Machine Learning for IoT

Homework 3

*****DUE DATE: 31 Jan (h23:59)*****

Submission Instructions:

Each group will send an e-mail to daniele.jahier@polito.it and valentino.peluso@polito.it (in cc) with subject *ML4IOT25 GroupN* (replace *N* with the group ID). Attached to the e-mail a single ZIP archive (*.zip*) named *HW3_GroupN.zip* (replace *N* with the group ID) containing the following files:

1. The code deliverables specified in the text of each exercise.
2. One-page pdf report, titled *GroupN_Homework3.pdf*, organized in different sections (one for each exercise). Each section should motivate the main adopted design choices and discuss the outcome of the exercise.

Late messages, or messages not compliant with the above specs, will be automatically discarded.

Exercise 1: Data Collection, Communication, and Storage (3 points)

1.1 Data Collection and Communication with the MQTT protocol (1pt)

On the RPI, develop a Python script (*publisher.py*) that **every 2 seconds**:

a) Measure temperature & humidity using the DHT-11 sensor.

b) Create a JSON message with the following fields:

Parameter	Description
mac_address	string, the MAC address of the RPI (obtained with *hex(uuid.getnode())*).
timestamp	integer, the timestamp in milliseconds.
temperature	integer, the temperature value in in °C.
humidity	integer, the humidity in %RH.

Example:

```
{
  mac_address: "0xf0b61e0bfe09",
  timestamp: 1664630309530,
  temperature: 21,
  humidity: 60,
}
```

c) Publish the JSON message to the Eclipse MQTT broker eclipse (mqtt.eclipseprojects.io at port 1883) using the student ID of a member of your team as message topic. (e.g., *s001122*).

1.2 Data Storage (1pt)

On Deepnote, create a new Python notebook (*subscriber.ipynb*) to develop an MQTT subscriber that receives the messages with temperature & humidity collected by the RPI and store the received data in Redis. For data storage, use two Redis TimeSeries, called *mac_address:*temperature and *mac_address:*humidity, respectively (e.g., *0xf0b61e0bfe09:temperature* and *0xf0b61e0bfe09:humidity*), where *mac_address* is the MAC address of the RPI publisher.

Run the notebook while collecting data with the script of 1.1.

1.3 Reporting (1pt)

In the report, explain why MQTT is a better choice than REST as the communication protocol for this application.

Deliverables

- A Python script named *publisher.py* that contains the code of 1.1. The code is intended to be run on the Raspberry PI without installing additional software or dependencies on the provided SD card. Moreover, the script must include all necessary classes and methods needed for its correct execution.
- A Deepnote notebook named *subscriber.ipynb* that contains the code of 1.2. The notebook is intended to be run on Deepnote, must use only the packages that get installed with the *ML4IoT Template* and must run without any additional dependency.

Exercise 2: Data Management & Visualization (3pts)

2.1 REST Server (1pt)

In Deepnote, duplicate the notebook **"LAB: 5.2 REST API"** of the *Material* project and extend the API to enables users to retrieve historical temperature & humidity data. Specifically, the API must be extended providing the additional resource and endpoint reported below.

Resource: HistoricalData

Parameter	Description
mac_address	string, the MAC address of the PC network card
timestamp	list of integers, each integer is the timestamp in milliseconds
temperature	list of integers, each integer is a temperature value in in °C.
humidity	list of integers, each integer is a humidity value in %RH.

Example:

```
{
  mac_address: "0xf0b61e0bfe09",
  timestamp: [1664630309530, 1664630310567, 1664630311542],
  temperature: [22, 22, 23],
  humidity: [60, 61, 59],
}
```

Endpoint: /data/{mac_address}

- ____ **/data/{mac_address}** (fill the blank space with the most appropriate HTTP method)

  Description: Retrieve temperature and humidity data collected by the device with the specified MAC address in the specified date range.

  Path Parameters:

Parameter	Description
mac_address	string (required), the MAC address of the RPI to retrieve.

  Query Parameters:

Parameter	Description
start_date	sring (required), the start of the date range in which to retrieve temperature & humidity data. Date must be specified with the ISO 8601 format (e.g. 2023-12-21).
end_date	sring (required), the end of the date range in which to retrieve temperature & humidity data. Date must be specified with the ISO 8601 format (e.g. 2023-12-22). It must be greater than start_date.

  Body Parameters: N/A

Response Status Code:

- 200 – OK: Everything worked as expected.
- 400 – Bad Request: Missing MAC address in the request parameters.
- 400 – Bad Request: Missing start date in the request parameters.
- 400 – Bad Request: Missing end date in the request parameters.
- 400 – Bad Request: Wrong format for start date in the request parameters.
- 400 – Bad Request: Wrong format for end date in the request parameters.
- 400 – Bad Request: End date smaller or equal than start date.
- 404 – Not Found: MAC address not found in the database.

Response Schema: HistoricalData

Response Example:

```
{
  mac_address: "0xf0b61e0bfe09",
  timestamp: [1664630309530, 1664630310567, 1664630311542],
  temperature: [22, 22, 23],
  humidity: [60, 61, 59],
}
```

2.2 REST Client for Data Visualization (1pt)

On Deepnote, create a Python notebook (*rest_client.ipynb*) to develop a REST client that sequentially executes the following actions:

a) Check the server status (using **GET /status**).
b) Add your RPI sensor node (using **POST /sensors**).
c) Retrieve the temperature & humidity data collected by your RPI from a given data range (using the endpoint implemented in 2.1).
d) Using Deepnote chart blocks, plot temperature & humidity data using two separate line plots:
   i. Timestamp (x-axis) vs. Temperature (y-axis).
   ii. Timestamp (x-axis) vs. Humidity (y-axis).

2.3 Reporting (1pt)

In the report, specify the most suitable HTTP method (GET, POST, PUT, or DELETE) that has been used for implementing the required functionalities for the **/data/{mac_address}** endpoint and motivate your answer.

Deliverables

- A Deepnote notebook named *rest_server.ipynb* that contains the code of 2.1. The notebook is intended to be run on Deepnote, must use only the packages that get installed with the *ML4IoT Template* and must run without any additional dependency.
- A Deepnote notebook named *rest_client.ipynb* that contains the code of 2.2. The notebook is intended to be run on Deepnote, must use only the packages that get installed with the *ML4IoT Template* and must run without any additional dependency.