Machine Learning for IoT

Homework 3

*****DUE DATE: 1 Feb (h23:59)*****

Submission Instructions:

Each group will send an e-mail to andrea.calimera@polito.it and valentino.peluso@polito.it (in cc) with subject *ML4IOT24 TeamN* (replace *N* with the team ID). Attached with the e-mail, a single ZIP archive (*.zip*) named *HW3_TeamN.zip* (replace *N* with the team ID) containing the following files:

1. The deliverables specified in the text of the exercises.
2. One-page pdf report, titled *TeamN_Homework3.pdf*, organized in different sections (one for each exercise). Each section should motivate the main adopted design choices.

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)

In VS Code, develop a Python script that monitors the battery status (battery level and power plugged) of your PC and publishes collected data to an MQTT broker. Continuously monitor the battery status with a sampling rate of approximately 1 second. Publish collected data every time 10 consecutive records have been collected, streaming only the last 10 records in each message.
The message must be sent in JSON format and must contain the parameters reported in the following table:

Parameter	Description
mac_address	string, the MAC address of the PC network card (use *hex(uuid.getnode())*).
events	list of *Event* items

where the *Event* item consists of the following parameters:

Parameter	Description
timestamp	integer, the timestamp in milliseconds.
battery_level	integer, the battery level in percentage.
power_plugged	integer, a flag indicating whether the power supply is plugged (1) or not (0).

Example:

```
{
  mac_address: "0xf0b61e0bfe09",
  events: [
    {
      timestamp: 1664630309530,
      battery_level: 90,
      power_plugged: 1,
    },
    {
      timestamp: 1664630310567,
      battery_level: 90,
      power_plugged: 0,
    },
    {
      timestamp: 1664630311542,
      battery_level: 89,
      power_plugged: 0,
    },
    …
  ]
}
```

Use the student ID of a member of your team as message topic (e.g., *s001122*).
Use the message broker provided by eclipse (mqtt.eclipseprojects.io at port 1883).

Run the script from the PCs of different team members (always use the same topic) to emulate a fleet of connected devices.

1.2 Data Storage (1pt)
In Deepnote, create a new Python notebook to develop an MQTT subscriber that receives the messages with battery status data of your PCs and store the received data in Redis. For data storage, define two Redis TimeSeries for each PC, called *mac_address:*battery and *mac_address:*power, respectively (e.g., *0xf0b61e0bfe09:battery* and *0xf0b61e0bfe09:power*), where *mac_address* is the MAC address of the corresponding PC.

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 a laptop and must use only the packages that get installed with *requirements.txt* provided during the labs. Moreover, the script should contain all the methods needed for its correct execution.
- A Python notebook named *subscriber.ipynb* that contains the code of 1.2. The code is intended to be run on Deepnote, must use only the packages that get installed with *requirements.txt* uploaded on the workspace and must run without any additional dependency.

Exercise 2: Data Management & Visualization (3pts)

2.1 REST Server (1pt)
In Deepnote, create a new Python notebook to develop a REST Server that enables users to retrieve and manage the collected battery data. The API must be compliant with the following specifications:

RESOURCES

Resource: BatteryStatus

Parameter	Description
mac_address	string, the MAC address of the PC network card
timestamps	list of integers, each integer is the timestamp in milliseconds
battery_levels	list of integers, each integer is the battery level in percentage
power_plugged	list of integers, each integer indicates whether the power supply is plugged (1) or not (0).

Example:

```
{
  mac_address: "0xf0b61e0bfe09",
  timestamps: [1664630309530, 1664630310567, 16646303115 42],
  battery_levels: [90, 90, 89],
  power_plugged: [1, 0, 0],
}
```

ENDPOINTS

Endpoint /devices

- ____ **/devices** (fill the form with the most appropriate HTTP method)
  Description: Retrieve the list of MAC addresses of the monitored devices.

  Path Parameters: N/A

  Query Parameters:

Parameter	Description
blt	int (optional), an integer representing the battery level threshold. If specified, return devices with a battery level lower or equal than the specified threshold in the last recorded timestamp. Valid values range from 0 to 100. If not provided, devices with any battery level will be returned.
plugged	int (optional), an integer flag indicating whether to return only devices with power plugged (1) or unplugged (0) in the last recorded timestamp. If not provided, devices with any power state will be returned.

  Response Status Code:
  - 200 – OK: Everything worked as expected.
  - 400 – Bad Request: invalid battery threshold value.
  - 400 – Bad Request: invalid plugged value.

  Response Schema:

Parameter	Description
mac_addresses	list of strings, each string is a MAC address.

  Response Example:

```
{
  mac_addresses: ["0xf0b61e0bfe09", "0xf0b41e2abe15"]
}
```

Endpoint /device/{mac_address}

- ____ **/device/{mac_address}** (fill the form with the most appropriate HTTP method)
  Description: Retrieve battery status information of 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 device to retrieve.

  Query Parameters:

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

Response Status Code:
- 200 – OK: Everything worked as expected.
- 400 – Bad Request: missing MAC address.
- 400 – Bad Request: missing start date.
- 400 – Bad Request: missing end date.
- 400 – Bad Request: wrong format for start date.
- 400 – Bad Request: wrong format for end date.
- 400 – Bad Request: end date smaller or equal than start date.
- 404 – Not Found: invalid MAC address.

Response Schema: BatteryStatus

Response Example:

```
{
  mac_address: "0xf0b61e0bfe09",
  timestamps: [1664630309530, 1664630310567, 1664630311542],
  battery_levels: [90, 90, 89],
  power_plugged: [1, 0, 0],
}
```

- **____ /device/{mac_address}/** (fill the form with the most appropriate HTTP method)
  Description: Delete the timeseries associated to the specified MAC address.

  Path Parameters:

Parameter	Description
mac_address	string (required), the MAC address of the device to delete.

  Query Parameters: N/A

  Response Status Code:
  - 200 – OK: Everything worked as expected.
  - 400 – Bad Request: missing MAC address.
  - 404 – Not Found: invalid MAC address.

  Response Schema: N/A

2.2 REST Client for Data Visualization (1pt)

In Deepnote, create a Python notebook to develop a REST client that sequentially executes the following actions:

a) Retrieve and print the list of all monitored devices.
b) For each device, retrieve the battery status from a given date range. Plot with a bar chart the number of records with power plugged and unplugged.
c) Retrieve and print the list of devices with battery level lower or equal than 25% and power unplugged in the last recorded timestamp.
d) Delete the data of the first device in the list of all monitored devices.

2.3 Reporting (1pt)

In the report, fill the following table selecting the most suitable HTTP method (GET, POST, PUT, or DELETE) for each row and motivate your answer.

Method	Endpoint	Description
	/devices	Retrieve the list of MAC addresses of the monitored devices.
	/device/{mac_address}	Retrieve battery status information of the device with the specified MAC address in the specified date range.
	/device/{mac_address}	Delete the timeseries associated to the specified MAC address.

Deliverables

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