**Second Programming Exam.**

**Deep Learning, 2022**

姓名：吳佳展 學號：110318517

import pandas as pd

import numpy as np

from random import randrange

from matplotlib import pyplot as plt

import pandas as pd

import numpy as np

import xgboost as xgb

import pickle, os, pymssql

from sklearn.preprocessing import MinMaxScaler

from tensorflow.keras.layers import LSTM, Dropout, Dense, Activation

from tensorflow.keras.models import Sequential, load\_model

from tensorflow.keras.callbacks import ModelCheckpoint, EarlyStopping

from sys import exit

def plot\_loss(x, y):

    plt.figure(figsize=(12,6))

    plt.plot(x)

    plt.plot(y)

    plt.title('LSTM Model loss')

    plt.ylabel('MSE')

    plt.xlabel('epoch')

    plt.legend(['Train\_loss', 'Val\_loss'], loc='upper right')

def Get\_data():

    df\_train = pd.read\_csv('predict1\_trn.csv')

    df\_test = pd.read\_csv('predict1\_tst.csv')

    X\_train, y\_train = df\_train.iloc[:,:24], df\_train.iloc[:,-6:]

    X\_test = df\_test.iloc[:,:24]

    return X\_train, y\_train, X\_test

def Normalize():

    df\_train = pd.read\_csv('predict1\_trn.csv')

    scaler = MinMaxScaler(feature\_range=(0,1))

    scaled = scaler.fit\_transform(df\_train)

    X\_scaled, y\_scaled = scaled[:,:24], scaled[:,-6:]

    return scaler, X\_scaled, y\_scaled

def Build\_Model(X, y):

    model = Sequential()

    model.add(LSTM(50, return\_sequences = True, input\_shape = (X.shape[1], 1)))

    model.add(Dropout(0.2))

    model.add(LSTM(50, return\_sequences = True))

    model.add(Dropout(0.2))

    model.add(LSTM(50))

    model.add(Dropout(0.2))

    model.add(Dense(y.shape[1]))

    model.summary()

    model.compile(loss ='mse', optimizer='adam', metrics=['mse']) #rmsprop

    return model

def Train\_Model(X, y):

    re\_X = X.reshape(X.shape[0], X.shape[1],1)

    file1 = 'Q2/predict1.h5'

    checkpoint = ModelCheckpoint(file1, monitor='val\_loss', verbose=2, save\_best\_only = True, mode='min')

    earlyStopping = EarlyStopping(monitor='val\_loss', patience=50, verbose=2, mode='auto')

    callbacks\_list= [checkpoint,earlyStopping]

    lstm\_model = Build\_Model(re\_X, y)

    history = lstm\_model.fit(re\_X, y, epochs=90, batch\_size=30, callbacks=callbacks\_list, validation\_split=0.1)

    # savemodel

    lstm\_model.save(file1)

    return history.history ['loss'], history.history ['val\_loss']

def Predict(model, X\_test):

    temp\_x = np.concatenate((X\_test.iloc[:,:].values, X\_test.iloc[:,-6:].values),axis=1)

    temp\_x = scaler.fit\_transform(temp\_x)

    y\_hat = model.predict(temp\_x[:,:24])

    # inverse

    all\_test = np.concatenate((temp\_x[:,:24], y\_hat),axis=1)

    actual\_test = scaler.inverse\_transform(all\_test)

    return y\_hat, actual\_test

def Export\_csv(data):

    cols = []

    for i in range(-23,7,1):

        if i <0 :

            cols.append(f't-{i\*(-1)}')

        elif i==0:

            cols.append('t')

        else:

            cols.append(f't+{i}')

    df\_fn = pd.DataFrame(data)

    df\_fn.to\_csv('Q2/predict1\_answer.csv', index=None, header=cols)

# read csv data

X\_train, y\_train, X\_test = Get\_data()

# normalize

scaler, X\_scaled, y\_scaled = Normalize()

# train model

loss, val\_loss = Train\_Model(X\_scaled, y\_scaled)

# plot loss

plot\_loss(loss, val\_loss)

# load model

pre\_model= load\_model('Q2/predict1.h5')

# predict value

y\_hat, actual = Predict(pre\_model, X\_test)

# export csv

Export\_csv = Export\_csv(actual)