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

path = ‘ ‘ ‘ [http://storage.googleapis.com/download. tensorflow.org/data/ecg.csv](http://storage.googleapis.com/download.%20tensorflow.org/data/ecg.csv)’ ‘ ‘

data = pd.read\_csv(path, header=None)

print(data.shape)

data.head()

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,

random\_state = 0, stratify=target

)

x\_test.shape

x\_train.shape

target.value\_counts()

train\_index = y\_train[y\_train == 1].index

train\_data = x\_train.loc[train\_index]

min\_max\_scaler = MinMaxScaler()

x\_train\_scaled = min\_max\_scaler.fit\_transform(

train\_data.copy())

x\_test\_scaled = min\_max\_scaler.transform(x\_test.copy())

x\_train.describe()

pd.DataFrame(x\_train\_scaled).describe()

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

model = AutoEncoder(output\_units=x\_train\_scaled.shape[1])

model.compile(loss='msle', metrics=['mse'], optimizers=’adam’)

history = model.fit(

x\_train\_scaled,

x\_train\_scaled,

epochs=20,

batch\_size=512,

validation\_data=(x\_test\_scaled, x\_test\_scaled)

)

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()