**LAB09-IoT input validation AI process**

**Class: M03 student ID: B2111933 Name: Truong Dang Truc Lam**

1. Clone code example from below modules using ref,site or slide

module 1 : Saving IoT Sensor Data to an IoT Server's Database

module 2 : Searching the Saved Database Data in an AI Program for Data Composition

module 3 : Validating Input Data in the IoT Server's Database Using an AI Method

**I choose module 2**

2.Select IoT software environment/IDE (multiple choice possible)

Ubuntu / Python 3 / nano

3. Design IoT software Architecture

| Items | Resource |  |  |  |
| --- | --- | --- | --- | --- |
| VM type |  |  |  |  |
| OS | Ubuntu |  |  |  |
| IP/URL |  |  |  |  |
| Language,version | Python 3 |  |  |  |
| framework |  |  |  |  |
| Libraries |  |  |  |  |
| Software tool | nano |  |  |  |
| Protocol; |  |  |  |  |
| Message broker |  |  |  |  |
| Software tool |  |  |  |  |
| Container |  |  |  |  |
| Code Reference | Dr.Noh’s tutorial |  |  |  |
|  |  |  |  |  |

4. Design your coding process (write used resource)

import sqlite3

import pandas as pd

import numpy as np

from sklearn.preprocessing import MinMaxScaler

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import LSTM, Dense

# Ensure the database and table exist

def initialize\_database():

conn = sqlite3.connect('iot\_data.db')

cursor = conn.cursor()

cursor.execute("""

CREATE TABLE IF NOT EXISTS sensor\_data (

id INTEGER PRIMARY KEY AUTOINCREMENT,

temperature REAL,

humidity REAL,

timestamp TEXT

)

""")

conn.commit()

conn.close()

# Insert test data if the table is empty

def insert\_test\_data():

conn = sqlite3.connect('iot\_data.db')

cursor = conn.cursor()

cursor.execute("SELECT COUNT(\*) FROM sensor\_data")

count = cursor.fetchone()[0]

if count == 0: # Insert sample data only if table is empty

sample\_data = [

(25.3, 60.5, '2025-03-18 12:00:00'),

(26.1, 58.2, '2025-03-18 12:05:00'),

(24.8, 62.1, '2025-03-18 12:10:00'),

(23.7, 64.0, '2025-03-18 12:15:00'),

(27.0, 55.8, '2025-03-18 12:20:00'),

(26.5, 57.1, '2025-03-18 12:25:00')

]

cursor.executemany("INSERT INTO sensor\_data (temperature, humidity, timestamp) VALUES (?, ?, ?)", sample\_data)

conn.commit()

conn.close()

# Load data from the IoT database

def load\_data():

conn = sqlite3.connect('iot\_data.db')

query = "SELECT temperature, humidity, timestamp FROM sensor\_data ORDER BY timestamp ASC"

df = pd.read\_sql\_query(query, conn)

conn.close()

if df.empty:

raise ValueError("No data found in sensor\_data table. Please check your database.")

return df

# Preprocessing

def preprocess\_data(df):

df['timestamp'] = pd.to\_datetime(df['timestamp'])

df.sort\_values(by='timestamp', inplace=True)

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

scaled\_data = scaler.fit\_transform(df[['temperature', 'humidity']])

X, y = [], []

seq\_length = 5 # Past 5 data points to predict the next

for i in range(len(scaled\_data) - seq\_length):

X.append(scaled\_data[i:i+seq\_length])

y.append(scaled\_data[i+seq\_length])

return np.array(X), np.array(y), scaler

# AI Model (LSTM)

def build\_lstm\_model(input\_shape):

model = Sequential([

LSTM(50, return\_sequences=True, input\_shape=input\_shape),

LSTM(50),

Dense(2)

])

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

return model

# Main execution

initialize\_database()

insert\_test\_data()

df = load\_data()

X, y, scaler = preprocess\_data(df)

model = build\_lstm\_model(X.shape[1:])

model.fit(X, y, epochs=10, batch\_size=16, verbose=1)

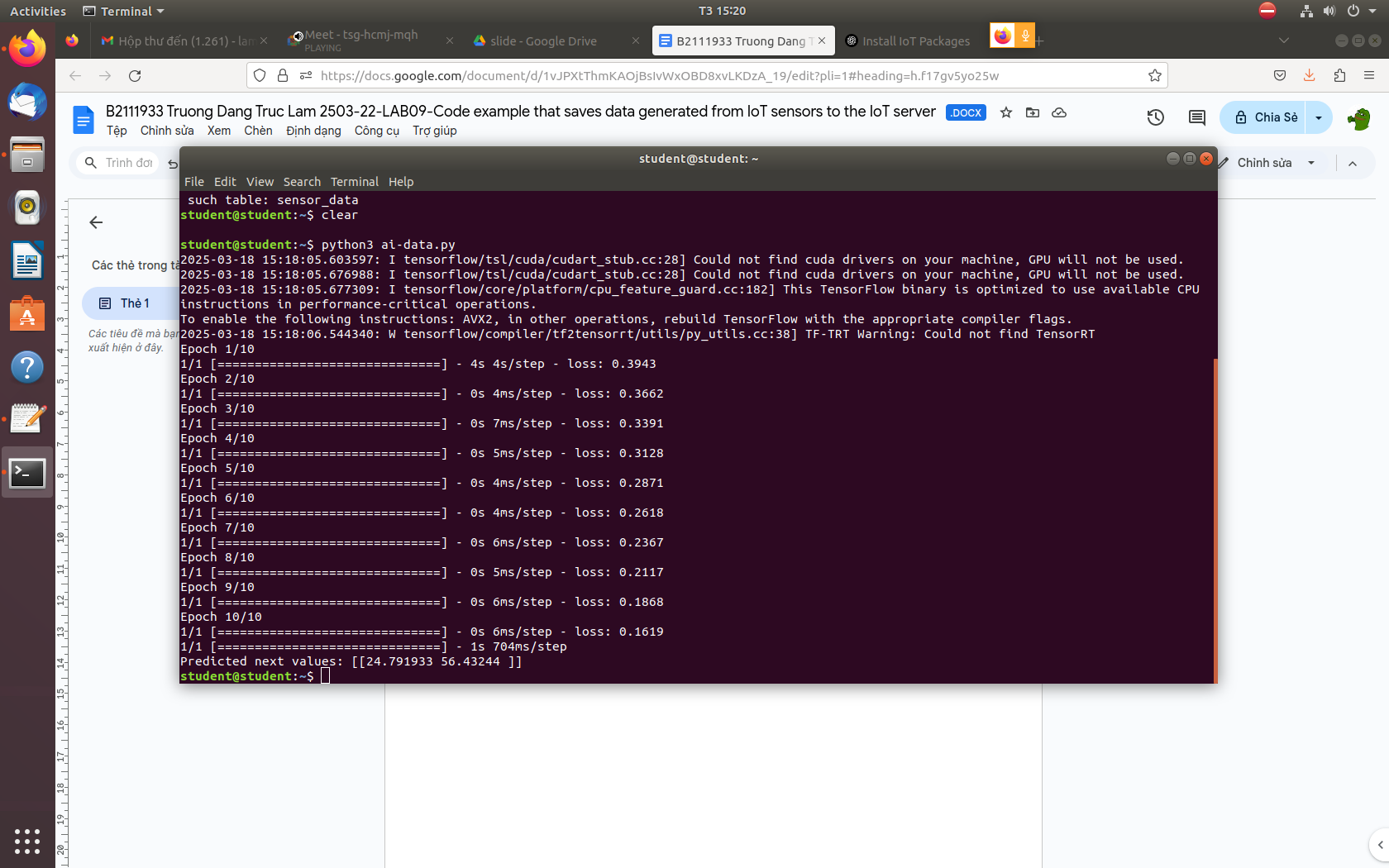
# Predict the next sensor data

next\_prediction = model.predict(np.expand\_dims(X[-1], axis=0))

predicted\_values = scaler.inverse\_transform(next\_prediction)

print("Predicted next values:", predicted\_values)

5. Execute your process and explain as far as you can (optional)



Executed code

This Python script **loads sensor data from an SQLite database, processes it, trains an LSTM model, and predicts future sensor values**.

## 1️⃣ Ensure Database and Table Exist

**Function: initialize\_database()**

* Creates the sensor\_data table if it does not exist.
* Prevents SQL errors due to missing tables.

## 2️⃣ Insert Test Data if Table is Empty

**Function: insert\_test\_data()**

* Checks if the table contains data.
* Inserts sample sensor readings if the table is empty.
* Ensures the script can run even if no real data is available.

## 3️⃣ Load Data from the Database

**Function: load\_data()**

* Connects to the SQLite database and retrieves temperature, humidity, and timestamp from sensor\_data.
* Orders the data by timestamp.
* Raises an error if the table is empty.

## 4️⃣ Preprocess Data for LSTM

**Function: preprocess\_data(df)**

* Converts the timestamp column to datetime.
* Sorts data in ascending order.
* Scales temperature and humidity values between 0 and 1 using MinMaxScaler.
* Prepares sequences of 5 past values (X) to predict the next value (y).

## 5️⃣ Build LSTM Model

**Function: build\_lstm\_model(input\_shape)**

* Creates an LSTM neural network with two stacked LSTM layers.
* Uses 50 units per layer.
* Outputs two values (temperature and humidity).
* Compiles the model with adam optimizer and mse loss.

## 6️⃣ Train the Model

* Uses model.fit() to train on X (input sequences) and y (targets).
* Runs for 10 epochs with a batch size of 16.

## 7️⃣ Make Predictions

* Takes the latest 5 sensor readings and predicts the next temperature and humidity.
* Uses scaler.inverse\_transform() to convert the prediction back to real values.
* Prints the predicted values.