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)
```