Assignment No.4

Title : ECG Anomaly detection using Autoencoders

Aim: Use Autoencoder to implement anomaly detection. Build the model by using:
a. Import required libraries
b. Upload / access the dataset
c. Encoder converts it into latent representation
d. Decoder networks convert it back to the original input
e. Compile the models with Optimizer, Loss, and Evaluation Metrics

Theory：

Steps/ Algorithm

1. Dataset link and libraries :
Dataset : http://storage.googleapis.com/download.tensorflow.org/data/ecg.csv
Libraries required :

Pandas and Numpy for data manipulation

Tensorflow/Keras for Neural Networks

Scikit-learn library for splitting the data into train-test samples, and for some basic model evaluation

For Model building and evaluation following libraries:

sklearn.metrics import accuracy_score

tensorflow.keras.optimizers import Adam

sklearn.preprocessing import MinMaxScaler

tensorflow.keras import Model, Sequential

tensorflow.keras.layers import Dense, Dropout

tensorflow.keras.losses import MeanSquaredLogarithmicError

Ref:https://www.analyticsvidhya.com/blog/2021/05/anomaly-detection-using-autoencoders-a-walk-through-in-python/

a) Import following libraries from SKlearn : i) MinMaxscaler (sklearn.preprocessing) ii) Accuracy(sklearn.metrics) . iii) train_test_split (model_selection)

b) Import Following libraries from tensorflow.keras : models , layers,optimizers,datasets , and set to respective values.

c) Grab to ECG.csv required dataset

d) Find shape of dataset

e) Use train_test_split from sklearn to build model (e.g. train_test_split(

features, target, test_size=0.2, stratify=target)

f) Take usecase Novelty detection hence select training data set as Target class is 1 i.e. Normal class

g) Scale the data using MinMaxScaler.

h) Create Autoencoder Subclass by extending model class from keras.

i) Select parameters as i)Encoder : 4 layers ii) Decoder : 4 layers iii) Activation Function : Relu iv) Model : sequential.

j) Configure model with following parametrs : epoch = 20 , batch size =512 and compile with Mean Squared Logarithmic loss and Adam optimizer.

```
e.g. 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)
```

k) Plot loss,Val_loss, Epochs and msle loss

l) Find threshold for anomaly and do predictions :

```
e.g. : 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
```

m) Get accuracy score

Sample Code with comments : Attach Printout with Output .

Conclusion: In such a way we use Autoencoder to implement anomaly

detection. To the build required model.