SMART WATER SYSTEM

622621121053: M.SUDHAGAR

PROJECT SUBMISSION PHASE 2

Topic: innovation

Project: smart water system

INTRODUCTION :

Smart water management systems can provide a more resilient and efficient water supply system, reducing costs and improving sustainability. High-technology solutions for the water sector include digital meters and sensors, supervisory control and data acquisition (SCADA) systems, and geographic information systems (GIS).

This explainer is adapted from proceedings of a workshop conducted by the Asian Development Bank (ADB) in Tashkent, Uzbekistan for the water sector. The workshop introduced smart systems and focused on remote monitoring of water networks using smart meters **and other instruments**

Project:

```
<!DOCTYPE html>
<html>
<head>
  <title>Water Consumption Analysis</title>
  <!—Include necessary CSS and JavaScript files →
  <link rel=”stylesheet” type=”text/css” href=”styles.css”>
  <script src=”script.js”></script>
</head>
<body>
  <h1>Water Consumption Analysis</h1>
  <p>Enter your water consumption data:</p>

  <!—Input form for water consumption data →
  <form id=”waterConsumptionForm”>
    <label for=”month”>Month:</label>
```

```
    <select id="month" name="month">
      <!—Add options for months here →
    </select><br>
    <label for="usage">Monthly Usage (in gallons):</label>
    <input type="number" id="usage" name="usage" required><br>
    <button type="submit">Submit</button>
  </form>

  <!—Display conservation suggestions here →
  <div id="suggestions">
    <!—Suggestions will be populated using JavaScript →
  </div>

  <script>
    // JavaScript code for handling form submission and machine learning integration
    Document.getElementById("waterConsumptionForm").addEventListener("submit",
function(event) {
      Event.preventDefault();

      // Get user input data
      Const month = document.getElementById("month").value;
      Const usage = parseFloat(document.getElementById("usage").value);

      // Send the data to a server for machine learning analysis
      // You would typically use AJAX or Fetch API to send a request to a server

      // Example: Assuming a server endpoint for analysis
      Fetch('/analyze', {
        Method: 'POST',
```

```
        Body: JSON.stringify({ month, usage }),
        Headers: {
          'Content-Type': 'application/json'
        }
      })
      .then(response => response.json())
      .then(data => {
        // Display the conservation suggestion received from the server
        Document.getElementById("suggestions").innerHTML = `<p><strong>Suggestion for ${month}:</strong> ${data.suggestion}</p>`;
      })
      .catch(error => {
        Console.error('Error:', error);
      });
    });
  </script>
</body>
</html>
```

MACHINE LEARNING ALGORITHM:

1. **Data Collection: Gather historical water consumption data, which may include daily, monthly, or hourly readings from water meters.**

2. **Data Preprocessing: Clean and prepare the data by handling missing values, outliers, and formatting it for analysis.**

3. **Feature Engineering: Create relevant features from the data, such as day of the week, holidays, weather data, or other contextual information that might influence water consumption.**

4. **Model Selection: Choose a suitable machine learning algorithm based on the nature of your problem. Time-series forecasting methods like ARIMA, Prophet, or machine learning algorithms like Random Forest, LSTM, or XGBoost can be effective.**

5. **Model Training: Train the selected model on historical data. Split the data into training and validation sets to evaluate the model's performance.**

6. **Hyperparameter Tuning: Optimize the model's hyperparameters to improve its accuracy.**

7. **Model Evaluation: Evaluate the model's performance using metrics like Mean Absolute Error (MAE), Mean Squared Error (MSE), or Root Mean Squared Error (RMSE).**

8. **Deployment: Once the model performs satisfactorily, deploy it to analyze real-time or future water consumption data.**

9. **Monitoring and Maintenance: Continuously monitor the model's performance and update it as new data becomes available.**

10. **Insights and Recommendations: Use the model's predictions to gain insights into water consumption patterns, identify anomalies, and make recommendations for water management and conservation.**

Result:

```
# Import necessary libraries
Import pandas as pd
From sklearn.model_selection import train_test_split
From sklearn.ensemble import RandomForestClassifier
From sklearn.metrics import accuracy_score
```

```
# Load your dataset
Data = pd.read_csv('your_dataset.csv')

# Preprocess your data (e.g., handle missing values, encode categorical variables)

# Split data into training and testing sets
X = data.drop('target_column', axis=1)
Y = data['target_column']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Initialize and train the machine learning model (e.g., Random Forest)
Model = RandomForestClassifier(n_estimators=100, random_state=42)
Model.fit(X_train, y_train)

# Make predictions on the test set
Y_pred = model.predict(X_test)

# Evaluate the model's performance (e.g., accuracy)
Accuracy = accuracy_score(y_test, y_pred)

# Print the result
Print("Accuracy:", accuracy)
```