GROUP-6 ANOMALY DETECTION USING AUTOENCODERS

MASKANI NAVEEN YADAV - 20201CEI0025

KATIPALLY YASHWANTH REDDY - 20201CEI0039

PERAM MAHENDRA REDDY - 20201CEI0112

INDLA MADHANKUMAR - 20201CEI0136


```
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 "ECG5000_TRAIN.txt" "ECG5000_TEST.txt" > ecg_final.txt
```

```
df = pd.read_csv("ecg_final.txt", sep='  ', header=None)
df.shape
```

```
<ipython-input-3-4e29b172af0b>:1: ParserWarning: Falling back to the 'python' engine because the 'c' engine does not support regex separ
  df = pd.read_csv("ecg_final.txt", sep='  ', header=None)
(5000, 141)
```

```
df.head()
```

	0	1	2	3	4	5	6	7	8	9	...	131	132	133	
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	0.933541	0.796
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	0.656881	0.787
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	0.531452	0.311
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	0.499111	0.600
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	0.958434	1.059

5 rows × 141 columns

```
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
```

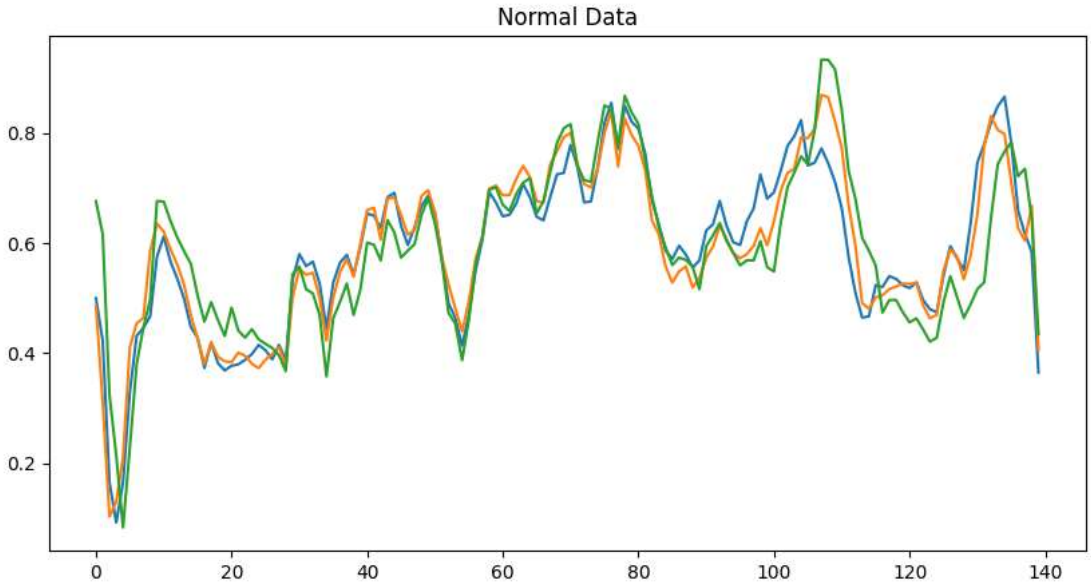
```
X_train,X_test,y_train,y_test=train_test_split(df.values,df.values[:,0:1],test_size=0.2,random_state=111)
```

```
scaler=MinMaxScaler()
data_scaled=scaler.fit(X_train)
train_data_scaled=data_scaled.transform(X_train)
test_data_scaled=data_scaled.transform(X_test)
```

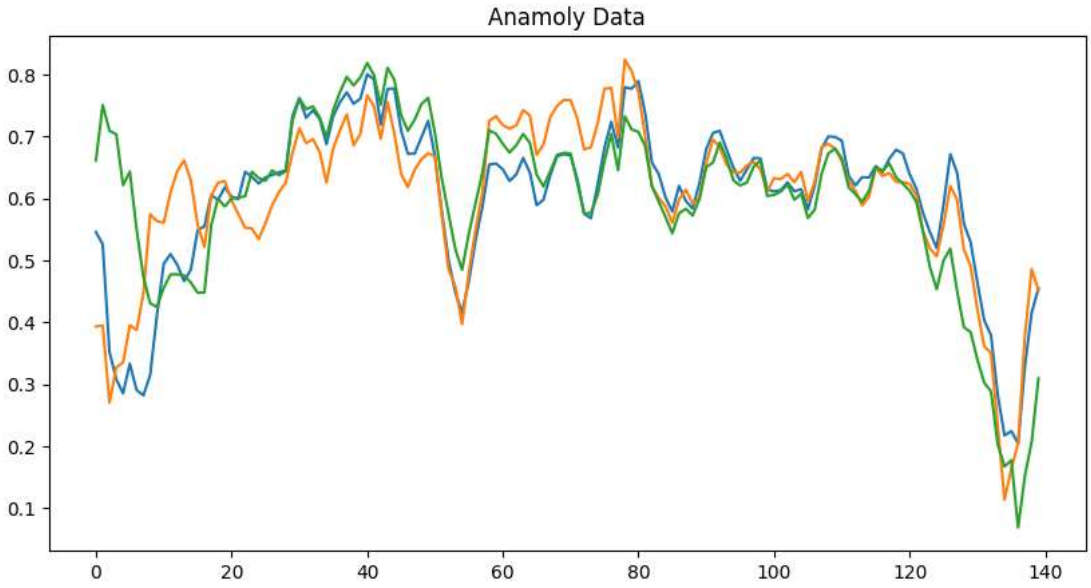
```
normal_train_data=pd.DataFrame(train_data_scaled).add_prefix("c").query("c0==0").values[:,1:]
anamoly_train_data=pd.DataFrame(train_data_scaled).add_prefix("c").query("c0>0").values[:,1:]
normal_test_data=pd.DataFrame(test_data_scaled).add_prefix("c").query("c0==0").values[:,1:]
```

```
anamoly_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()
```

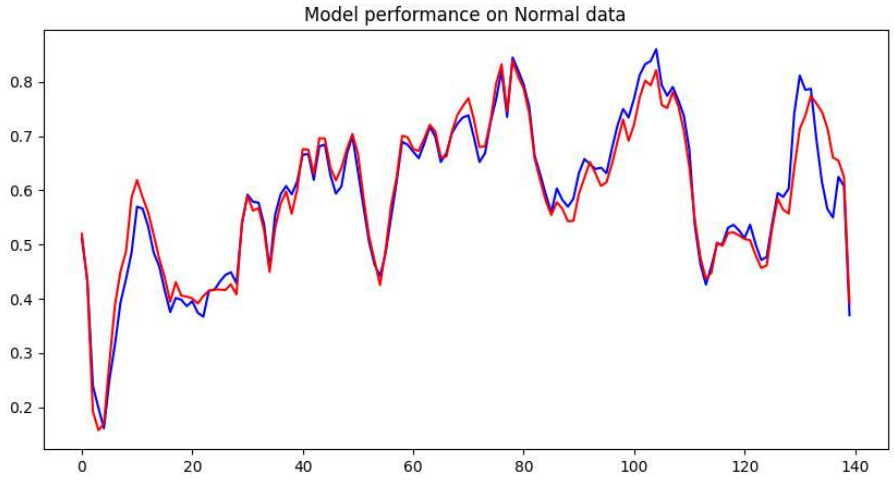



```
plt.plot(anamoly_train_data[0])
plt.plot(anamoly_train_data[1])
plt.plot(anamoly_train_data[2])
plt.title("Anamoly Data")
plt.show()
```

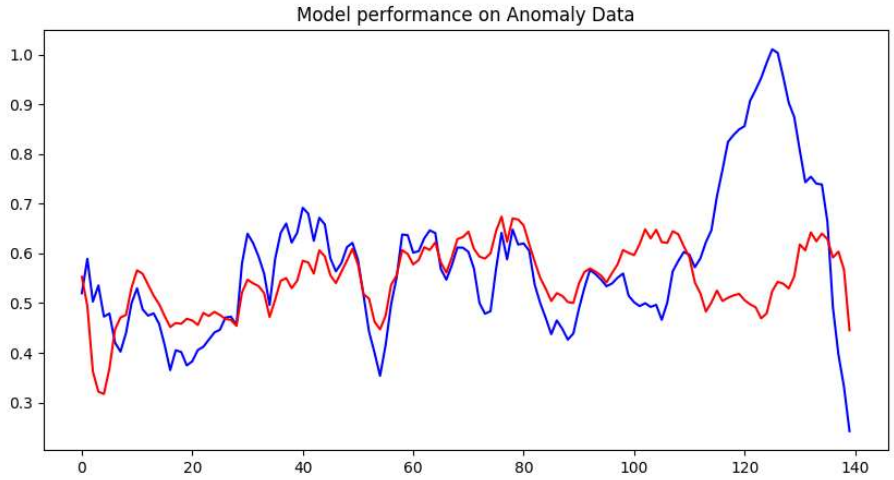



```
model = tf.keras.Sequential()
model.add(tf.keras.layers.Dense(64, activation="relu"))
model.add(tf.keras.layers.Dense(32, activation="relu"))
model.add(tf.keras.layers.Dense(16, activation="relu"))
model.add(tf.keras.layers.Dense(8, activation="relu"))
model.add(tf.keras.layers.Dense(16, activation="relu"))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

     Epoch 1/50
     20/20 [==============================] - 3s 23ms/step - loss: 0.1194 - val_loss: 0.1005
     Epoch 2/50
     20/20 [==============================] - 0s 11ms/step - loss: 0.0674 - val_loss: 0.0784
     Epoch 3/50
     20/20 [==============================] - 0s 13ms/step - loss: 0.0500 - val_loss: 0.0762
     Epoch 4/50
     20/20 [==============================] - 0s 11ms/step - loss: 0.0481 - val_loss: 0.0754
     Epoch 5/50
     20/20 [==============================] - 0s 10ms/step - loss: 0.0478 - val_loss: 0.0747
     Epoch 6/50
     20/20 [==============================] - 0s 13ms/step - loss: 0.0476 - val_loss: 0.0741
     Epoch 7/50
     20/20 [==============================] - 0s 14ms/step - loss: 0.0474 - val_loss: 0.0738
     Epoch 8/50
     20/20 [==============================] - 0s 10ms/step - loss: 0.0472 - val_loss: 0.0736
     Epoch 9/50
     20/20 [==============================] - 0s 13ms/step - loss: 0.0470 - val_loss: 0.0732
     Epoch 10/50
     20/20 [==============================] - 0s 12ms/step - loss: 0.0470 - val_loss: 0.0732
     Epoch 11/50
     20/20 [==============================] - 0s 8ms/step - loss: 0.0468 - val_loss: 0.0729
     Epoch 12/50
     20/20 [==============================] - 0s 7ms/step - loss: 0.0468 - val_loss: 0.0728
     Epoch 13/50
     20/20 [==============================] - 0s 7ms/step - loss: 0.0466 - val_loss: 0.0715
     Epoch 14/50
     20/20 [==============================] - 0s 8ms/step - loss: 0.0446 - val_loss: 0.0652
     Epoch 15/50
     20/20 [==============================] - 0s 14ms/step - loss: 0.0401 - val_loss: 0.0623
     Epoch 16/50
     20/20 [==============================] - 0s 11ms/step - loss: 0.0374 - val_loss: 0.0619
     Epoch 17/50
     20/20 [==============================] - 0s 14ms/step - loss: 0.0363 - val_loss: 0.0604
     Epoch 18/50
     20/20 [==============================] - 0s 13ms/step - loss: 0.0358 - val_loss: 0.0596
     Epoch 19/50
     20/20 [==============================] - 0s 7ms/step - loss: 0.0354 - val_loss: 0.0593
     Epoch 20/50
     20/20 [==============================] - 0s 7ms/step - loss: 0.0351 - val_loss: 0.0594
     Epoch 21/50
     20/20 [==============================] - 0s 8ms/step - loss: 0.0350 - val_loss: 0.0594


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()
```

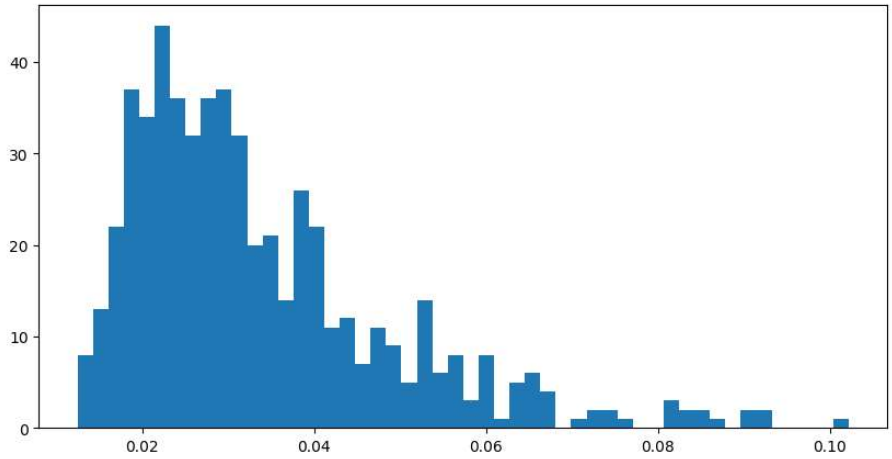



```
encoder_out_a = model.encoder(anamoly_test_data).numpy()
decoder_out_a = model.decoder(encoder_out_a).numpy()
plt.plot(anamoly_test_data[0], 'b')
plt.plot(decoder_out_a[0], 'r')
plt.title("Model performance on Anomaly Data")
plt.show()
```



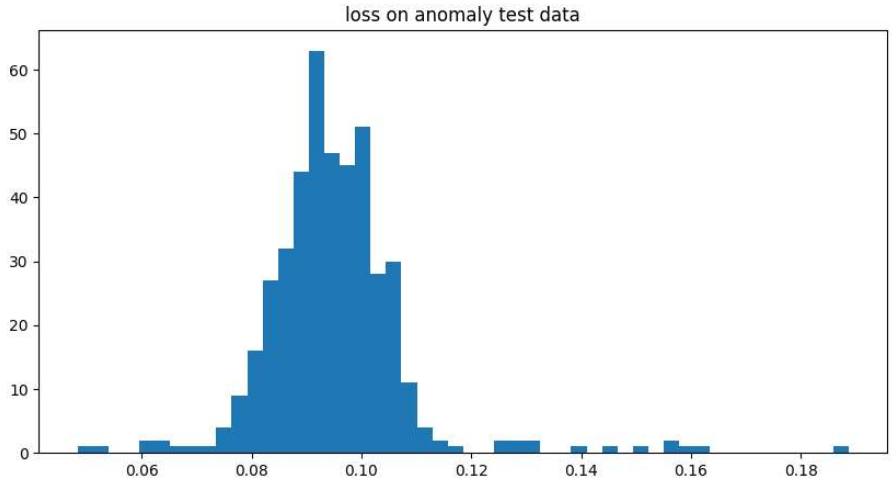

```
reconstruction = model.predict(normal_test_data)
train_loss = tf.keras.losses.mae(reconstruction, normal_test_data)
plt.hist(train_loss, bins=50)
```

```
18/18 [==============================] - 0s 3ms/step
(array([ 8., 13., 22., 37., 34., 44., 36., 32., 36., 37., 32., 20., 21.,
        14., 26., 22., 11., 12.,  7., 11.,  9.,  5., 14.,  6.,  8.,  3.,
         8.,  1.,  5.,  6.,  4.,  0.,  1.,  2.,  2.,  1.,  0.,  0.,  3.,
         2.,  2.,  1.,  0.,  2.,  2.,  0.,  0.,  0.,  0.,  1.]),
 array([0.01243564, 0.01423151, 0.01602739, 0.01782327, 0.01961915,
        0.02141503, 0.02321091, 0.02500679, 0.02680267, 0.02859855,
        0.03039443, 0.03219031, 0.03398619, 0.03578207, 0.03757795,
        0.03937383, 0.04116971, 0.04296559, 0.04476147, 0.04655735,
        0.04835323, 0.05014911, 0.05194499, 0.05374086, 0.05553674,
        0.05733262, 0.0591285 , 0.06092438, 0.06272026, 0.06451614,
        0.06631202, 0.0681079 , 0.06990378, 0.07169966, 0.07349554,
        0.07529142, 0.0770873 , 0.07888318, 0.08067906, 0.08247494,
        0.08427082, 0.0860667 , 0.08786258, 0.08965846, 0.09145434,
        0.09325021, 0.09504609, 0.09684197, 0.09863785, 0.10043373,
        0.10222961]),
 <BarContainer object of 50 artists>)
```

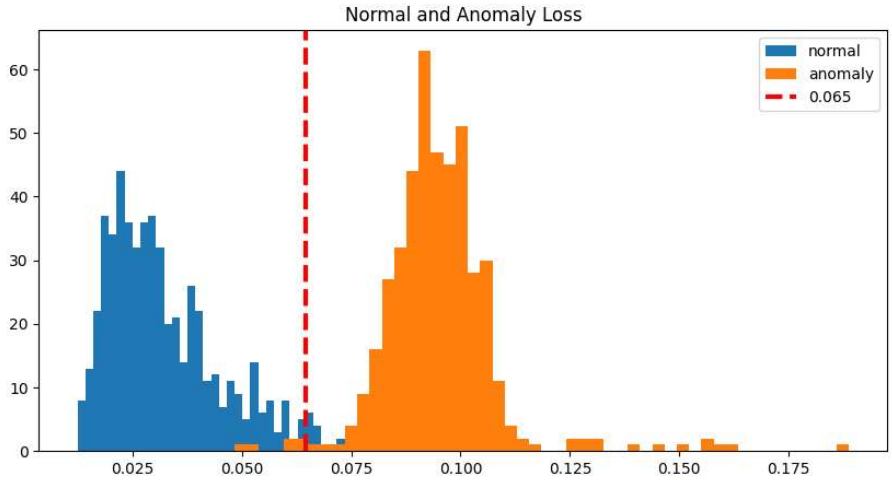



```
threshold = np.mean(train_loss) + 2*np.std(train_loss)
reconstruction_a = model.predict(anamoly_test_data)
train_loss_a = tf.keras.losses.mae(reconstruction_a, anamoly_test_data)
plt.hist(train_loss_a, bins=50)
plt.title("loss on anomaly test data")
plt.show()
```

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



```
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=534>
```

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

```
<tf.Tensor: shape=(), dtype=int64, numpy=432>
```

```
normal_test_data.shape
```

```
(563, 140)
```

```
anamoly_test_data.shape
```

```
(437, 140)
```

```
Start coding or generate with AI.
```