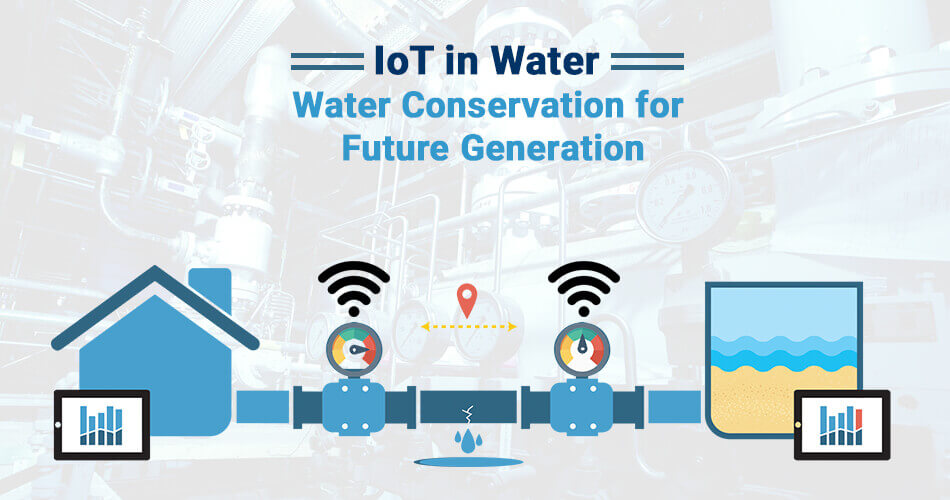
SMART WATER SYSTEM-IOT

Team Member

au312821106015: R.R.Naviya

Phase3: Document Submission



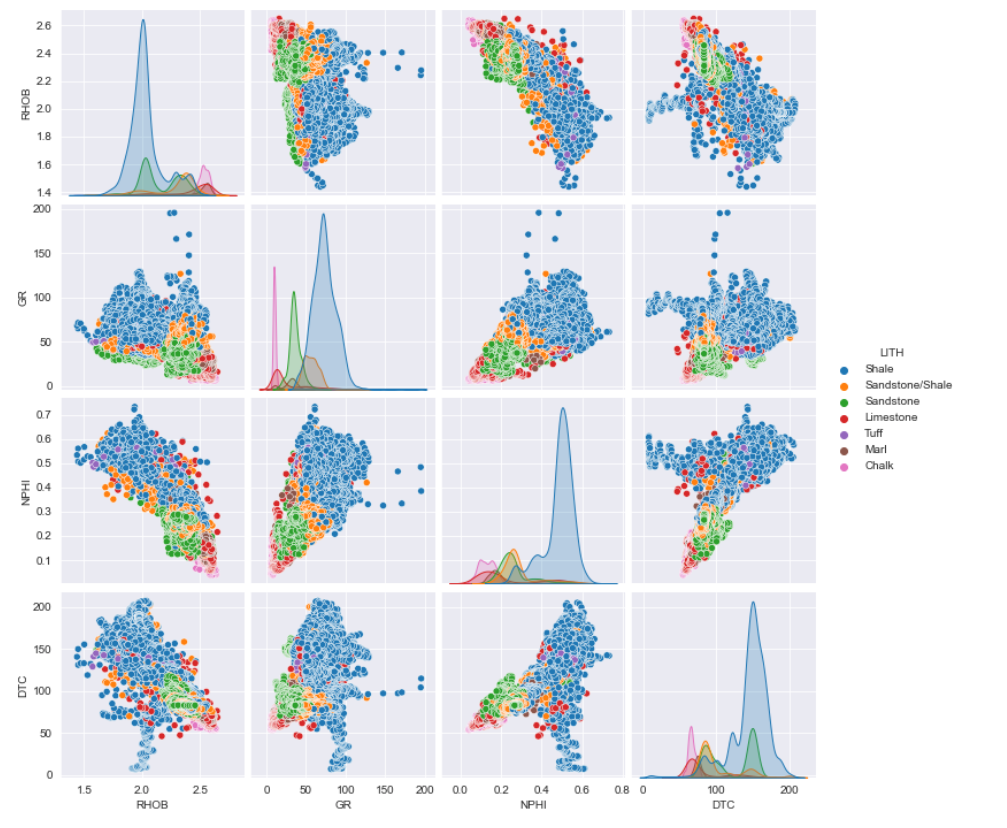
Introduction:

Smart water management involves the intelligent planning, monitoring, and optimization of water-related processes to ensure the efficient and sustainable use of this precious resource. It plays a pivotal role in environmental conservation, addressing water scarcity issues, and enhancing overall water quality. The integration of technology, data analytics, and proactive strategies is key to achieving smart water management goals.

Introduction to IOT Smart water system with python:

A smart water system using Python is a versatile solution that uses Python programming to connect with sensors, process data, and control water-related processes. Python facilitates data collection from sensors, IoT connectivity, and the development of user interfaces. It also enables data analysis and the implementation of control systems. Python's adaptability and extensive libraries make it a powerful choice for creating an efficient and integrated smart water system.

DATA PREPARATION:

* Data preparation for a smart water system involves several steps to ensure that data is collected, cleaned, and analyzed effectively.
*  Data preparation is a critical part of a smart water system, as it lays the foundation for meaningful analysis and decision-making to improve water quality, efficiency, and sustainability.

Loading and preprocessing datasets in smart water system using IOT:

1. Data Collection: Collect data from sensors and other sources. This data may include water quality readings, flow rates, pressure, and more. Ensure the data is in a format that can be easily processed using Python.

2. Loading Data: Use Python libraries like `pandas` to load the dataset into your Python environment.

3. Data Exploration: Examine the dataset to understand its structure and contents. Check for missing values, outliers, and anomalies that may need preprocessing. Use functions like `info()`, `describe()`, and `head()` to get an overview of the data.

4. Data Cleaning: Clean the data by handling missing values and outliers. You can use methods to deal with missing data, and `z-score` or interquartile range (IQR) to identify and

remove outliers.

5. Data Transformation: Depending on your specific needs, you might need to transform the data. For example, you can convert timestamps to datetime objects, normalize numerical values, or one-hot encode categorical variables.

6. Feature Engineering: Create new features or derive insights from the existing data if necessary. Feature engineering can improve the performance of machine learning models later on.

7. Data Splitting: If you plan to use the data for training and testing machine learning models, split the dataset into training and testing sets using train test split from `scikit-learn`.

8. Scaling and Standardization: Depending on the machine learning algorithms you intend to use, you might need to scale or standardize the data to ensure that all features have the same scale.

9. Save Preprocessed Data: Save the preprocessed data to a new file if needed, so that you can use it for analysis, modeling, and further development of your smart water system.

This preprocessing prepares your dataset for analysis, modeling, and any other tasks necessary for your smart water system. The specific steps you take will depend on the nature of your dataset and the objectives of your system.

Python program:

# Import necessary libraries

import pandas as pd

# Load the dataset (simulated example)

# Replace 'your\_dataset.csv' with the actual path to your dataset

Dataset path = 'your\_dataset.csv'

data = pd .read csv(dataset path)

# Data Preprocessing

# Example: Drop missing values, convert data types, and perform other preprocessing steps

data = data.dropna() # Remove rows with missing values

data['timestamp'] = pd.to\_datetime(data['timestamp']) # Convert timestamp to datetime

data.set\_index('timestamp', inplace=True) # Set timestamp as the index

# You can add more preprocessing steps based on your specific data and requirements

# IoT Data Analysis

# Perform data analysis or any specific operations related to your smart water system here

# For example, calculate statistics, visualize data, or apply machine learning algorithms

# Save the preprocessed dataset (if needed)

# Replace 'preprocessed\_dataset.csv' with the desired filename

data.to\_csv('preprocessed\_dataset.csv')

# Additional actions as per your requirements

# End of the program

Data Acquisition:

Creating a Python program for data acquisition in a smart water system involves various components, such as sensors, communication protocols, and data storage. Here's a simplified example using Python to acquire data from a hypothetical water level sensor and save it to a CSV file. Please note that in a real smart water system, you would likely use more advanced hardware and data storage solutions.

In this example:

Read water level function simulates data acquisition from a water level sensor (replace it with actual sensor communication code).

Data is collected with timestamps and saved to a CSV file (water\_level\_data.csv) for simplicity.

The program runs in an infinite loop, acquiring data at regular intervals (here, every 60 seconds).

Adjust the data acquisition and storage components according to the specific sensors and data storage mechanisms used in your smart water system.

Remember that in a real-world application, you may need to consider error handling, data preprocessing, and data transmission to a central server or cloud for comprehensive smart water system functionality.

Program for Data Acquisition:

import time

import csv

# Simulated water level sensor function (replace with actual sensor code)

def read\_water\_level():

# Replace this with actual sensor data acquisition logic

return 50.0 # Simulated water level in centimeters

# Data acquisition and storage

def main():

while True:

# Read water level from the sensor

water\_level = read\_water\_level()

# Get the current timestamp

timestamp = time.strftime("%Y-%m-%d %H:%M:%S")

# Store data in a CSV file

with open('water\_level\_data.csv', 'a', newline='') as csv\_file:

csv\_writer = csv.writer(csv\_file)

csv\_writer.writerow([timestamp, water\_level])

# Print the data for verification (you can replace this with actual data handling)

print(f"Timestamp: {timestamp}, Water Level: {water\_level} cm")

PYTHON SCRIPT:

python

import paho.mqtt.client as mqtt

import time

import json

# Define the MQTT broker and topic

mqtt\_broker = "your\_mqtt\_broker\_address"

mqtt\_topic = "water\_usage\_data"

# Simulated water usage data

water\_data = {

"location": "Sensor A",

"flow\_rate": 5.3,

"pressure": 30,

"timestamp": int(time.time())

}

# Callback when the client connects to the MQTT broker

def on\_connect(client, userdata, flags, rc):

print("Connected to MQTT broker with result code " + str(rc))

# Initialize the MQTT client

client = mqtt.Client()

client.on\_connect = on\_connect

# Connect to the MQTT broker

client.connect(mqtt\_broker, 1883, 60)

while True:

# Simulate collecting real-time water usage data

# Replace this with actual data collection from water sensors

water\_data["flow\_rate"] = water\_data["flow\_rate"] + 0.2

water\_data["timestamp"] = int(time.time())

# Publish the water usage data to the MQTT topic

client.publish(mqtt\_topic, json.dumps(water\_data))

print(f"Published: {water\_data}")

# Adjust the time interval for data updates as needed

time.sleep(10) # Update data every 10 seconds

# Keep the script running

client.loop\_forever()

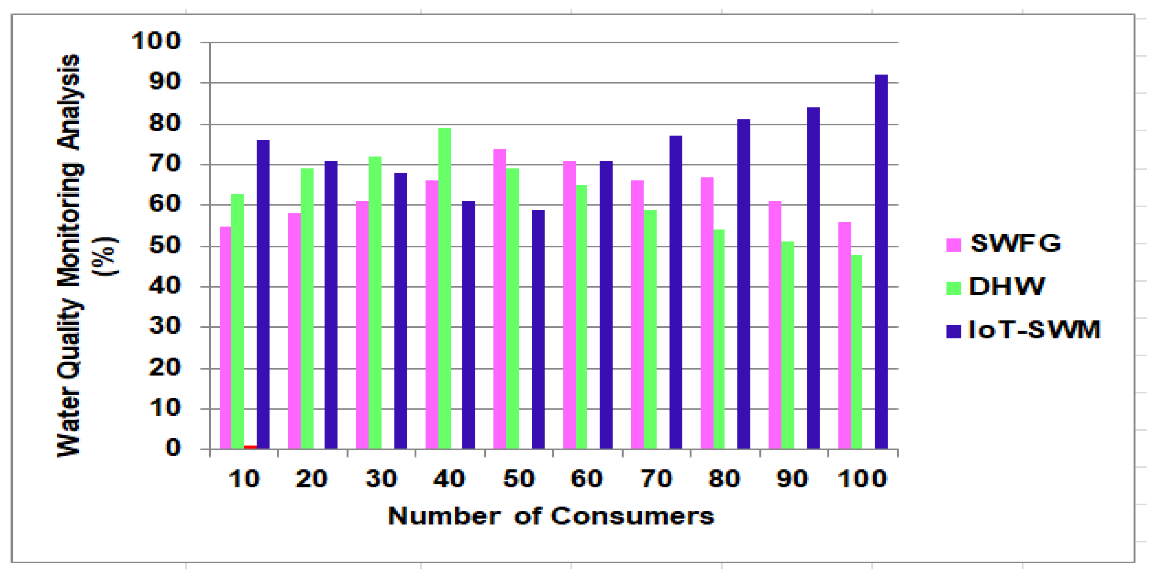
Visualization of analytics results:

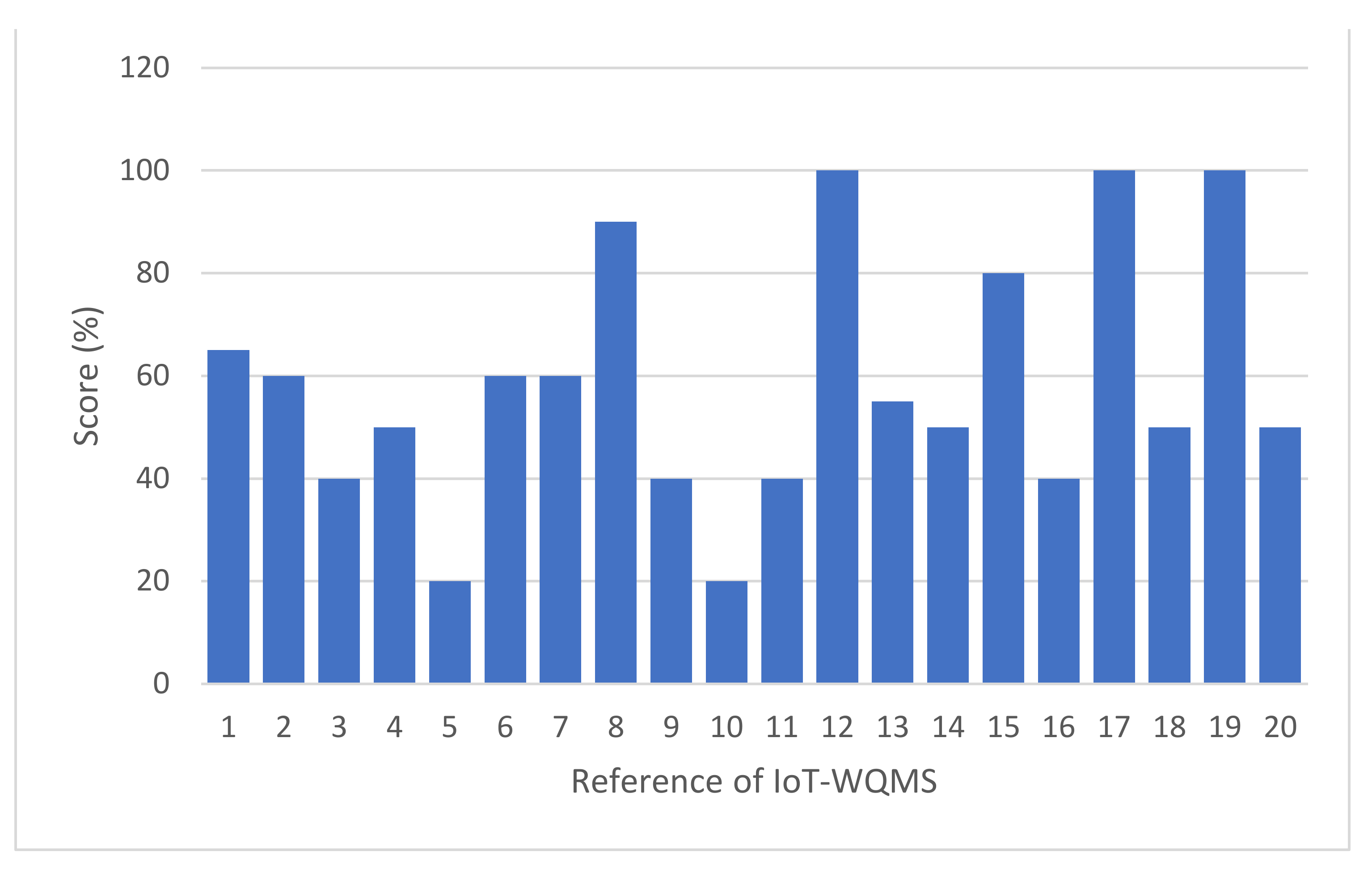
Visualizing analytics results for a smart water system can provide valuable insights and aid in decision-making.

A bar plot:

Compare data from different sensors or parameters using bar charts. For example, you can compare water quality at various locations.

Choose visualization techniques based on the specific data and goals of your smart water system analytics. Interactive and user-friendly dashboards can make it easier for stakeholders to interpret the results and make informed decisions.





DATA ANALYSIS FOR SMART WATER SYSTEM

Conclusion:

a smart water system leveraging IoT technology offers a transformative approach to water resource management. By seamlessly integrating sensors, data analytics, and real-time connectivity, it empowers stakeholders to monitor water quality, detect issues, and make data-driven decisions. This system fosters efficient water distribution, reduces waste, and conserves resources, making it a critical tool in addressing global water challenges. It not only ensures clean and accessible water for communities but also supports sustainability and cost savings. As IoT continues to advance, smart water systems will become even more instrumental in securing a resilient and eco-conscious water future, marking a significant step towards a more sustainable and efficient water infrastructure.