import numpy as np

from keras.models import Sequential

from keras.layers import Dense, Dropout, LSTM, Bidirectional, Conv1D, MaxPooling1D, Flatten

from sklearn.preprocessing import MinMaxScaler

import pandas as pd

from google.colab import drive

drive.mount("/content/gdrive")

# Define the input attributes and output variable

features = ['co','NO','No2','NOx','O3','PM10','PM2.5','SO2','WIND\_DIREC','WIND\_SPEED']

target = 'PM2.5'

# Load the dataset

df = pd.read\_csv('/content/gdrive/My Drive/dataset\_1.csv', usecols=['co','NO','No2','NOx','O3','PM10','PM2.5','SO2','WIND\_DIREC','WIND\_SPEED'])

dataset = df.to\_numpy()

print(dataset.shape)

# Convert the dataframe to a numpy array

dataset = df.values.astype('float32')

# Split the dataset into training and testing sets

# Split the dataset into training and testing sets

train\_size = int(0.8 \* len(dataset))

train\_dataset = dataset[:train\_size]

test\_dataset = dataset[train\_size:]

# Scale the input data to a range of 0 to 1

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

train\_data = scaler.fit\_transform(train\_data)

test\_data = scaler.transform(test\_data)

# Split the input and output data for training and testing sets

train\_X, train\_y = train\_data[:, :-1], train\_data[:, -1]

test\_X, test\_y = test\_data[:, :-1], test\_data[:, -1]

# Reshape the input data to be 3-dimensional for the LSTM-CNN model

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

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

# Define the LSTM-CNN model architecture

model = Sequential()

model.add(Conv1D(filters=64, kernel\_size=1, activation='relu', input\_shape=(1, 9)))

model.add(MaxPooling1D(pool\_size=1))

model.add(Bidirectional(LSTM(50)))

model.add(Dense(1))

model.compile(loss='mse', optimizer='adam')

# Train the LSTM-CNN model

history = model.fit(train\_X, train\_y, epochs=100, batch\_size=72, validation\_data=(test\_X, test\_y), verbose=2, shuffle=False)

# Define the LSTM-CNN model

model = Sequential()

model.add(Bidirectional(LSTM(64, activation='relu'), input\_shape=(10, 1)))

model.add(Conv1D(filters=32, kernel\_size=3, activation='relu'))

model.add(MaxPooling1D(pool\_size=2))

model.add(Flatten())

model.add(Dense(1, activation='linear'))

# Compile the model

model.compile(loss='mse', optimizer='adam')

# Train the model

model.fit(train\_X, train\_y, epochs=100, batch\_size=32, validation\_data=(test\_X, test\_y))

# Evaluate the model on the test data

mse = model.evaluate(test\_X, test\_y)

print('Mean Squared Error:', mse)

# Make predictions on new data

new\_data = np.array([[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]])

new\_data\_scaled = scaler.transform(new\_data)

new\_data\_reshaped = new\_data\_scaled.reshape((new\_data\_scaled.shape[0], new\_data\_scaled.shape[1], 1))

prediction = model.predict(new\_data\_reshaped)

print('Predicted PM2.5:', prediction)