Model Training and Evaluation

Water Consumption Monitoring System in public places using IoT, Python, and machine learning with complex models involves several steps, including data collection, preprocessing, model training, and deployment. Here's an example of how you can implement this using a more complex machine learning model, such as a Random Forest regressor, to predict water consumption based on sensor data.

Step 1: Data Collection (IoT Device):

Assuming you have sensors collecting data (such as flow rates, pressure, temperature, etc.), you need to send this data to a server for processing. Below is an example of an MQTT publisher code on a Raspberry Pi to send water consumption data to a server.

Code:

```
import paho.mqtt.client as mqtt

import random

import time
```

```
# MQTT Configurations
mqtt_broker = "BROKER_IP_ADDRESS"
mqtt_port = 1883
mqtt_topic = "water_consumption"

# MQTT Client Setup
client = mqtt.Client()

# Connect to MQTT Broker
client.connect(mqtt_broker, mqtt_port, 60)

try:
    while True:
        # Simulate water consumption data (replace this with actual sensor data)
        flow_rate = random.uniform(1, 10)  # Random flow rate between 1 and 10
        pressure = random.uniform(20, 80)  # Random pressure between 20 and 80
```

```
        # Publish water consumption data to MQTT topic
        client.publish(mqtt_topic, f"{flow_rate},{pressure}")


        time.sleep(1)  # Send data every 1 second (adjust as needed)
except KeyboardInterrupt:
    client.disconnect()  # Disconnect from MQTT broker on program exit
```

we can also load a dataset.

Dataset Structure:

A typical dataset for this project might have the following columns:

- Flow Rate (in liters per minute): The rate at which water flows through the sensor.
- Pressure (in psi or any appropriate unit): The pressure of the water.
- Temperature (in Celsius or Fahrenheit): The temperature of the water, if applicable.
- Consumption (in liters): The actual water consumption corresponding to the given flow rate, pressure, and temperature.

1	1	8.109848	21.899193	10.076577	187.675701
2	2	1.599232	41.779820	26.738010	93.553623
3	3	5.484721	27.653086	22.390789	174.060260
4	4	3.360401	39.923447	13.970097	148.128873
5	5	3.163135	49.534574	21.546262	178.230790
6	6	3.418357	50.723772	18.213492	191.605457
7	7	8.675133	20.841508	23.942162	204.745015
8	8	8.675425	41.714684	11.897535	373.790129
9	9	2.188548	79.492024	20.749485	194.721562
10	10	9.744849	40.992073	18.390596	417.852174
11	11	3.688883	64.573214	17.725160	255.928163
12	12	9.897592	30.342402	15.000898	315.317623
13	13	5.514285	43.650009	20.023466	260.722047
14	14	3.510278	55.898336	17.115460	213.334149
15	15	6.970096	25.529082	26.107737	204.047893
16	16	1.685584	79.475776	10.400206	144.363284
17	17	5.812023	21.566796	26.944403	152.291125
18	18	2.404278	48.481246	24.861101	141.423502
19	19	4.778020	70.546973	21.489230	358.564088
20	20	2.810582	67.326746	16.860993	206.088300
21	21	8.318015	71.762634	12.327594	609.250243
22	22	9.793000	22.476366	26.618612	246.729655
23	23	4.532489	39.157494	16.744304	194.225227
24	24	2.387880	27.603965	23.229042	89.144001
25	25	1.004690	74.946022	12.199855	87.497386

Step 2: Data Processing:

Feature Extraction:

Received data (flow rate, pressure, temperature) is stored in the data dictionary.

The features (flow_rate, pressure, temperature) are extracted from the received data and stored in separate lists within the data dictionary.

Code:

```
data = {'flow_rate': [], 'pressure': [], 'temperature': [], 'consumption': []}

# Extract features and store in data dictionary

flow_rate, pressure, temperature = map(float, message.payload.split(','))

data['flow_rate'].append(flow_rate)

data['pressure'].append(pressure)

data['temperature'].append(temperature)
```

Target Variable Calculation:

Water consumption is calculated using a simple formula based on the extracted features.

The calculated water consumption values are stored in the data dictionary as the target variable (consumption).

Code:

```
# Calculate water consumption based on features and store in data dictionary

consumption = flow_rate * pressure * temperature * 0.01
# Example calculation for water consumption

data['consumption'].append(consumption)
```

Step 3:Data Preparation:

The features are organized into a Pandas DataFrame called features.

The target variable (consumption) is separated from the features to create the target variable.

Code:

features = pd.DataFrame(data) # Features (flow_rate, pressure, temperature)

target = features.pop('consumption') # Target variable (water consumption)

Machine Learning:

Standardization:

Features are standardized using StandardScaler to ensure all features have the same scale, which is important for many machine learning algorithms.

Code:

from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()

features = scaler.fit_transform(features)

Step 4:Data Splitting:

The data is split into training and testing sets. 80% of the data is used for training (X_train, y_train), and 20% is used for testing (X_test, y_test).

Code:

```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(features, target, test_size=0.2, random_state=42)
```

Step 5:Model Initialization and Training:

A Random Forest Regressor model is initialized with 100 estimators and trained using the training data.

python

Code:

```
from sklearn.ensemble import RandomForestRegressor

model = RandomForestRegressor(n_estimators=100, random_state=42)

model.fit(X_train, y_train)
```

Step 6:Model Evaluation:

Predictions are made using the test data, and Mean Squared Error (MSE) is calculated to evaluate the model's performance.

Code:

```
from sklearn.metrics import mean_squared_error

predictions = model.predict(X_test)

mse = mean_squared_error(y_test, predictions)

print("Mean Squared Error: ", mse)
```

In summary, the code processes real-time data by extracting features, calculates the target variable (water consumption), standardizes the features, and splits the data into training and testing sets. It then initializes a Random Forest Regressor model, trains the model with the training data, makes predictions on the test data, and evaluates the model using Mean Squared Error. The model is ready to be deployed for real-time predictions after this step.