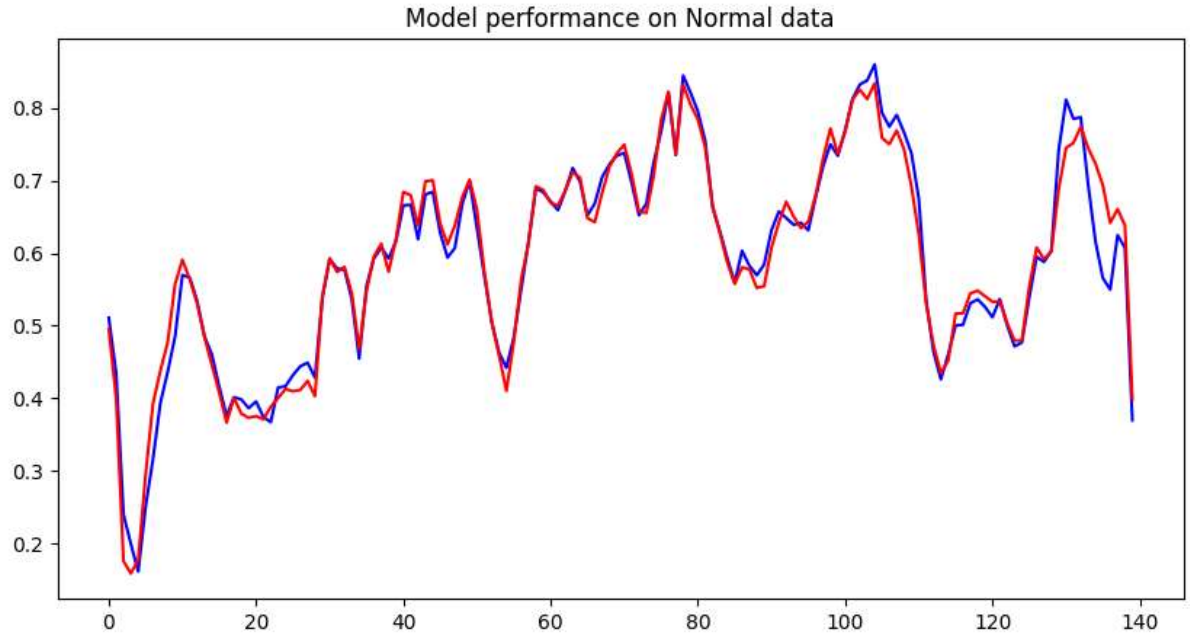
```
        self.encoder = tf.keras.Sequential([
                      tf.keras.layers.Dense(64, activation="relu"),
                      tf.keras.layers.Dense(32, activation="relu"),
                      tf.keras.layers.Dense(16, activation="relu"),
                      tf.keras.layers.Dense(8, activation="relu")
                  ])
        self.decoder = tf.keras.Sequential([
                      tf.keras.layers.Dense(16, activation="relu"),
                      tf.keras.layers.Dense(32, activation="relu"),
                      tf.keras.layers.Dense(64, activation="relu"),
                      tf.keras.layers.Dense(140, activation="sigmoid")
                  ])
      def call(self, x):
        encoded = self.encoder(x)
        decoded = self.decoder(encoded)
        return decoded


    model = AutoEncoder()
    early_stopping = tf.keras.callbacks.EarlyStopping(monitor="val_loss", patience=2, mode="min")
    model.compile(optimizer='adam', loss="mae")
    history = model.fit(normal_train_data, normal_train_data, epochs=50, batch_size=120,
                        validation_data=(train_data_scaled[:,1:], train_data_scaled[:, 1:]),
                        shuffle=True,
                        callbacks=[early_stopping]
                        )
```

```
Epoch 1/50
20/20 [==============================] - 4s 37ms/step - loss: 0.1219 - val_loss: 0.1064
Epoch 2/50
20/20 [==============================] - 0s 15ms/step - loss: 0.0725 - val_loss: 0.0801
Epoch 3/50
20/20 [==============================] - 0s 20ms/step - loss: 0.0508 - val_loss: 0.0756
Epoch 4/50
20/20 [==============================] - 0s 25ms/step - loss: 0.0479 - val_loss: 0.0748
Epoch 5/50
20/20 [==============================] - 0s 17ms/step - loss: 0.0474 - val_loss: 0.0743
Epoch 6/50
20/20 [==============================] - 0s 14ms/step - loss: 0.0467 - val_loss: 0.0738
Epoch 7/50
20/20 [==============================] - 1s 27ms/step - loss: 0.0454 - val_loss: 0.0714
Epoch 8/50
20/20 [==============================] - 0s 18ms/step - loss: 0.0423 - val_loss: 0.0672
Epoch 9/50
20/20 [==============================] - 0s 12ms/step - loss: 0.0387 - val_loss: 0.0638
Epoch 10/50
20/20 [==============================] - 0s 15ms/step - loss: 0.0375 - val_loss: 0.0630
Epoch 11/50
20/20 [==============================] - 0s 23ms/step - loss: 0.0371 - val_loss: 0.0629
Epoch 12/50
20/20 [==============================] - 0s 6ms/step - loss: 0.0368 - val_loss: 0.0625
Epoch 13/50
20/20 [==============================] - 0s 6ms/step - loss: 0.0367 - val_loss: 0.0623
Epoch 14/50
20/20 [==============================] - 0s 8ms/step - loss: 0.0364 - val_loss: 0.0620
Epoch 15/50
20/20 [==============================] - 0s 8ms/step - loss: 0.0362 - val_loss: 0.0619
Epoch 16/50
20/20 [==============================] - 0s 7ms/step - loss: 0.0360 - val_loss: 0.0614
Epoch 17/50
20/20 [==============================] - 0s 12ms/step - loss: 0.0358 - val_loss: 0.0608
Epoch 18/50
20/20 [==============================] - 0s 11ms/step - loss: 0.0357 - val_loss: 0.0609
Epoch 19/50
20/20 [==============================] - 0s 10ms/step - loss: 0.0354 - val_loss: 0.0602
Epoch 20/50
20/20 [==============================] - 0s 13ms/step - loss: 0.0352 - val_loss: 0.0602
Epoch 21/50
20/20 [==============================] - 0s 11ms/step - loss: 0.0351 - val_loss: 0.0599
Epoch 22/50
20/20 [==============================] - 0s 10ms/step - loss: 0.0350 - val_loss: 0.0598
Epoch 23/50
20/20 [==============================] - 0s 15ms/step - loss: 0.0350 - val_loss: 0.0598
Epoch 24/50
20/20 [==============================] - 0s 15ms/step - loss: 0.0349 - val_loss: 0.0593
Epoch 25/50
```

```
20/20 [==============================] - 0s 11ms/step - loss: 0.0346 - val_loss: 0.0591
Epoch 26/50
20/20 [==============================] - 0s 14ms/step - loss: 0.0345 - val_loss: 0.0592
Epoch 27/50
20/20 [==============================] - 0s 7ms/step - loss: 0.0343 - val_loss: 0.0584
Epoch 28/50
20/20 [==============================] - 0s 7ms/step - loss: 0.0343 - val_loss: 0.0590
Epoch 29/50
20/20 [==============================] - 0s 8ms/step - loss: 0.0340 - val_loss: 0.0583
```

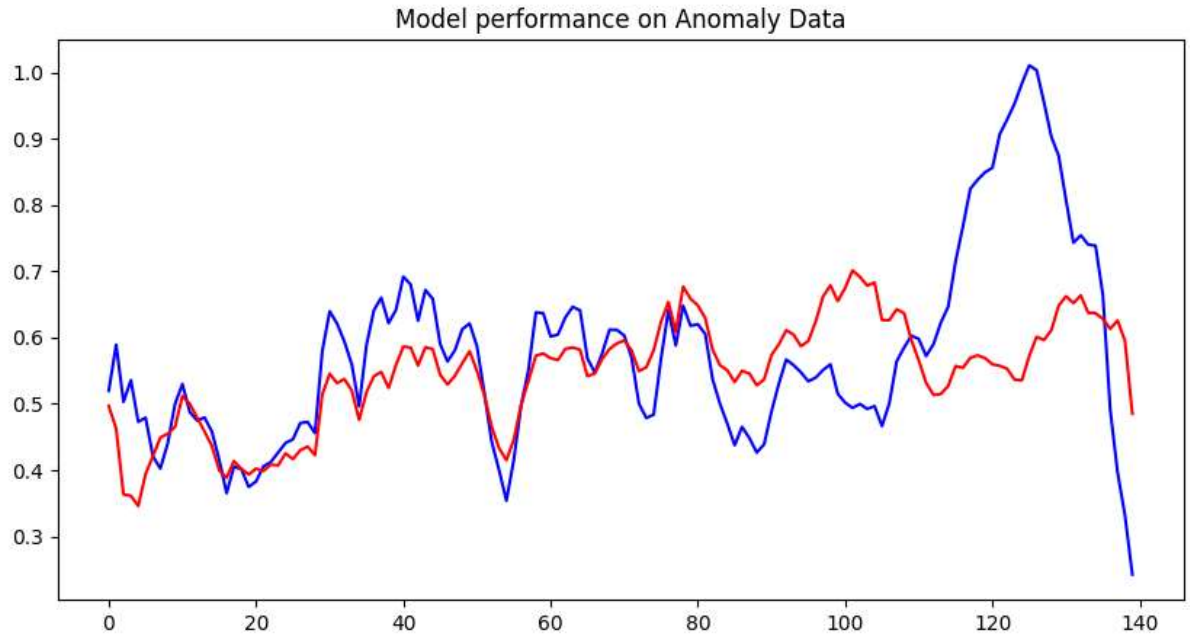
```
encoder_out = model.encoder(normal_test_data).numpy() #8 unit representation of data
decoder_out = model.decoder(encoder_out).numpy()
```

```
plt.plot(normal_test_data[0], 'b')
plt.plot(decoder_out[0], 'r')
plt.title("Model performance on Normal data")
plt.show()
```



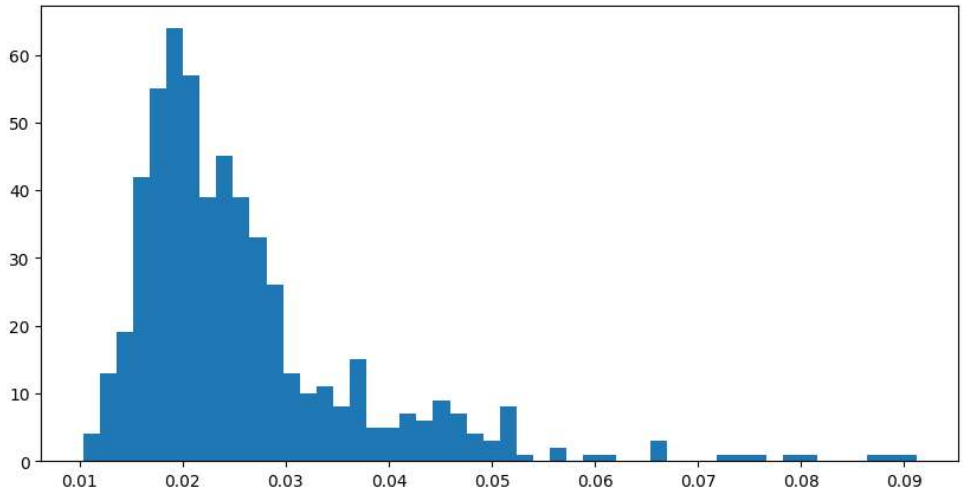

```
encoder_out_a = model.encoder(anomaly_test_data).numpy() #8 unit representation of data
decoder_out_a = model.decoder(encoder_out_a).numpy()
```

```
plt.plot(anomaly_test_data[0], 'b')
plt.plot(decoder_out_a[0], 'r')
plt.title("Model performance on Anomaly Data")
plt.show()
```




```
#calculate loss
reconstruction = model.predict(normal_test_data)
train_loss = tf.keras.losses.mae(reconstruction, normal_test_data)
plt.hist(train_loss, bins=50)
```

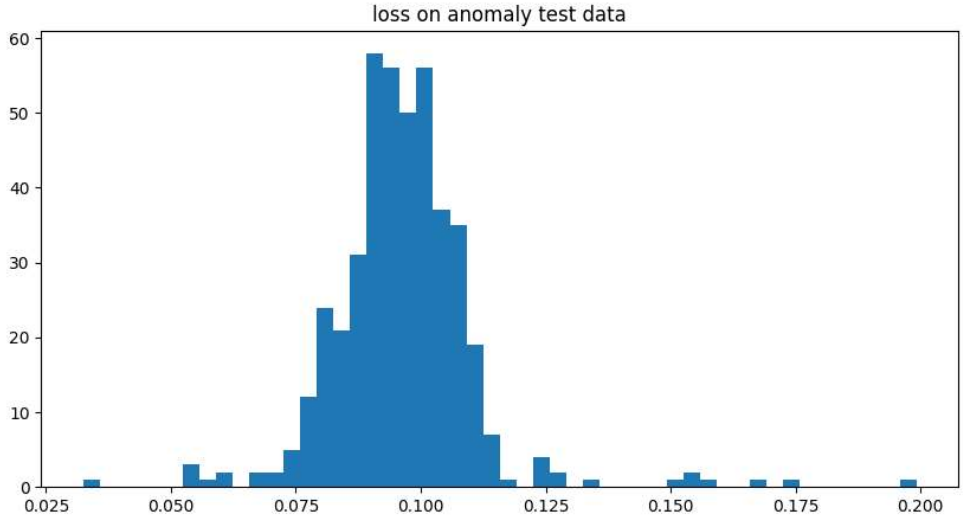
```
18/18 [==============================] - 0s 2ms/step
(array([ 4., 13., 19., 42., 55., 64., 57., 39., 45., 39., 33., 26., 13.,
        10., 11.,  8., 15.,  5.,  5.,  7.,  6.,  9.,  7.,  4.,  3.,  8.,
         1.,  0.,  2.,  0.,  1.,  1.,  0.,  0.,  3.,  0.,  0.,  0.,  1.,
         1.,  1.,  0.,  1.,  1.,  0.,  0.,  0.,  1.,  1.,  1.]),
 array([0.01031468, 0.01193354, 0.01355239, 0.01517124, 0.0167901 ,
        0.01840895, 0.02002781, 0.02164666, 0.02326551, 0.02488437,
        0.02650322, 0.02812207, 0.02974093, 0.03135978, 0.03297863,
        0.03459749, 0.03621634, 0.03783519, 0.03945405, 0.0410729 ,
        0.04269176, 0.04431061, 0.04592946, 0.04754832, 0.04916717,
        0.05078602, 0.05240488, 0.05402373, 0.05564258, 0.05726144,
        0.05888029, 0.06049914, 0.062118  , 0.06373685, 0.06535571,
        0.06697456, 0.06859341, 0.07021227, 0.07183112, 0.07344997,
        0.07506883, 0.07668768, 0.07830653, 0.07992539, 0.08154424,
        0.08316309, 0.08478195, 0.0864008 , 0.08801965, 0.08963851,
        0.09125736]),
 <BarContainer object of 50 artists>)
```



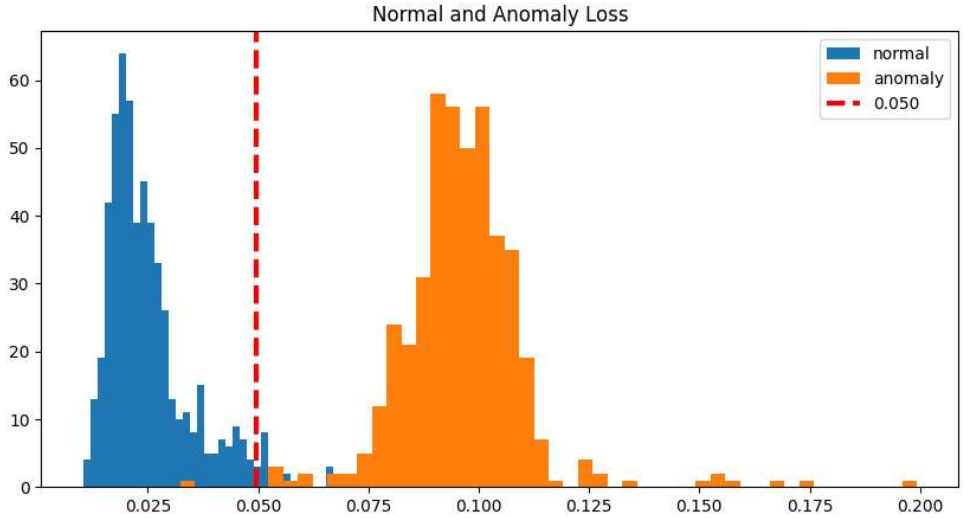
```
threshold = np.mean(train_loss) + 2*np.std(train_loss)
reconstruction_a = model.predict(anomaly_test_data)
train_loss_a = tf.keras.losses.mae(reconstruction_a, anomaly_test_data)
```

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

```
plt.hist(train_loss_a, bins=50)
plt.title("loss on anomaly test data")
plt.show()
```




```
plt.hist(train_loss, bins=50, label='normal')
plt.hist(train_loss_a, bins=50, label='anomaly')
plt.axvline(threshold, color='r', linewidth=3, linestyle='dashed', label='{:0.3f}'.format(threshold))
plt.legend(loc='upper right')
plt.title("Normal and Anomaly Loss")
plt.show()
```




```
preds = tf.math.less(train_loss, threshold)
tf.math.count_nonzero(preds)
```

```
<tf.Tensor: shape=(), dtype=int64, numpy=537>
```

```
preds_a = tf.math.greater(train_loss_a, threshold)
tf.math.count_nonzero(preds_a)
```

```
<tf.Tensor: shape=(), dtype=int64, numpy=436>
```

Start coding or generate with AI.