```
pip install tensorflow
```

```
    Requirement already satisfied: tensorflow in /usr/local/lib/python3.10/dist-packages (2.15.0)
    Requirement already satisfied: absl-py>=1.0.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (1.4.0)
    Requirement already satisfied: astunparse>=1.6.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (1.6.3
    Requirement already satisfied: flatbuffers>=23.5.26 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (23
    Requirement already satisfied: gast!=0.5.0,!=0.5.1,!=0.5.2,>=0.2.1 in /usr/local/lib/python3.10/dist-packages (from
    Requirement already satisfied: google-pasta>=0.1.1 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (0.2
    Requirement already satisfied: h5py>=2.9.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (3.9.0)
    Requirement already satisfied: libclang>=13.0.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (16.0.6
    Requirement already satisfied: ml-dtypes~=0.2.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (0.2.0)
    Requirement already satisfied: numpy<2.0.0,>=1.23.5 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (1.
    Requirement already satisfied: opt-einsum>=2.3.2 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (3.3.0
    Requirement already satisfied: packaging in /usr/local/lib/python3.10/dist-packages (from tensorflow) (23.2)
    Requirement already satisfied: protobuf!=4.21.0,!=4.21.1,!=4.21.2,!=4.21.3,!=4.21.4,!=4.21.5,<5.0.0dev,>=3.20.3 in /
    Requirement already satisfied: setuptools in /usr/local/lib/python3.10/dist-packages (from tensorflow) (67.7.2)
    Requirement already satisfied: six>=1.12.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (1.16.0)
    Requirement already satisfied: termcolor>=1.1.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (2.4.0)
    Requirement already satisfied: typing-extensions>=3.6.6 in /usr/local/lib/python3.10/dist-packages (from tensorflow)
    Requirement already satisfied: wrapt<1.15,>=1.11.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (1.1
    Requirement already satisfied: tensorflow-io-gcs-filesystem>=0.23.1 in /usr/local/lib/python3.10/dist-packages (from
    Requirement already satisfied: grpcio<2.0,>=1.24.3 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (1.6
    Requirement already satisfied: tensorboard<2.16,>=2.15 in /usr/local/lib/python3.10/dist-packages (from tensorflow)
    Requirement already satisfied: tensorflow-estimator<2.16,>=2.15.0 in /usr/local/lib/python3.10/dist-packages (from t
    Requirement already satisfied: keras<2.16,>=2.15.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (2.1
    Requirement already satisfied: wheel<1.0,>=0.23.0 in /usr/local/lib/python3.10/dist-packages (from astunparse>=1.6.0
    Requirement already satisfied: google-auth<3,>=1.6.3 in /usr/local/lib/python3.10/dist-packages (from tensorboard<2.
    Requirement already satisfied: google-auth-oauthlib<2,>=0.5 in /usr/local/lib/python3.10/dist-packages (from tensorb
    Requirement already satisfied: markdown>=2.6.8 in /usr/local/lib/python3.10/dist-packages (from tensorboard<2.16,>=2
    Requirement already satisfied: requests<3,>=2.21.0 in /usr/local/lib/python3.10/dist-packages (from tensorboard<2.16
    Requirement already satisfied: tensorboard-data-server<0.8.0,>=0.7.0 in /usr/local/lib/python3.10/dist-packages (fro
    Requirement already satisfied: werkzeug>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from tensorboard<2.16,>=2
    Requirement already satisfied: cachetools<6.0,>=2.0.0 in /usr/local/lib/python3.10/dist-packages (from google-auth<3
    Requirement already satisfied: pyasn1-modules>=0.2.1 in /usr/local/lib/python3.10/dist-packages (from google-auth<3,
    Requirement already satisfied: rsa<5,>=3.1.4 in /usr/local/lib/python3.10/dist-packages (from google-auth<3,>=1.6.3-
    Requirement already satisfied: requests-oauthlib>=0.7.0 in /usr/local/lib/python3.10/dist-packages (from google-auth
    Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.10/dist-packages (from requests<3,
    Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-packages (from requests<3,>=2.21.0->te
    Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.10/dist-packages (from requests<3,>=2.21
```

```
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.10/dist-packages (from requests<3,>=2.21
Requirement already satisfied: MarkupSafe>=2.1.1 in /usr/local/lib/python3.10/dist-packages (from werkzeug>=1.0.1->t
Requirement already satisfied: pyasn1<0.6.0,>=0.4.6 in /usr/local/lib/python3.10/dist-packages (from pyasn1-modules>
Requirement already satisfied: oauthlib>=3.0.0 in /usr/local/lib/python3.10/dist-packages (from requests-oauthlib>=0
```

```
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

# Load and Combine the dataset
!cat "ECG5000_TRAIN.txt" "ECG5000_TEST.txt" > ecg_final.txt
df = pd.read_csv("ecg_final.txt", sep='  ', header=None)

df.head()
```

```
<ipython-input-34-5ed4a34550af>:12: ParserWarning: Falling back to the 'python' engine because the 'c' engine does n
  df = pd.read_csv("ecg_final.txt", sep='  ', header=None)
```

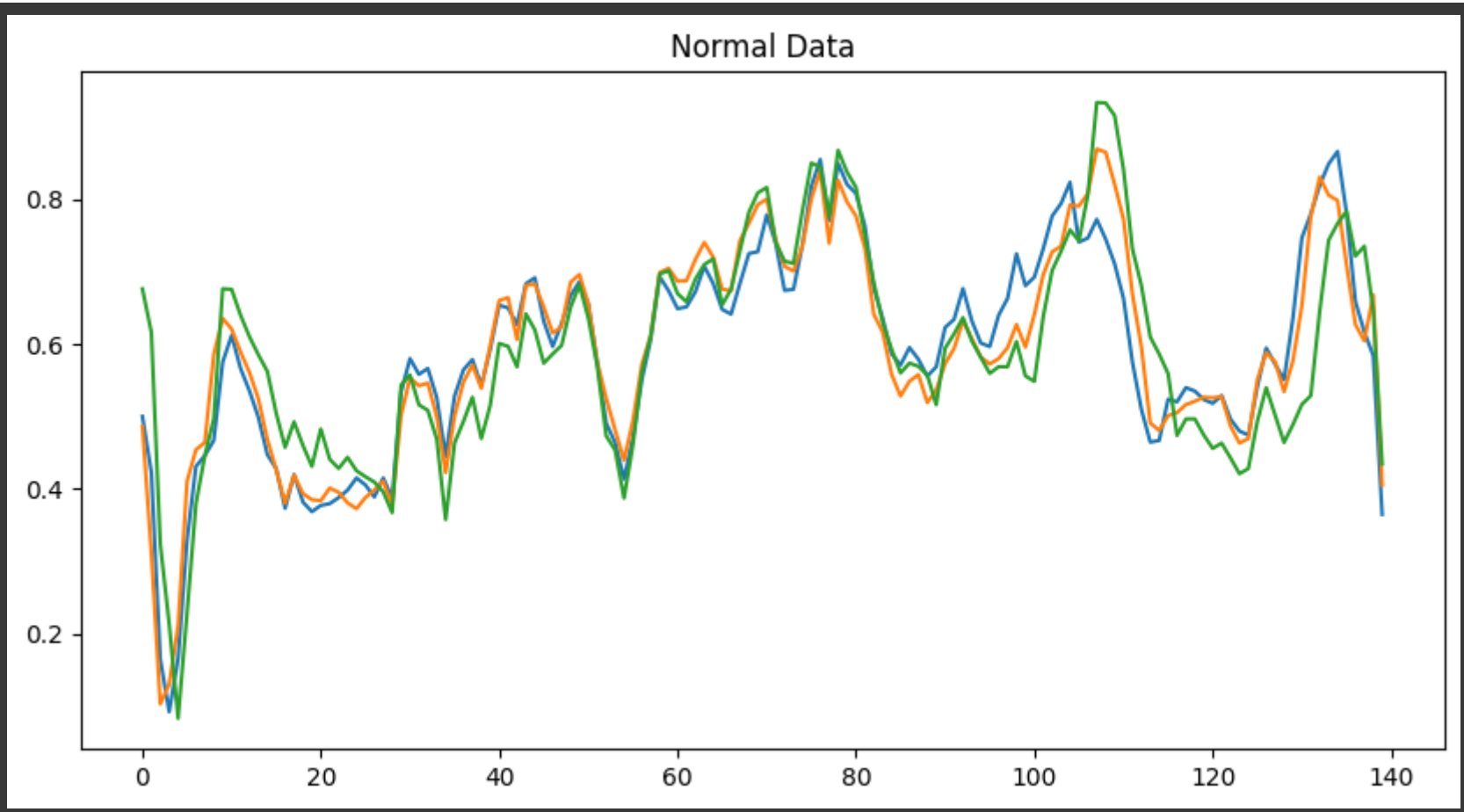
	0	1	2	3	4	5	6	7	8	9	...	131	132	1
0	1.0	-0.112522	-2.827204	-3.773897	-4.349751	-4.376041	-3.474986	-2.181408	-1.818286	-1.250522	...	0.160348	0.792168	0.933
1	1.0	-1.100878	-3.996840	-4.285843	-4.506579	-4.022377	-3.234368	-1.566126	-0.992258	-0.754680	...	0.560327	0.538356	0.656
2	1.0	-0.567088	-2.593450	-3.874230	-4.584095	-4.187449	-3.151462	-1.742940	-1.490659	-1.183580	...	1.284825	0.886073	0.531
3	1.0	0.490473	-1.914407	-3.616364	-4.318823	-4.268016	-3.881110	-2.993280	-1.671131	-1.333884	...	0.491173	0.350816	0.499
4	1.0	0.800232	-0.874252	-2.384761	-3.973292	-4.338224	-3.802422	-2.534510	-1.783423	-1.594450	...	0.966606	1.148884	0.958

5 rows × 141 columns

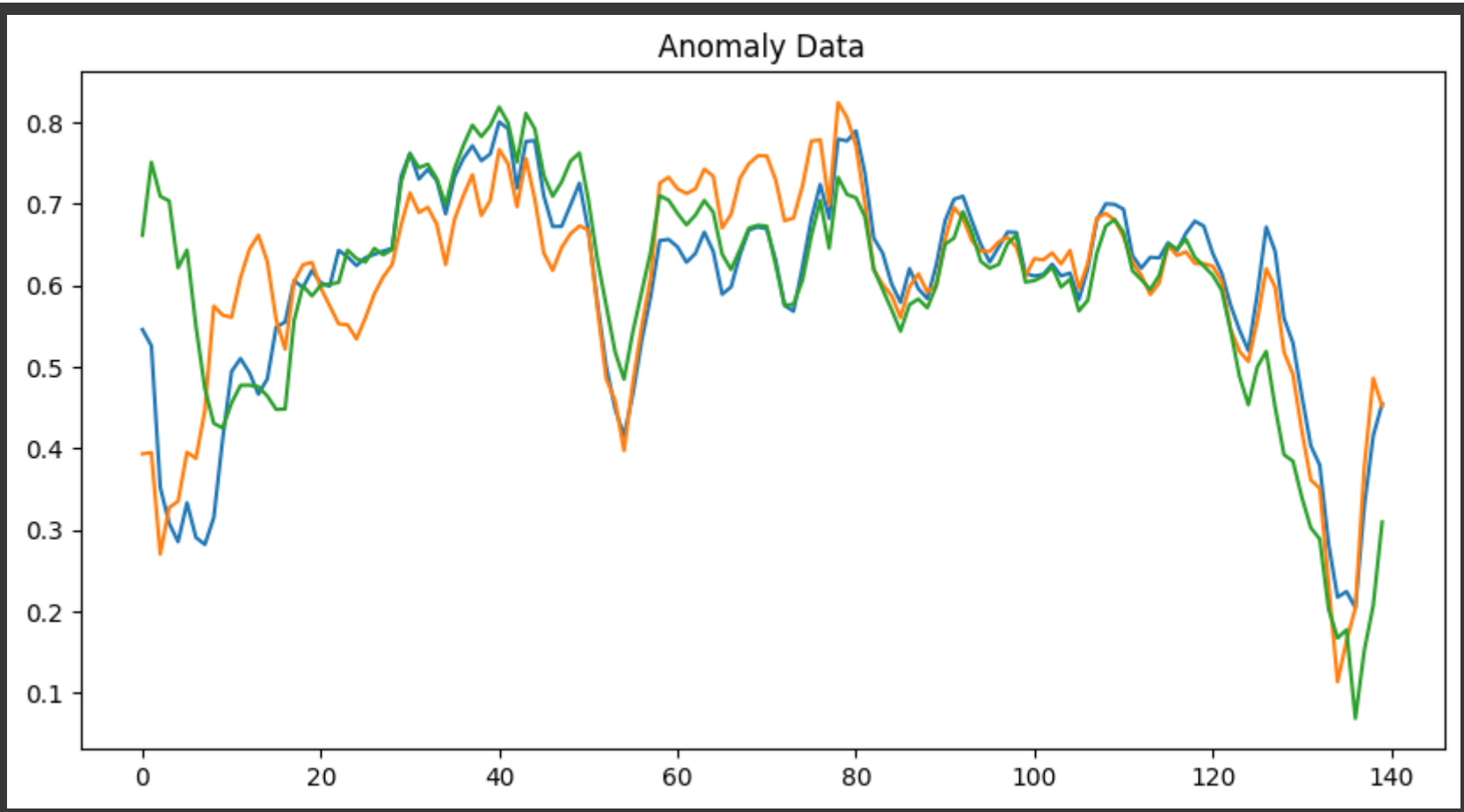
```
# Train-Test Splitting and Scaling
x_train, x_test, y_train, y_test = train_test_split(df.values, df.values[:,0:1], test_size=0.2, random_state=111)
scaler = MinMaxScaler()
data_scaled = scaler.fit(x_train)
train_data_scaled = data_scaled.transform(x_train)
test_data_scaled = data_scaled.transform(x_test)
```

```
# Separate Anomaly and Normal Data
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:]

# Data Visualization
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()
```


Anomaly Data
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
20
40
60
80
100
120
140

```
# Modelling
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

# Compile and train the Model
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]
                    )
```

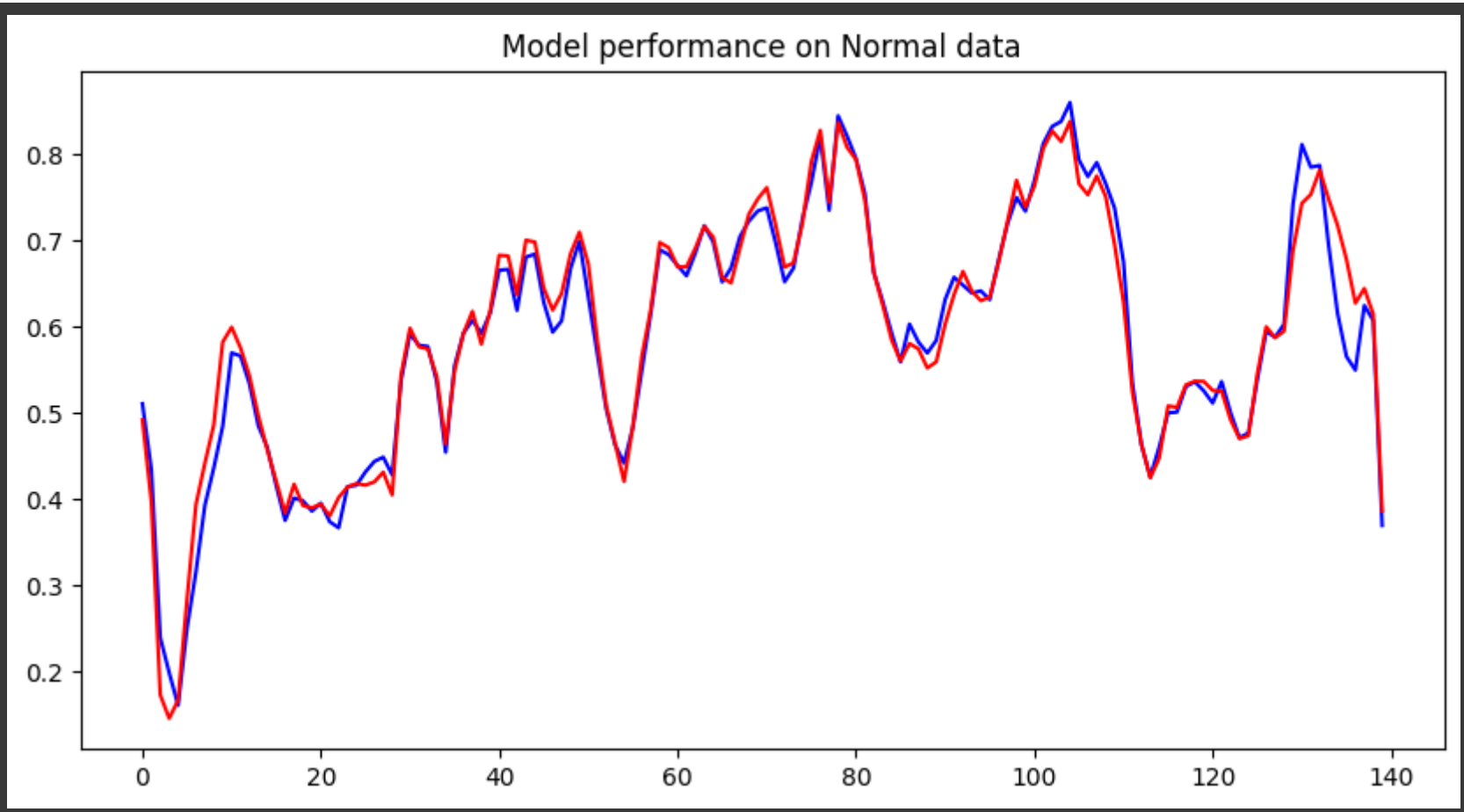
```
Epoch 1/50
20/20 [==============================] - 2s 25ms/step - loss: 0.1288 - val_loss: 0.1118
Epoch 2/50
20/20 [==============================] - 0s 15ms/step - loss: 0.0788 - val_loss: 0.0811
Epoch 3/50
20/20 [==============================] - 0s 12ms/step - loss: 0.0522 - val_loss: 0.0773
```

```
Epoch 4/50
20/20 [==============================] - 0s 12ms/step - loss: 0.0484 - val_loss: 0.0760
Epoch 5/50
20/20 [==============================] - 0s 12ms/step - loss: 0.0479 - val_loss: 0.0756
Epoch 6/50
20/20 [==============================] - 0s 14ms/step - loss: 0.0477 - val_loss: 0.0750
Epoch 7/50
20/20 [==============================] - 0s 15ms/step - loss: 0.0473 - val_loss: 0.0741
Epoch 8/50
20/20 [==============================] - 0s 16ms/step - loss: 0.0468 - val_loss: 0.0731
Epoch 9/50
20/20 [==============================] - 0s 13ms/step - loss: 0.0452 - val_loss: 0.0697
Epoch 10/50
20/20 [==============================] - 0s 9ms/step - loss: 0.0411 - val_loss: 0.0643
Epoch 11/50
20/20 [==============================] - 0s 9ms/step - loss: 0.0378 - val_loss: 0.0619
Epoch 12/50
20/20 [==============================] - 0s 13ms/step - loss: 0.0366 - val_loss: 0.0617
Epoch 13/50
20/20 [==============================] - 0s 13ms/step - loss: 0.0362 - val_loss: 0.0608
Epoch 14/50
20/20 [==============================] - 0s 9ms/step - loss: 0.0358 - val_loss: 0.0608
Epoch 15/50
20/20 [==============================] - 0s 9ms/step - loss: 0.0356 - val_loss: 0.0611
Epoch 16/50
20/20 [==============================] - 0s 10ms/step - loss: 0.0356 - val_loss: 0.0608
Epoch 17/50
20/20 [==============================] - 0s 13ms/step - loss: 0.0353 - val_loss: 0.0602
Epoch 18/50
20/20 [==============================] - 0s 13ms/step - loss: 0.0351 - val_loss: 0.0601
Epoch 19/50
20/20 [==============================] - 0s 13ms/step - loss: 0.0349 - val_loss: 0.0599
Epoch 20/50
20/20 [==============================] - 0s 8ms/step - loss: 0.0348 - val_loss: 0.0595
Epoch 21/50
20/20 [==============================] - 0s 9ms/step - loss: 0.0346 - val_loss: 0.0592
Epoch 22/50
20/20 [==============================] - 0s 8ms/step - loss: 0.0343 - val_loss: 0.0589
Epoch 23/50
20/20 [==============================] - 0s 9ms/step - loss: 0.0339 - val_loss: 0.0583
Epoch 24/50
```

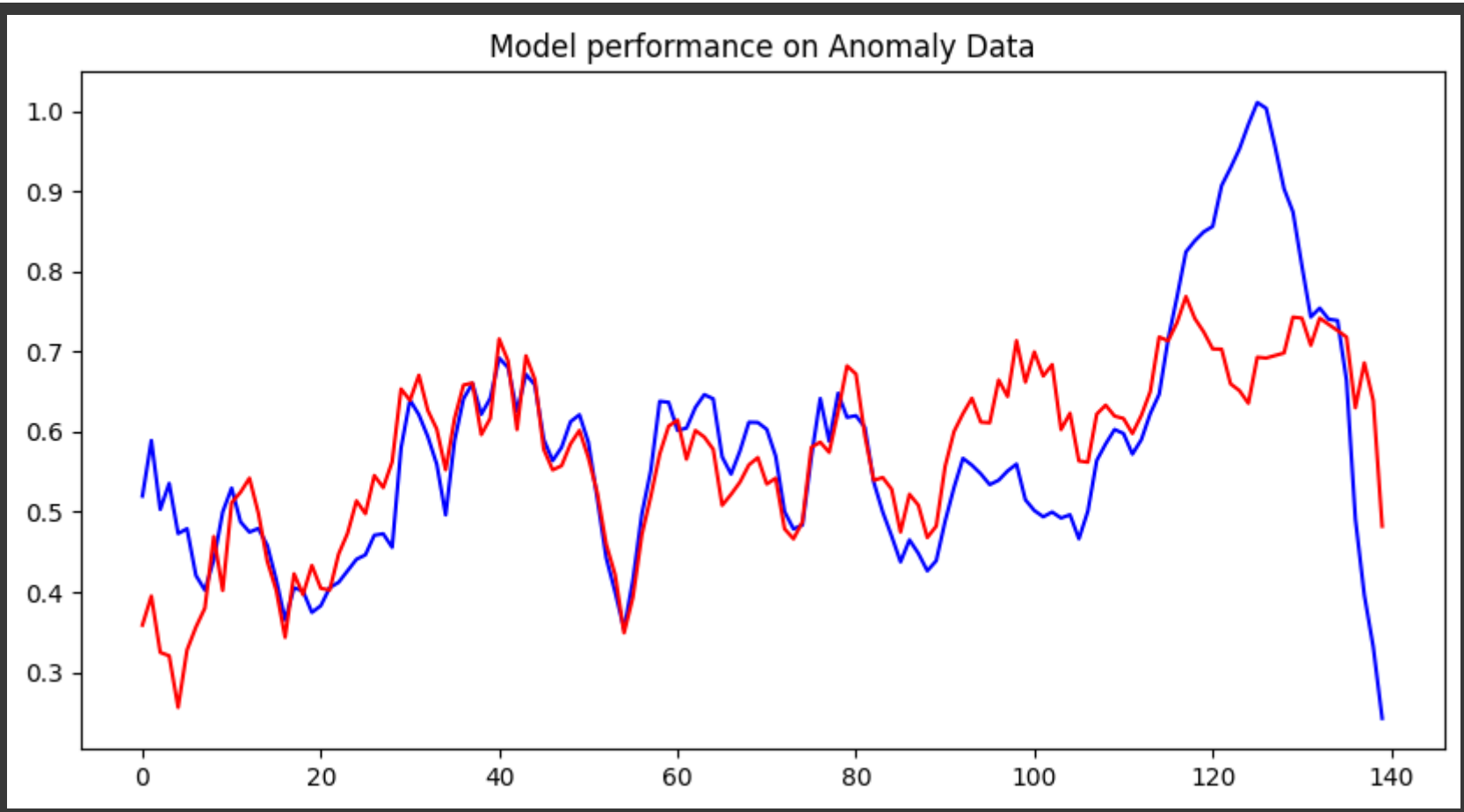
```
20/20 [==============================] - 0s 8ms/step - loss: 0.0330 - val_loss: 0.0573
Epoch 25/50
20/20 [==============================] - 0s 9ms/step - loss: 0.0315 - val_loss: 0.0540
Epoch 26/50
20/20 [==============================] - 0s 8ms/step - loss: 0.0299 - val_loss: 0.0538
Epoch 27/50
20/20 [==============================] - 0s 13ms/step - loss: 0.0291 - val_loss: 0.0519
Epoch 28/50
20/20 [==============================] - 0s 12ms/step - loss: 0.0283 - val_loss: 0.0520
Epoch 29/50
20/20 [==============================] - 0s 9ms/step - loss: 0.0281 - val_loss: 0.0523
```

```
# Model Evaluation
encoder_out = model.encoder(normal_test_data).numpy()
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()
```



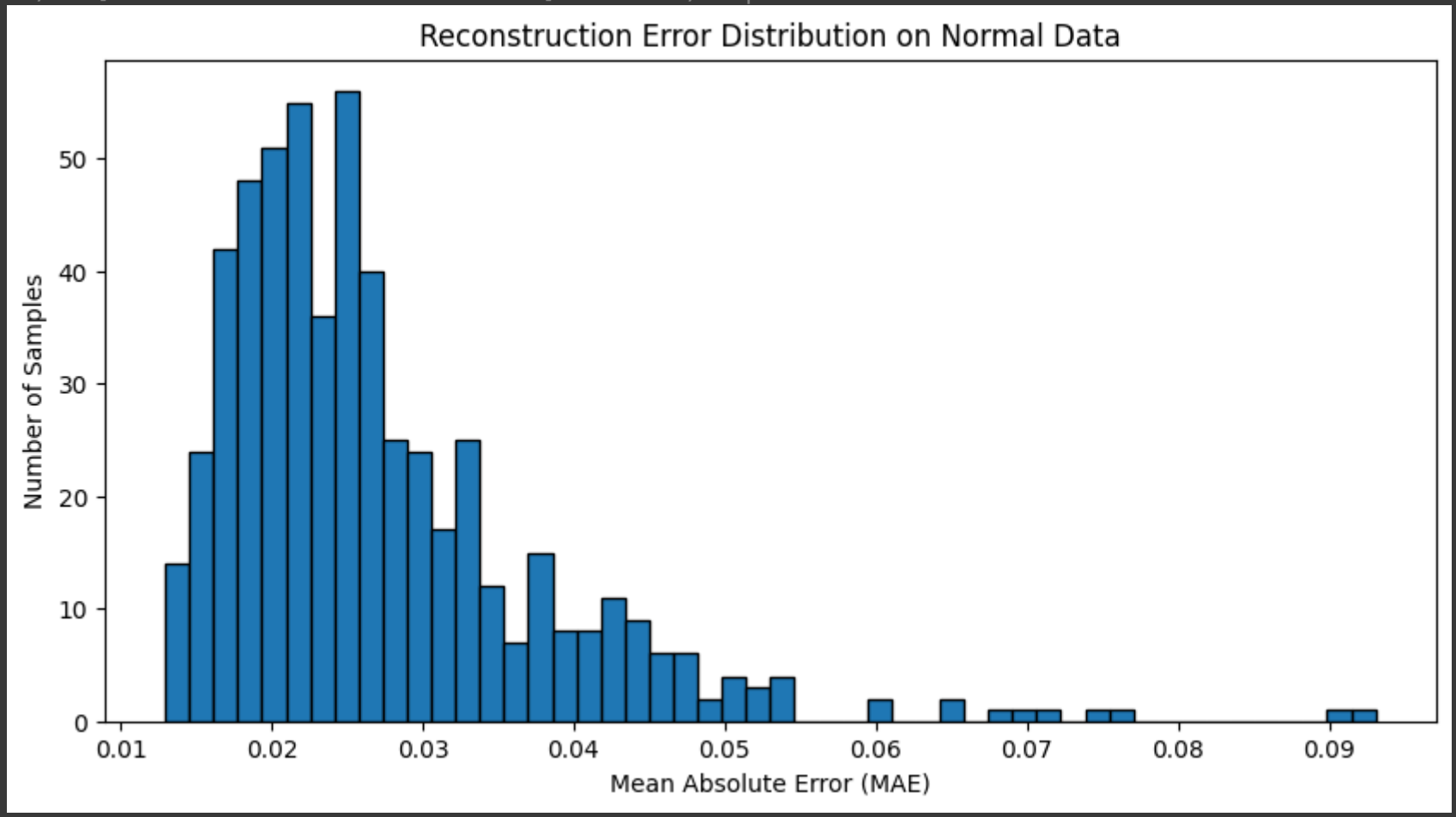

```
encoder_out_a = model.encoder(anomaly_test_data).numpy()
decoder_out_a = model.decoder(encoder_out_a).numpy()
plt.plot(anomaly_test_data[0], 'b')
plt.plot(decoder_out_a[0], 'r')
plt.title("Model performance on Anomaly Data")
plt.show()
```




```
# Calculate Loss
reconstruction = model.predict(normal_test_data)
train_loss = tf.keras.losses.mae(reconstruction, normal_test_data)

# Plotting Histogram
plt.hist(train_loss, bins=50, edgecolor='black')
plt.xlabel('Mean Absolute Error (MAE)')
plt.ylabel('Number of Samples')
plt.title('Reconstruction Error Distribution on Normal Data')
plt.show()
```

```
18/18 [==============================] - 0s 2ms/step
```




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

```
14/14 [==============================] - 0s 3ms/step
```