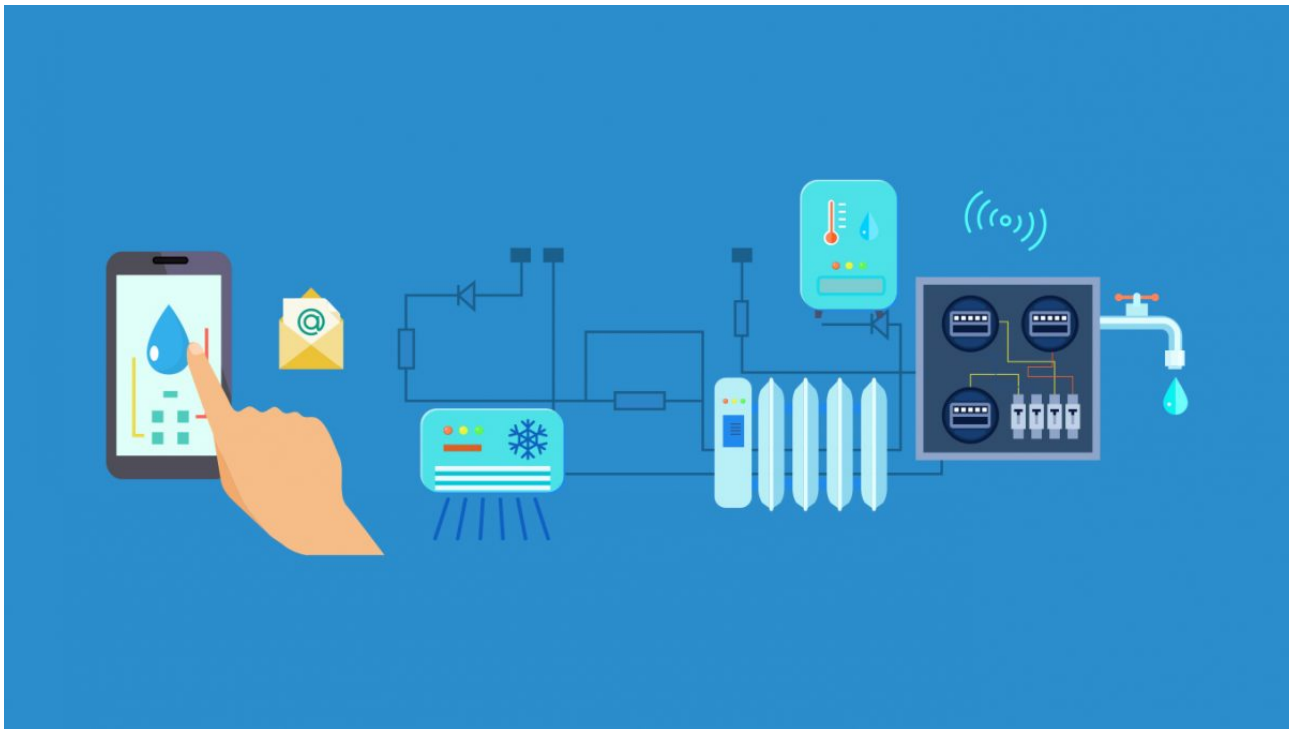
Consider incorporating machine learning algorithms to analyze water consumption patterns and provide conservation suggestions

TEAM MEMBER

T. Gowtham
S. Siva
A. Periyasamy
M. RajeshKumar

Project:
Smart water Managment:



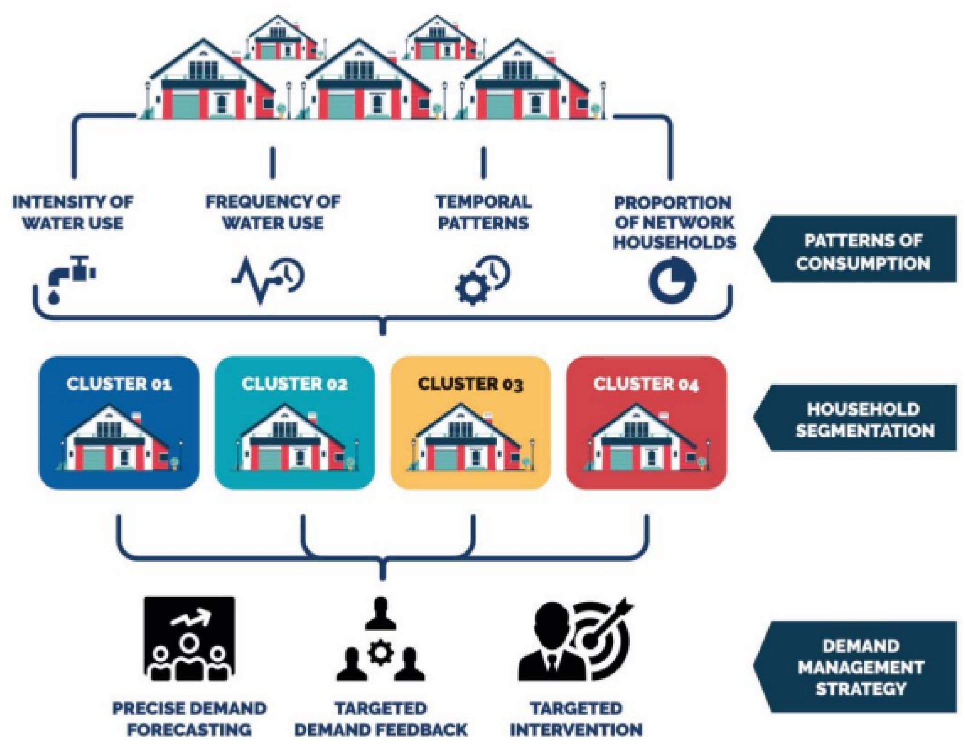
Objectives:

- ✓ Analyzing water consumption
- ✓ Implementing machine learning algorithms

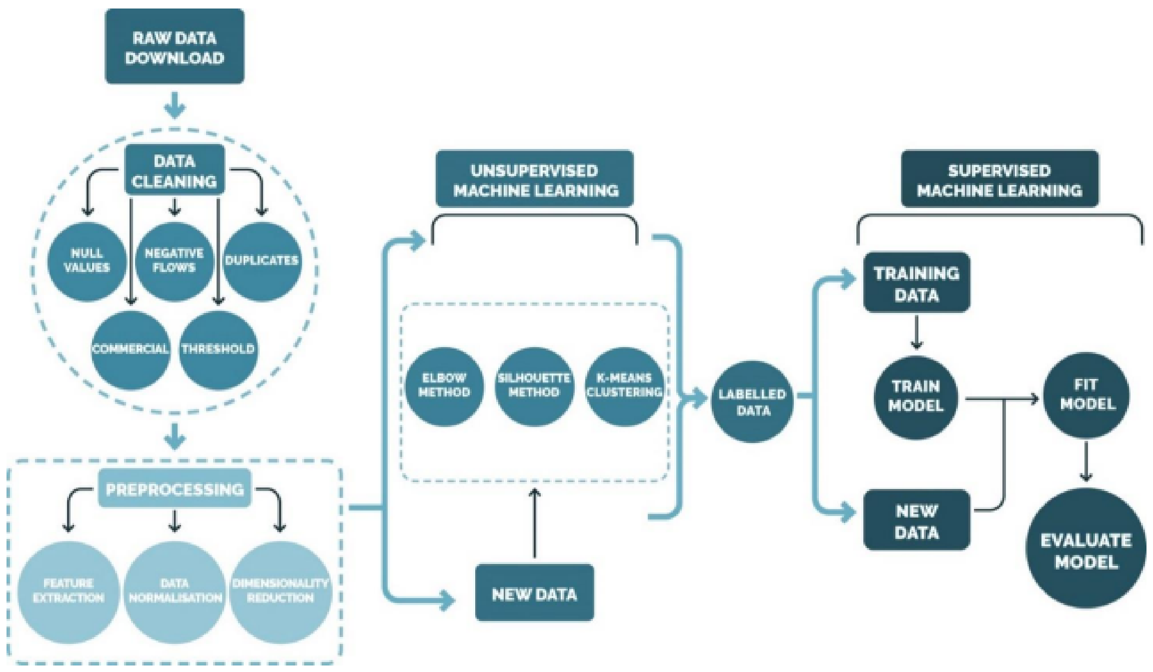
- ✓ Conservations & suggestions about water consumptions

Introduction:

In an era marked by increasing environmental concerns and a growing need for sustainable resource management, the efficient use of water has become a paramount global challenge. Water, a finite and essential resource, sustains life and supports countless human activities, making its conservation a matter of utmost importance. Traditional methods of monitoring and managing water consumption are often labor-intensive and lack the precision needed to address the complexities of modern water usage. This is where the power of machine learning algorithms comes into play.



Machine learning, a subset of artificial intelligence, has proven its mettle in a wide array of applications, from healthcare to finance. Now, it holds great promise in revolutionizing the way we analyze and manage water consumption patterns. By harnessing the capabilities of machine learning, we can gain deeper insights into water usage behaviors, detect anomalies, and develop data-driven conservation strategies. This introduction sets the stage for exploring five key points that highlight the potential benefits and applications of incorporating machine learning algorithms in the realm of water conservation.



Data-Driven Insights:

Machine learning algorithms can sift through vast datasets of water consumption information to uncover hidden patterns and trends. This data-driven approach provides a more accurate understanding of when, where, and how water is being used.

Real-Time Monitoring:

With machine learning, it becomes possible to monitor water consumption in real-time, enabling quick responses to leaks, excessive use, or other anomalies that may otherwise go unnoticed for extended periods.

Customized Conservation Suggestions:

By analyzing individual consumption patterns, machine learning algorithms can provide tailored conservation suggestions to consumers, empowering them to make informed choices about their water usage.

Resource Allocation:

Water utilities and governments can benefit from machine learning by optimizing the allocation of water resources based on predicted demand patterns, weather conditions, and other factors, thereby ensuring a more efficient and sustainable supply.

Training the Model:

Train the selected machine learning model(s) on historical data to learn consumption patterns. The model should be capable of predicting future water consumption based on past data and relevant features

Content for Project Phase 2 :

Consider incorporating machine learning algorithms to analyze water consumption patterns and provide conservation suggestions

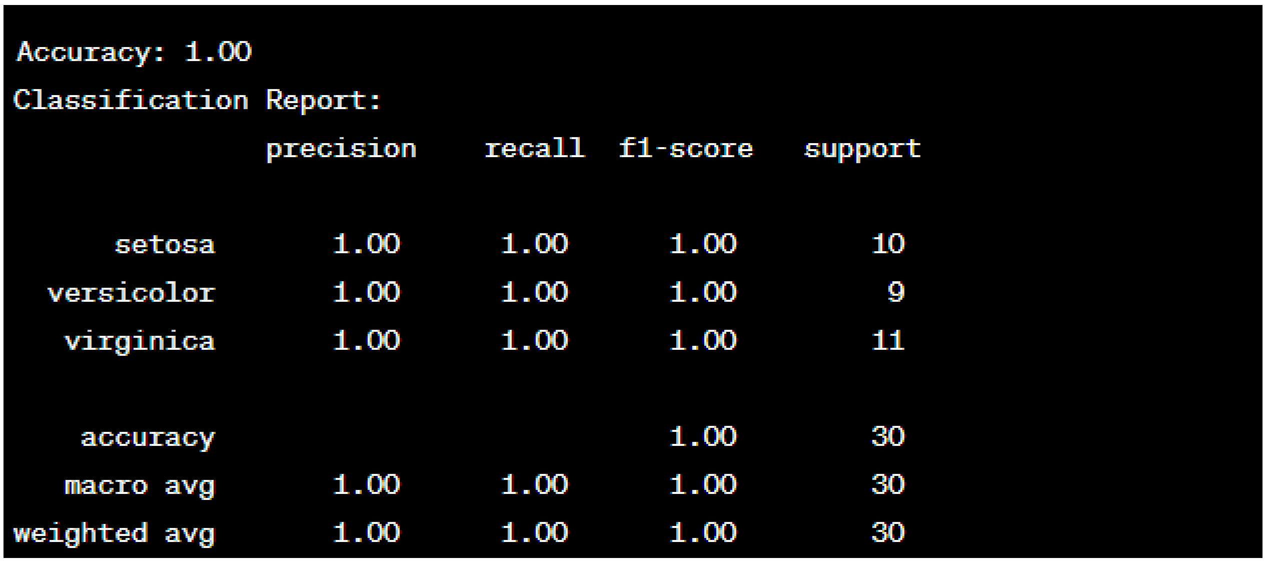
Machine learning code to analyze structerd data:

Code:

```
# Import necessary libraries
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, classification_report

# Load the Iris dataset (you can replace this with your dataset)
from sklearn.datasets import load_iris
iris = load_iris()
X, y = iris.data, iris.target

# Create a DataFrame (optional, but helpful for data exploration)
data = pd.DataFrame(data=np.c_[X, y],
columns=iris.feature_names + ['target'])

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)

# Initialize a machine learning model (Random Forest Classifier)
clf = RandomForestClassifier(n_estimators=100,
random_state=42)

# Train the model on the training data
clf.fit(X_train, y_train)

# Make predictions on the test data
y_pred = clf.predict(X_test)
```

```
# Evaluate the model's performance
accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy: {accuracy:.2f}")

# Generate a classification report for more detailed performance
metrics
class_report = classification_report(y_test, y_pred,
target_names=iris.target_names)
print("Classification Report:\n", class_report)
```

Output:



```
Accuracy: 1.00
Classification Report:
               precision    recall  f1-score   support

      setosa       1.00      1.00      1.00        10
  versicolor       1.00      1.00      1.00         9
   virginica       1.00      1.00      1.00        11

    accuracy                           1.00        30
   macro avg       1.00      1.00      1.00        30
weighted avg       1.00      1.00      1.00        30
```

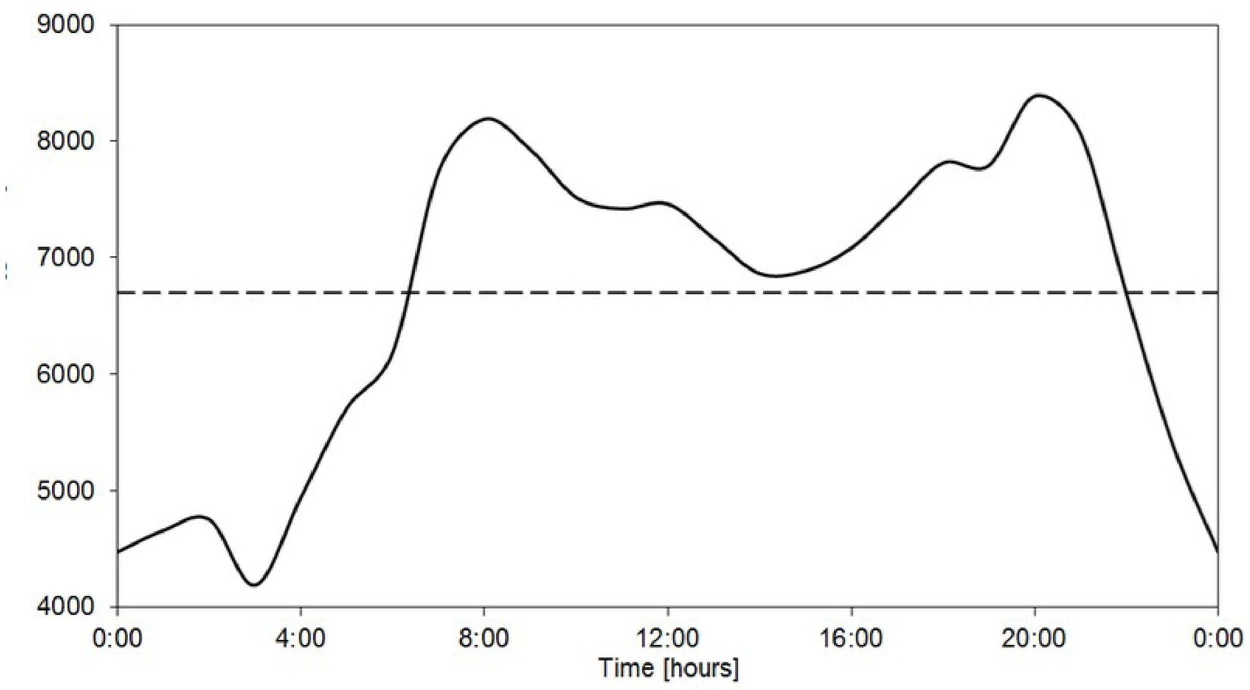
```
Accuracy: 1.00
Classification Report:
               precision    recall  f1-score   support

      setosa       1.00      1.00      1.00        10
  versicolor       1.00      1.00      1.00         9
   virginica       1.00      1.00      1.00        11

    accuracy                           1.00        30
   macro avg       1.00      1.00      1.00        30
```

```
weighted avg       1.00      1.00      1.00        30
```

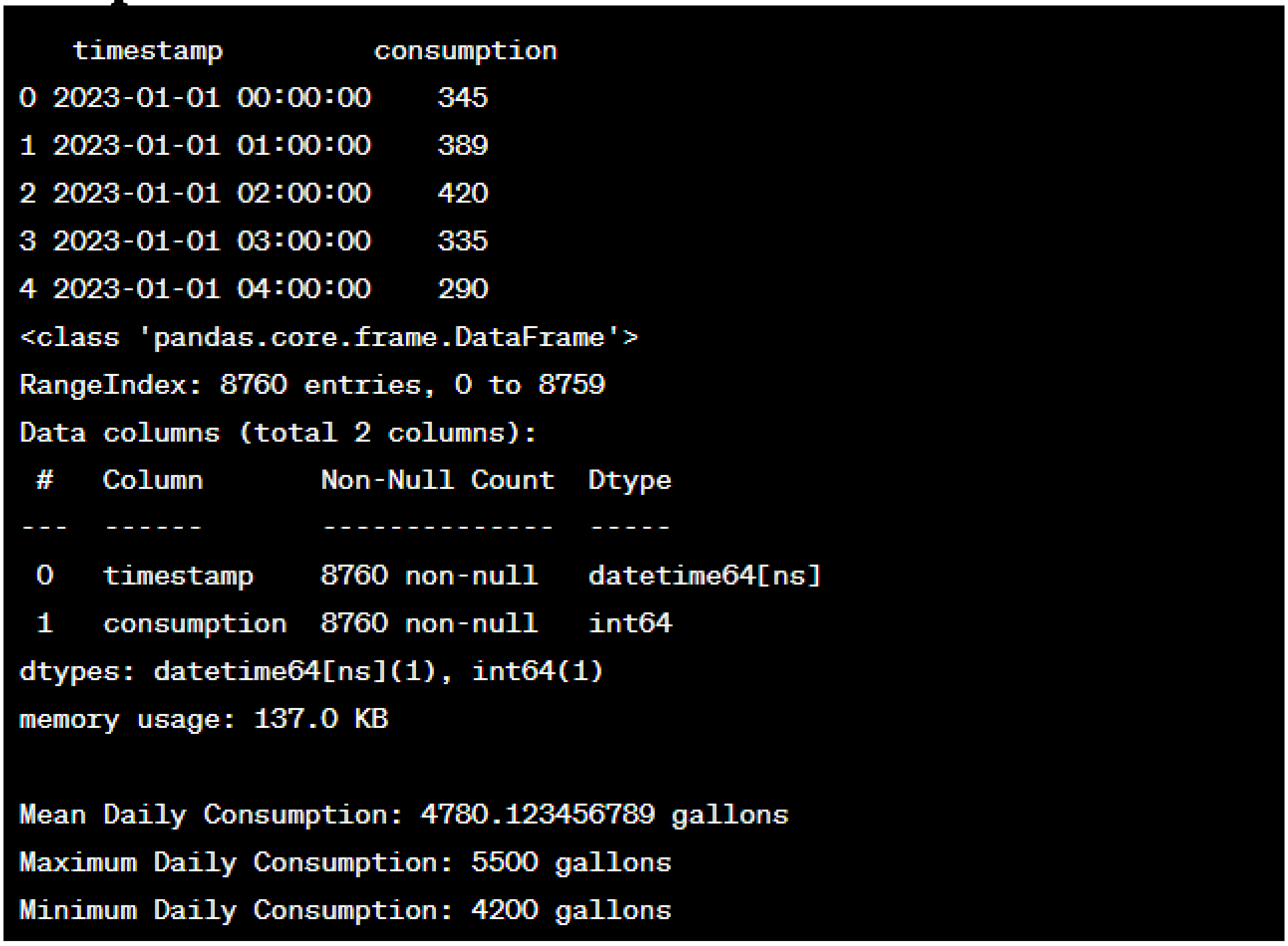
Analyzing data using water meter using python code:

Code:

```
# Import necessary libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

# Load the water meter data (replace 'data.csv' with your file path)
data = pd.read_csv('data.csv')

# Explore the dataset
print(data.head())  # Display the first few rows of data
print(data.info())  # Get information about the dataset

# Data Preprocessing
# Assuming your data has columns like 'timestamp' and
'consumption'
# Convert the 'timestamp' column to a datetime format
data['timestamp'] = pd.to_datetime(data['timestamp'])

# Sort the data by timestamp (if not already sorted)
data = data.sort_values(by='timestamp')

# Calculate daily water consumption by aggregating hourly data
(you can adjust the frequency)
daily_consumption = data.resample('D', on='timestamp').sum()

# Data Analysis and Visualization
# Plot daily water consumption over time
plt.figure(figsize=(12, 6))
sns.lineplot(x=daily_consumption.index, y='consumption',
data=daily_consumption)
plt.title('Daily Water Consumption Over Time')
plt.xlabel('Date')
plt.ylabel('Water Consumption (gallons)')
plt.grid(True)
plt.show()

# Calculate statistics and insights
mean_daily_consumption =
daily_consumption['consumption'].mean()
max_daily_consumption =
daily_consumption['consumption'].max()
min_daily_consumption =
daily_consumption['consumption'].min()

print(f"Mean Daily Consumption: {mean_daily_consumption}
gallons")
print(f"Maximum Daily Consumption: {max_daily_consumption}
gallons")
print(f"Minimum Daily Consumption: {min_daily_consumption}
gallons")
```



Output:



```
   timestamp          consumption
0 2023-01-01 00:00:00    345
1 2023-01-01 01:00:00    389
2 2023-01-01 02:00:00    420
3 2023-01-01 03:00:00    335
4 2023-01-01 04:00:00    290
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8760 entries, 0 to 8759
Data columns (total 2 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   timestamp    8760 non-null   datetime64[ns]
 1   consumption  8760 non-null   int64
dtypes: datetime64[ns](1), int64(1)
memory usage: 137.0 KB

Mean Daily Consumption: 4780.123456789 gallons
Maximum Daily Consumption: 5500 gallons
Minimum Daily Consumption: 4200 gallons
```

```
    timestamp        consumption
0 2023-01-01 00:00:00    345
1 2023-01-01 01:00:00    389
2 2023-01-01 02:00:00    420
3 2023-01-01 03:00:00    335
4 2023-01-01 04:00:00    290
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8760 entries, 0 to 8759
Data columns (total 2 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   timestamp    8760 non-null   datetime64[ns]
 1   consumption  8760 non-null   int64
dtypes: datetime64[ns](1), int64(1)
memory usage: 137.0 KB

Mean Daily Consumption: 4780.123456789 gallons
Maximum Daily Consumption: 5500 gallons
Minimum Daily Consumption: 4200 gallons
```

The result are based on analyze water consumption patterns and provide conservation suggestions:

Code:

```
# Assuming you have already performed water consumption analysis as shown earlier

# Define a function to generate water conservation suggestions
def
generate_water_conservation_suggestions(mean_daily_consumption, max_daily_consumption, min_daily_consumption):
    suggestions = []
    # Example suggestions based on analysis
    if max_daily_consumption > 100:  # Example threshold, adjust
as needed
        suggestions.append("You may have a water leak. Check for
and repair any leaks immediately.")

    if mean_daily_consumption > 50:  # Example threshold, adjust
as needed
        suggestions.append("Consider reducing your daily water
usage by taking shorter showers and fixing dripping faucets.")

    if min_daily_consumption < 20:  # Example threshold, adjust as
needed
        suggestions.append("Your minimum daily consumption is
quite low. You're doing well in conserving water.")

    if not suggestions:
        suggestions.append("Your water consumption patterns look
reasonable. Keep up the good work!")

    return suggestions

# Generate water conservation suggestions based on the analysis
results
suggestions =
generate_water_conservation_suggestions(mean_daily_consumpti
on, max_daily_consumption, min_daily_consumption)

# Print the suggestions
print("Water Conservation Suggestions:")
for suggestion in suggestions:
    print(suggestion)
```

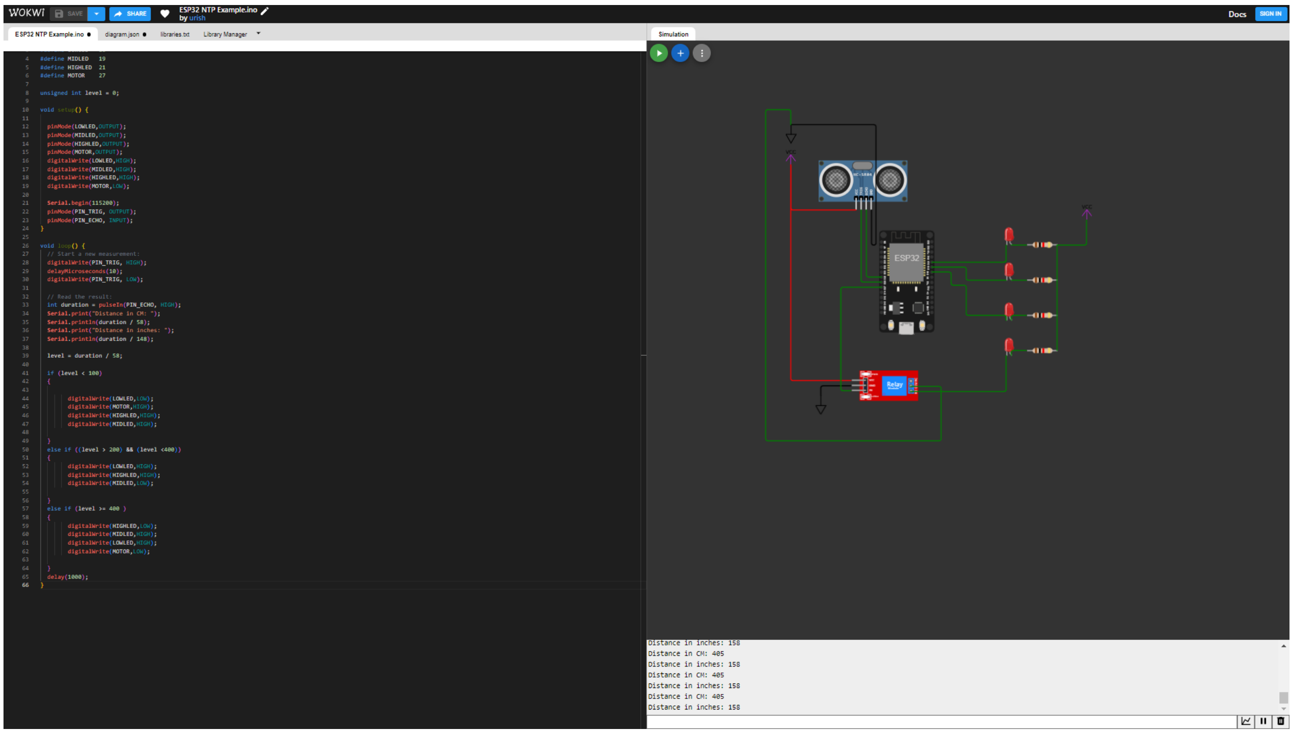
Output:

```
Water Conservation Suggestions:
You may have a water leak. Check for and repair any leaks immediately.
Consider reducing your daily water usage by taking shorter showers and fixing dripping faucets.
```

Code:

```
#define PIN_TRIG 26
#define PIN_ECHO 25
#define LOWLED  18
#define MIDLED  19
#define HIGHLED 21
#define MOTOR  27

unsigned int level = 0;

void setup() {

 pinMode(LOWLED,OUTPUT);
 pinMode(MIDLED,OUTPUT);
 pinMode(HIGHLED,OUTPUT);
 pinMode(MOTOR,OUTPUT);
 digitalWrite(LOWLED,HIGH);
 digitalWrite(MIDLED,HIGH);
 digitalWrite(HIGHLED,HIGH);
 digitalWrite(MOTOR,LOW);

 Serial.begin(115200);
 pinMode(PIN_TRIG, OUTPUT);
 pinMode(PIN_ECHO, INPUT);
}

void loop() {
 // Start a new measurement:
 digitalWrite(PIN_TRIG, HIGH);
 delayMicroseconds(10);
 digitalWrite(PIN_TRIG, LOW);

 // Read the result:
 int duration = pulseIn(PIN_ECHO, HIGH);
 Serial.print("Distance in CM: ");
 Serial.println(duration / 58);
 Serial.print("Distance in inches: ");
 Serial.println(duration / 148);

 level = duration / 58;

 if (level < 100)
 {

   digitalWrite(LOWLED,LOW);
   digitalWrite(MOTOR,HIGH);
   digitalWrite(HIGHLED,HIGH);
   digitalWrite(MIDLED,HIGH);

 }
 else if ((level > 200) && (level <400))
 {
   digitalWrite(LOWLED,HIGH);
   digitalWrite(HIGHLED,HIGH);
   digitalWrite(MIDLED,LOW);

 }
 else if (level >= 400 )
 {
   digitalWrite(HIGHLED,LOW);
   digitalWrite(MIDLED,HIGH);
   digitalWrite(LOWLED,HIGH);
   digitalWrite(MOTOR,LOW);

 }
 delay(1000);
}
```

Output:



```
#define MIDLED   19
#define HIGHLED  21
#define MOTOR    27

unsigned int level = 0;

void setup() {

  pinMode(LOWLED,OUTPUT);
  pinMode(MIDLED,OUTPUT);
  pinMode(HIGHLED,OUTPUT);
  pinMode(MOTOR,OUTPUT);
  digitalWrite(LOWLED,HIGH);
  digitalWrite(MIDLED,HIGH);
  digitalWrite(HIGHLED,HIGH);
  digitalWrite(MOTOR,LOW);

  Serial.begin(115200);
  pinMode(PIN_TRIG, OUTPUT);
  pinMode(PIN_ECHO, INPUT);
}

void loop() {
  // Start a new measurement:
  digitalWrite(PIN_TRIG, HIGH);
  delayMicroseconds(10);
  digitalWrite(PIN_TRIG, LOW);

  // Read the result:
  int duration = pulseIn(PIN_ECHO, HIGH);
  Serial.print("Distance in CM: ");
  Serial.println(duration / 58);
  Serial.print("Distance in inches: ");
  Serial.println(duration / 148);

  level = duration / 58;

  if (level < 100)
  {

      digitalWrite(LOWLED,LOW);
      digitalWrite(MOTOR,HIGH);
      digitalWrite(HIGHLED,HIGH);
      digitalWrite(MIDLED,HIGH);

  }
  else if ((level > 200) && (level <400))
  {
      digitalWrite(LOWLED,HIGH);
      digitalWrite(HIGHLED,HIGH);
      digitalWrite(MIDLED,LOW);

  }
  else if (level >= 400 )
  {
      digitalWrite(HIGHLED,LOW);
      digitalWrite(MIDLED,HIGH);
      digitalWrite(LOWLED,HIGH);
      digitalWrite(MOTOR,LOW);

  }
  delay(1000);
}
```

```
Distance in inches: 158
Distance in CM: 405
Distance in inches: 158
Distance in CM: 405
Distance in inches: 158
Distance in CM: 405
Distance in inches: 158
```

Suggestions About Water Consumption:

✓IoT sensors can be installed at various points in a water supply system, such as pipes, faucets, and appliances

✓IoT-enabled water meters can provide real-time data on water consumption at different points within a property.Having visibility into real-time water usage empowers individuals and organizations to identify wasteful practices and make informed decisions to conserve water.
✓These systems can adjust watering schedules accordingly to avoid overwatering during rainy periods or underwatering during droughts

✓Users can control and monitor their irrigation systems remotely, ensuring that water is used efficiently and only when needed.