Eduvest – Journal of Universal Studies
Volume 5 Number 11, November, 2025
p- ISSN 2775-3735- e-ISSN 2775-3727

Drinking Water Quality Determination Information System Using AI and IOT-Based C4.5 Method

Nurkhaidah*, Aa zezen Zaenal Abidin, Yuli Murdianingsih
Universitas Mandiri, Indonesia
Email: nk.nurkhaidah@gmail.com* zezen@universitasmandiri.ac.id,
yuli@universitasmandiri.ac.id

ABSTRACT

The increasing consumer demand for clean water poses a challenge in maintaining the quality of refillable drinking water. One issue that is often overlooked is the cleanliness of water towers or storage tanks, which can lead to the accumulation of dirt, moss, and bacteria, thereby reducing the quality of water supplied to consumers. However, there is currently no automated system capable of monitoring water quality parameters such as turbidity, Total Dissolved Solids (TDS), and water level in real time at refill depots. This study aims to utilize Internet of Things (IoT) technology and apply the C4.5 algorithm method. This research designs and builds an Internet of Things (IoT)-based information system to monitor drinking water quality in real time using ultrasonic, TDS, and turbidity sensors. The data gathered are taken directly in the field using a microcontroller equipped with ultrasonic, TDS, and turbidity sensors. These data are then processed using data mining techniques and the C4.5 algorithm method. The sensor results yielded 130 data points, of which 91 data points were used as training data and 39 data points as testing data. The implementation of the C4.5 algorithm was carried out with three parameters: water level, which measures the height of water in a reservoir; TDS, which measures the Total Dissolved Solids value; and the turbidity sensor, which measures water quality by detecting its turbidity level. The calculation results showed that the TDS attribute had the highest gain value, making it the root of the decision tree. An accuracy of 97.44% was obtained from the confusion matrix calculation results.

KEYWORDS *C4.5 Algorithm, Information System, Internet of Things (IoT).*

INTRODUCTION

Water is a fundamental necessity for human life and serves as a primary requirement for sustaining health and well-being (Akhai & Taneja, 2025; Mukherjee & Dash, 2024). Every day, people need clean water for daily activities such as drinking, cooking, washing, and bathing (Mapuka, Nel, & Kalumba, 2024). Access to clean water prevents disease, and humans cannot survive more than 4–5 days without drinking water. Water is also the most affected substance by pollution (Archana, Kanakalakshmi, Nithya, Kaarunya, & Renugadevi, 2025; Rusprayunita et al., 2025). Diseases that affect humans can be transmitted and spread through water (Asa et al., 2024). According to the World Health Organization (WHO), approximately 2 billion people globally lack access to safe drinking water services, and contaminated water is responsible for an estimated 485,000 diarrheal deaths annually (WHO, 2022). This alarming statistic underscores the critical importance of ensuring water quality, particularly in developing countries where access to clean water infrastructure remains limited (Irene, Irene, & Daniels, 2025; Tella, Festus, Olaoluwa, & Oladapo, 2025).

Refillable drinking water is currently one of the products available on the market, offering a wide range of advantages and benefits (Baco et al., 2019). It is a type of business that processes raw water into drinking water and then sells it to consumers (Cullmann, Rechlitz,

Sundermann, & Wägner, 2025; Kurniawan, Waskito, & Verawati, 2025). However, the cleanliness of a toren or water storage tank is often a major challenge or neglected without maintenance. The accumulation of dirt, moss, and bacteria on water towers or storage tanks can significantly compromise water quality (Chaurasia & Sharma, 2025). Specifically, inadequate maintenance of storage facilities can lead to microbial contamination, including the growth of Escherichia coli (E. coli) bacteria, formation of biofilm layers, and proliferation of algae and moss (Smith et al., 2021). These contaminants not only deteriorate the physical and chemical properties of water but also pose serious health risks to consumers, including gastrointestinal infections and waterborne diseases (Jones & Anderson, 2020). Despite the recognized importance of water tower hygiene, national surveillance data on contamination levels at refill depots in Indonesia remain limited, highlighting a critical gap in public health monitoring and regulation enforcement. The water needed is clean and healthy water that is deemed suitable for consumption (Machona, Morara Ogendi, & Ahana, 2025; Masum, Ahmad, Arowosaiye, & Aziz, 2025).

According to the Central Statistics Agency (2024) as reported by Ika Wahyu Pradipta, the number of clean water consumers is increasing, reflecting greater public awareness of the need for clean water. Nationally, in 2023 the number of clean water consumers was 17,070,496, an increase of 728,666 consumers or 4.46% compared to 2022 (Pradipta, 2024). According to the Central Statistics Agency of Subang Regency (2021), as cited by Agus Mulia, the number of refillable drinking water sources in 2018 was recorded at six and increased to nine in 2020 (Mulia, 2021). This increase requires refillable drinking water depots to pay more attention to the quality and safety of the drinking water produced (Wijaya et al., 2019). Water quality checks need to be carried out regularly so that the water produced meets the provisions set by the regulations of the Minister of Health (Ishaque & Zia ur Rehman, 2025). However, there are practical obstacles, such as the absence of an automatic system to monitor water quality parameters, such as turbidity, Total Dissolved Solids (TDS), and water level in the storage tank at the Berkah Belendung refill depot in Belendung Village, Cibogo District.

The Berkah Belendung depot currently serves approximately 150–200 customers daily and conducts manual water tower cleaning only once every 3–4 months, which may be insufficient to prevent contamination buildup. This operational context highlights the urgent need for continuous, automated monitoring to ensure consistent water quality and timely maintenance interventions. The water testing process, when carried out manually, requires considerable time and cost. Therefore, a system is needed that can monitor the condition of the reservoir in real time and provide alerts about when cleaning is necessary to maintain water quality.

Good water quality is an essential requirement for human consumption, and with today's technological advancements, it can be monitored using the Internet of Things (IoT) (Fazrie Ramadhan et al., 2023). Through the use of IoT, parameters such as water level, TDS, and turbidity sensors can be monitored in real time to maintain water efficiency and cleanliness. Several previous studies have explored IoT-based water quality monitoring systems, each with distinct strengths and limitations. For example, Alwansyah & Fahrurozi (2024) demonstrated that the application of IoT in water monitoring systems improves water management efficiency through accurate data acquisition. However, their system relied on manual threshold interpretation and did not utilize machine learning algorithms for automated water quality

classification. Similarly, research by Surdin et al. (2024) and Ramadhan et al. (2023) focused on water quality detection based on TDS and pH sensors using NodeMCU microcontrollers, but lacked classification methods. Although these studies successfully integrated sensors, they did not include predictive analytics capabilities, which limits their usefulness for proactive maintenance and decision-making.

In contrast, this study uses a WeMos D1 R32 microcontroller equipped with the C4.5 algorithm for the classification of water quality suitable for consumption, as well as a more secure and robust hardware design. Research by Wijaya et al. (2019) implemented an IoT-based mineral water quality monitoring system using the Node-RED platform and the Simple Additive Weighting (SAW) method, employing pH, TDS, and turbidity sensors to help consumers choose high-quality water depots. The study found that the Mekar Jaya depot performed very well by these parameters. The key innovation in the current research lies in the adoption of the C4.5 algorithm to classify water quality data based on turbidity, TDS, and water level parameters. The developed system is accessible through a local website interface and Wi-Fi connection.

This research addresses critical gaps in the literature by integrating IoT sensor technology with artificial intelligence-based classification. Unlike previous studies that focused on data collection or simple ranking, this work contributes by: (1) implementing the C4.5 decision tree algorithm for automated, rule-based quality classification; (2) utilizing the advanced WeMos D1 R32 microcontroller with improved security; (3) developing a web-based interface for real-time local monitoring; and (4) providing interpretable decision rules for maintenance and operational scheduling at refill depots. These features represent a substantial technical and practical advancement for small-scale water depot operations.

The new research aims to design an AI- and IoT-based drinking water quality determination system using the C4.5 method. This system is expected to reduce contamination risk and improve efficiency by enabling real-time monitoring of water conditions. The research focuses on measuring water level, TDS, and turbidity using Ultrasonic, TDS, and Turbidity sensors integrated with WeMos D1 R32 devices, programmed in C++ and PHP. The location for system deployment is the Berkah Belendung Depot. Through this system, depot owners will be able to monitor and assess water quality accurately and transparently, receiving automatic decisions about the suitability of refillable drinking water.

METHOD

This research employs an applied research design with a quantitative approach, utilizing experimental and developmental methodologies to design, implement, and evaluate an IoT-based water quality monitoring system. The research was conducted at the Berkah Belendung refillable water depot located in Belendung Village, Cibogo District, Subang Regency, West Java, Indonesia, over a period of six months from January to June 2024.

The population of this study consists of all water quality measurement data points collected from the water storage system at the Berkah Belendung depot during the observation period. A total of 130 data points were collected using a purposive sampling technique, representing various time intervals and operational conditions to ensure comprehensive coverage of water quality variations. The sample was divided into 91 data points (70%) for

training the C4.5 algorithm model and 39 data points (30%) for testing model performance, following standard machine learning data partitioning practices.

Data collection was conducted through direct field measurements using IoT sensor instrumentation. Primary data sources included: (1) real-time sensor readings from ultrasonic sensors for water level measurement, TDS sensors for dissolved solids detection, and turbidity sensors for water clarity assessment; (2) timestamp information automatically recorded with each measurement; and (3) classification labels (suitable/not suitable for consumption) determined based on Indonesian Ministry of Health regulations (Permenkes No. 492/2010) regarding drinking water quality standards. Secondary data sources included documentation from the depot regarding maintenance schedules, customer complaints, and previous manual water quality test results.

Data analysis was carried out using multiple techniques: (1) descriptive statistical analysis to characterize the distribution of water quality parameters; (2) C4.5 decision tree algorithm implementation for classification modeling, including entropy calculation, information gain computation, and decision rule generation; (3) confusion matrix analysis to evaluate model performance through accuracy, precision, recall, and F1-score metrics; and (4) comparative analysis with RapidMiner software to validate the decision tree structure and classification results. The C4.5 algorithm was selected for its ability to generate interpretable decision rules, handle both categorical and continuous data, and provide robust classification performance with relatively small datasets.

This research stage includes several steps: literature study, documentation, system analysis and design, system creation, and trial and evaluation. In the literature study stage, various sources such as journals, books, and articles related to AI and IoT were consulted to support the design and implementation of the system. The documentation stage involved collecting related documents from various sources, including websites and toren refill stations. In the analysis and system design stage, a review of literature study results and field data was conducted to prepare initial tool planning and produce a system interface design. The system creation stage involved implementing Internet of Things technology using the C4.5 algorithm. Finally, in the trial and evaluation stage, a series of tests was conducted on the developed tools and systems to assess the feasibility and effectiveness of their use.

RESULT AND DISCUSSION

System Implementation and Testing

1. Laptop Specifications

In the implementation of this final project, the author conducted research using one laptop unit with the following specifications:

Table 1. Laptop Specifications

No.	Name	Laptop Description
1	Laptop	Lenovo
2	*Processor*	*AMD Ryzen 7*
3	*RAM*	*16 GB*
4	*Operation System*	*Windows 11 (64-bit)*

2. Laragon Installation

Laragon is an all in one installation for local web development including apache, PHP, MySQL, phpMyAdmin, Node.jsm, Mongo DB and others. Laragon application software can be downloaded through the official website in https://laragon.org/download.

To install the laragon application just double-click the Laragon Full (64-bit) file: Apache 2.4, Nginx, MySQL 9.1/8.4, PHP 8.4/8.3/8.2/8.1, Node.js 22/23, Python 3.13, Redis, Memcached, PostgreSQL 15/16/17, npm, git. It has been downloaded on the application, then next for each page until the installation page appears and wait for the installation process to complete. The installation process is shown as shown in Figure 1.

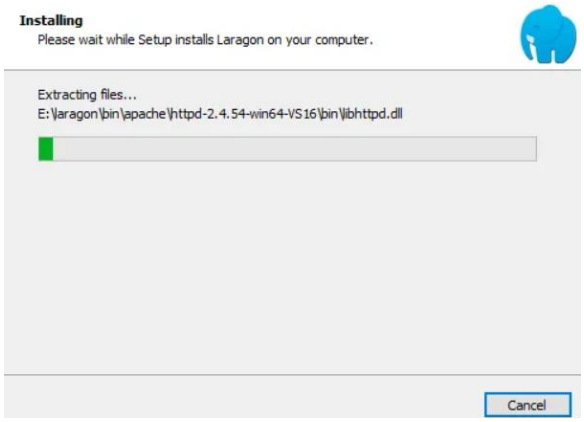


Figure 1. Laragon Installation Process

When the installation is complete it will display on the web on the local PC, as shown in figure 2.

Figure 2. Running Laragon View

3. Arduino IDE Installation

Visit the Arduino IDE https://www.arduino.cc/en/software/ website. to get the latest arduino IDE software download link such as Arduino IDE 2.3.6.

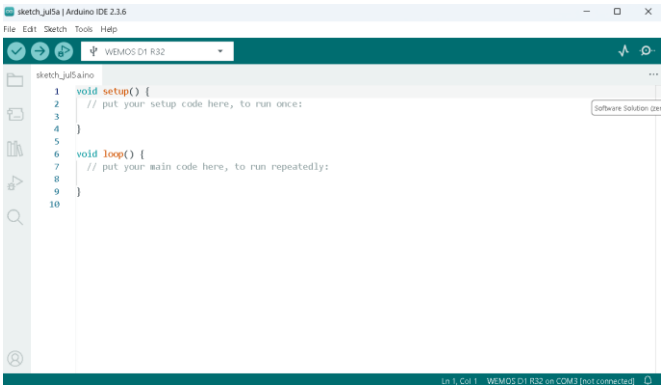


Figure 3. Arduino IDE Display

To install the arduino IDE we just need to click on the previously downloaded file, then an installation page will appear and just select next on each page and wait until the installation process is complete. After the installation is complete, the arduino IDE application is ready to use, as shown in figure 3.

4. Tool Assembly

The author made this tool by using the wemos D1 R32 as a microcontroller, water level to measure the water level in a place, TDS (Total Dissolved Solids) measures the value of the amount of dissolved solids in water, turbidity sensor or turbidity sensor is used to measure water quality by detecting the level of turbidity. The finished tool is shown in figure 4.

Figure 4. Drinking Water Quality Apparatus

5. Database Implementation

An application must have a space for data to be stored. So as to facilitate users in the need for further analysis. The author uses MySQL applications. The following is a database table:

1) Database Data Sensor

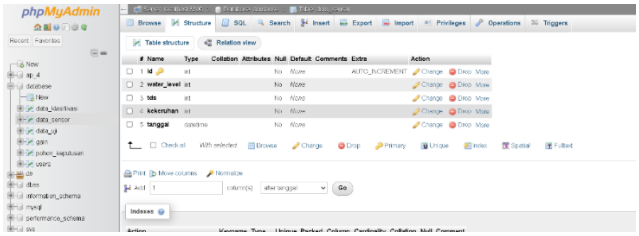
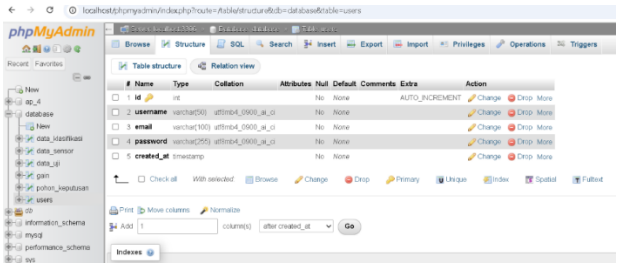


Figure 5. Sensor Data Database

In Figure 5 is a database table of sensor data with the file id as the primary key, water_level, tds, turbidity and date.

2) Database Users



Gambar 6. Database Users

Figure 6 of the database users table contains a field to store user credentials that will be used as login access to the system. The user table consists of id as the primary key, username, email, password, creat_at.

3) Classification Database

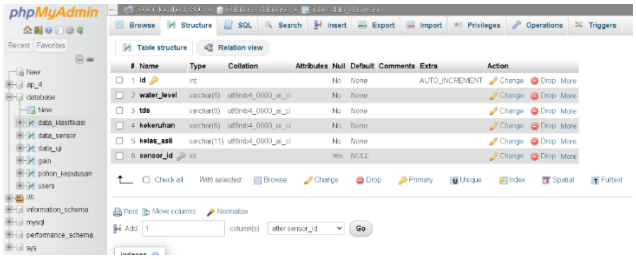


Figure 7. Classification Database

The classification database will store the classification results from sensor data or from parameters that will be processed by class. The attributes in this table are id as the primary key, water_level, tds, turbidity, kelas_asli for the classification of the sensor data, sensor_id as a foreign key to maintain relationships between tables in the database.

4) Test Data Database

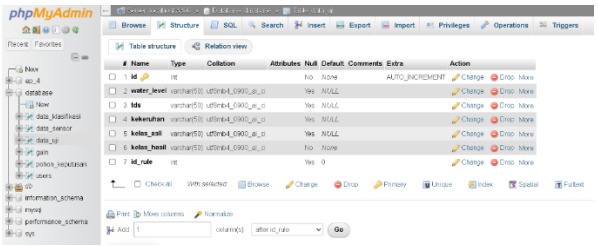


Figure 8. Test Data Database

A test data database is a table to store the tested data and the feasibility data in its class will then be predicted. Data_uji have ID attributes as primary key, water_level, tds, turbidity, kelas_asli, kelas_hasil, id_rule.

5) Database Gain

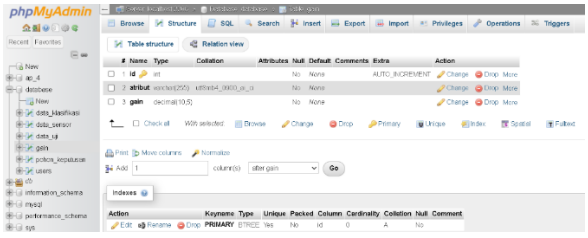


Figure 9. Database Gain

A gain table database to store the calculation of decision tree data where it selects the best attribute as the separator node in the decision tree. In the gain table, it consists of id as the primary key, attribute, gain.

6) Decision Tree Database

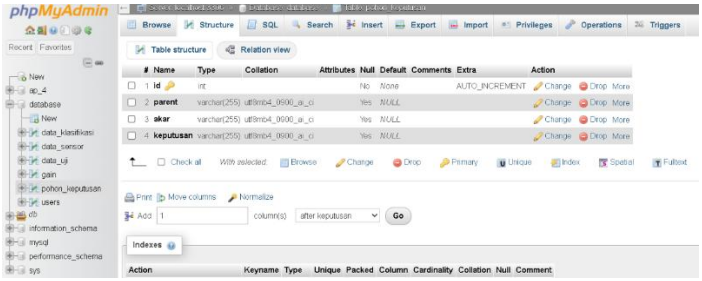
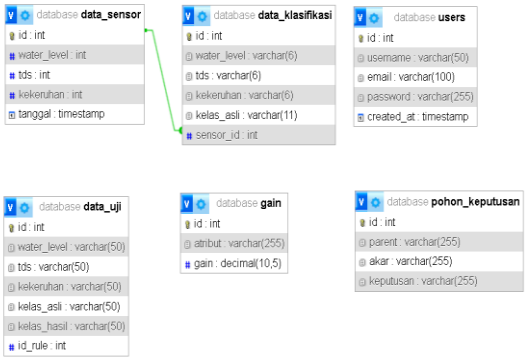


Figure 10. Decision Tree Database

The database of the pohon_keputusan table is as the store of the selected root then it will create a branch and determine the feasibility of the selected attribute. The pohon_keputusan table consists of id as the primary key, parent, root, decision.

7) Designer Database



Gambar 11. Designer Database

A Designer Database is the logical and physical structure of a database, determining what data will be stored, how it will be interconnected and how it will be stored in a database management system.

6. System Interface Implementation

The implementation of the system interface is in accordance with the design of the interface that has been described in the previous chapter.

1) Login Page

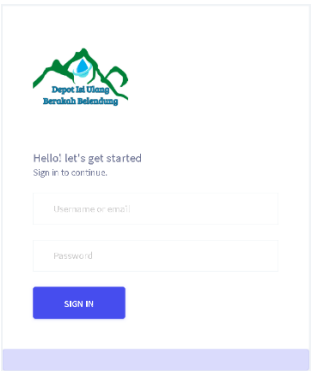


Figure 12. Login

Figure 12 is a login page, the user must log in before entering the main page of the website by entering the username and password that has been registered beforehand.

2) Successfully Login

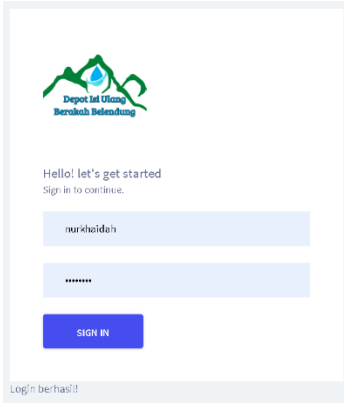


Figure 13. Login Successfully

Figure 13 If you log in successfully, a successful login will appear then it will move to the dashboard.

3) Failed Login

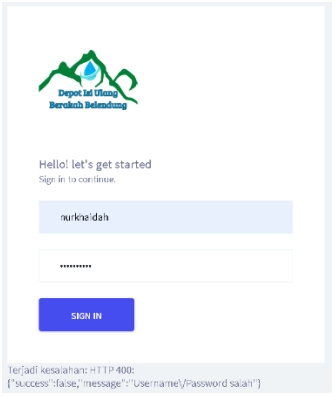


Figure 14. Failed Login

Figure 14 When entering the username and password must be correct according to what has been registered, if the error appears, the information will occur: HTTP 400: {"Success": false,"message":"Username\/Password is wrong". If it does not match the username and password, the login page will still be displayed.

4) Dashboard Page

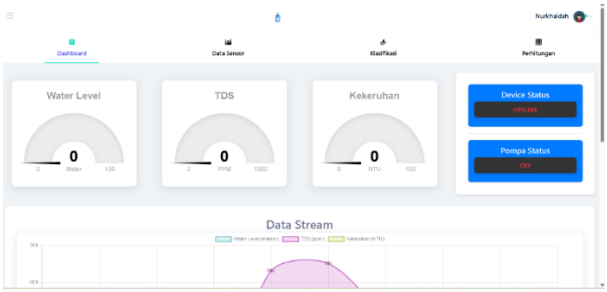


Figure 15 Dashboard

Figure 15 If you successfully log in, you will move to the dashboard or home. On this page, a sensor value of water level, TDS, turbidity, other features and a real-time graph of the hardware is displayed.

5) Data Sensor

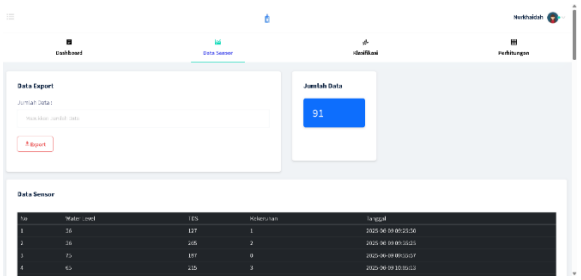


Figure 1. Data *Sensor*

Figure 16 of the *sensor data page* displays the sensor data table from the table, if you need a certain amount of data, just enter the amount of data, then click *the export button*, then *the user* will get *an excel file*.

6) classification

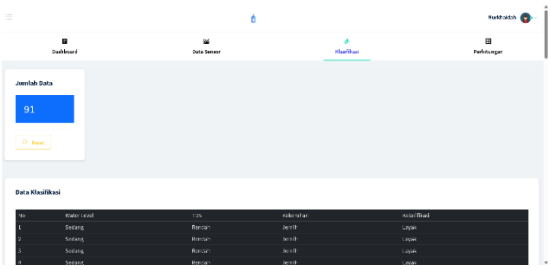


Figure 2. Klasifikasi

In figure 17 the classification page shows data that has been processed or classified. If you click *the reset button,* the data will be lost, but the data will return if the data in the sensor data table is still there and if there is a change or there is an addition of new data. Then the classification table will change automatically and calculate automatically from the data in the sensor data table.

7) Calculations on *c4.5*

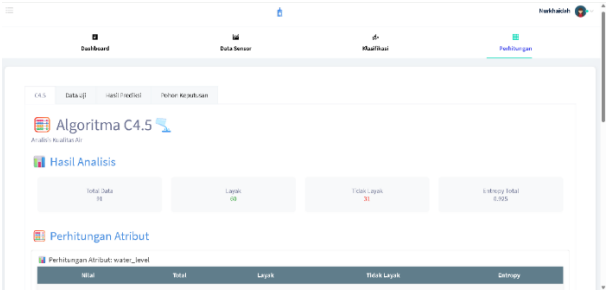


Figure 3. Calculation page on c4.5

In figure 18, the calculation page on the *C4.5 menu* displays the calculation of the results of attribute analysis from *water level, TDS* and turbidity parameters. In addition, it displays the results of the selected attributes among *the water level, TDS* and turbidity parameters.

8) Calculations on test data

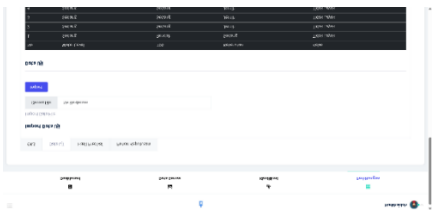


Figure 4. Calculation Page on Test Data

Figure 19 of the calculation page on the test data menu displays the test data, besides it can upload data, click *choose file* and then click *the import button.*

9) Calculation on prediction results

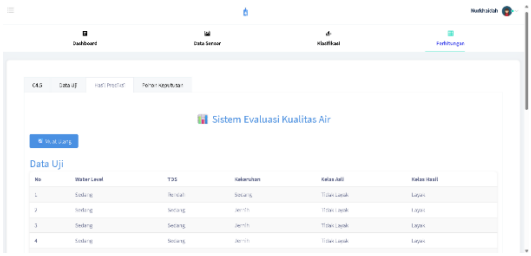


Figure 5 Calculation Page on Prediction Results

Figure 20 shows the test data table and the result class. In addition, there is a calculation of the elements in the confusion matrix for the binary classification problem consisting of four main components.

10) Calculations on the decision tree

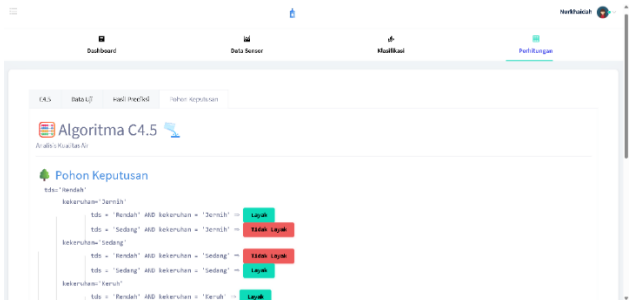


Figure 6 Decision Tree Page

Figure 21 shows the final results of the *C4.5 decision tree algorithm* .

7. *PROGRAM IMPLEMENTATION*

1) Network and *Server Connections*

Figure 7 *Configuration* Network and *Server*

In Figure 22 configure *the Wifi* and server address has the main function of connecting the *board* to *the wifi network*, transmitting data from the *sensor* to *the local* server with *IP addresses, the server* receives the data via *php file (reciveLogModel.PHP* and *reciveDataModel.PHP)* to store or *Logging. ServerDataLog* to send *log data* to the *server* and *serverDataDB* to send the sensor's primary data to the *database.*

2) Pin Declaration and *Sensor Constant*

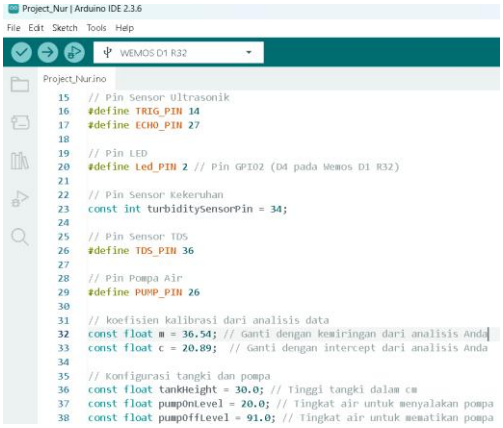


Figure 8. Declaration *Pin* and *Konstanta* Sensor

Figure 23 defines the *pins* used for each *sensor* and actuator, *calibration parameters* and system *configuration*.

3) System Initialization and *Setup*

```
//setup awal (dijalankan sekali saat board menyala)
void setup() {
  Serial.begin(115200); //memulai komunikasi serial

  // Konfigurasi pin LED sebagai output
  pinMode(Led_PIN, OUTPUT);
  digitalWrite(Led_PIN, LOW); // LED mati dulu

//sambungkan ke wifi
  connectToWifi();

  // Konfigurasi pin sensor & pompa
  pinMode(TRIG_PIN, OUTPUT);
  pinMode(ECHO_PIN, INPUT);
  pinMode(turbiditySensorPin, INPUT);
  pinMode(TDS_PIN, INPUT);
  pinMode(PUMP_PIN, OUTPUT);
  digitalWrite(PUMP_PIN, HIGH); // Matikan pompa pada awal

//membuat task RTOS untuk multitasking
  xTaskCreate(
    measureSensorsTask, // Fungsi task
    "Ukur Sensor", // Nama task
    3072, // Ukuran stack dalam byte
    NULL, // Parameter task
    1, // Prioritas task
    NULL // Handle task
  );
```

Figure 9 Initialization and *Setup* System

In Figure 24 the *setup function* performs serial communication initialization, *pin configuration*, *wifi connection* and creates three *FreeRTOS tasks* for *multitasking*.

4) Water *Level* Measurement Function

```
//fungsi untuk mengukur level air dengan sensor ultrasonik
int measureWaterLevel() {
    long duration;
  float distance;
  // Mengirim sinyal trigger ultrasonik
  digitalWrite(TRIG_PIN, LOW);
  delayMicroseconds(2);
  digitalWrite(TRIG_PIN, HIGH);
  delayMicroseconds(10);
  digitalWrite(TRIG_PIN, LOW);
  // Mengukur durasi sinyal kembali
  duration = pulseIn(ECHO_PIN, HIGH);
  // Menghitung jarak dalam cm
  distance = (duration * 0.034) / 2;

  // Menghitung tingkat air dalam cm
  float waterLevelCm = tankHeight - distance;

  // Clamping nilai agar tidak melebihi batas tangki
  if (waterLevelCm < 0) waterLevelCm = 0;
  if (waterLevelCm > tankHeight) waterLevelCm = tankHeight;

  // Konversi ke persentase (0-100%) dengan pembulatan
  int waterLevelPercent = static_cast<int>((waterLevelCm / tankHeight) * 100 + 0.5);
```

Figure 10 Measurement Function *Level* Water

This function measures the water level using *an ultrasonic sensor* and converts it to *a percentage.*

5) Pump Control Function

```
//fungsi untuk kontrol pompa berdasarkan level air
String measurePompa(int waterLevelPercent) { // DIUBAH: Parameter sekarang persentase
  // Kontrol pompa air berdasarkan tingkat air (dalam %)
  if (waterLevelPercent <= pumpOnLevel) {
    digitalWrite(PUMP_PIN, LOW); // Nyalakan pompa
    Serial.println("Pompa Air: ON");
    return "ON";
  } else if (waterLevelPercent >= pumpOffLevel) {
    digitalWrite(PUMP_PIN, HIGH); // Matikan pompa
    Serial.println("Pompa Air: OFF");
    return "OFF";
  }

  return digitalRead(PUMP_PIN) == LOW ? "ON" : "OFF";
}
```

Figure 11 Pump Control Function

This function controls the pump based on *the measured water* level.

6) TDS Measurement Function

```
//fungsi untuk membaca sensor TDS
int measureTDS() {
  int sensorValue = analogRead(TDS_PIN);
  float voltage = sensorValue * (3.3 / 4095.0); // Konversi ke volt

  // Hitung TDS (ppm) pakai rumus persamaan kalibrasi
  int tds = m * voltage + c;

  Serial.print("TDS: ");
  Serial.print(tds);
  Serial.println(" ppm");

  return tds;
}
```

Figure 12 Measurement Function *TDS*

This function reads the *TDS value* from the *sensor* and converts it using calibration equations.

7) Reading *Sensors* and Transmitting Data

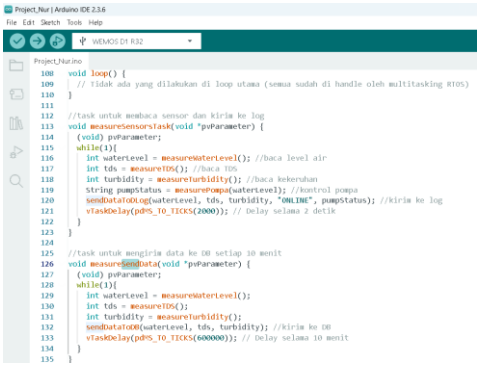


Figure 13 *Task* Read *Sensor* and Send Data

In this image, *measureSensorTask* during this function will read *the sensor data* after the read data is sent to the *log server* using *sendDataToLog*. Data is sent once every 2 seconds. Meanwhile, *measureSendData* reads the sensor and sends it to the *database* every 10 minutes using *sendDataToDB*.

8) Data Transmission to *Server Function*

Figure 14 Data Transmission to Server Function

This function sends *sensor data* to the *server* using *the HTTP protocol.* On sending *sensor data* to *the log server*, the check is connected to *wifi*. *HTTP client* to access *the DataLog server.* Then the *sensor data* will be sent to the *server* via *POST.* The sendDataToLog *function* to send *sensor data* and pump status to the *log server* in the previous task every 2 seconds. The data is sent via *HTTP POST* to the *PHP server address.*

9) Sensor Data Delivery Function to *Database*

```
//fungsi kirim data ke database utama
void sendDataToDB(int waterLevel, int tds, int turbidity) {
 if (WiFi.status() == WL_CONNECTED) {
  HTTPClient http;
  http.begin(serverDataDB); // Ganti dengan serverDataDB
  http.addHeader("Content-Type", "application/x-www-form-urlencoded");

  String postData = "water_level=" + String(waterLevel)  + "&tds=" + String(tds) + "&kekeruhan=" + String(turbidity);

  int httpResponseCode = http.POST(postData);

  if (httpResponseCode > 0) {
   String response = http.getString();
   Serial.println(httpResponseCode);
   Serial.println(response);
  } else {
   Serial.print("Error on sending POST: ");
   Serial.println(httpResponseCode);
  }
  http.end();
 } else {
  Serial.println("Error in WiFi connection");
 }
}
```

Figure 15 Data Transmission Function *Sensor* to *Database*

This function sends *sensor data* to the *server* using *the HTTP protocol.* The *sendDataToDB* function to send *sensor data* to the *main database server* via *HTTP POST.*

10) Function Classifies Each *Parameter*

```
// Fungsi klasifikasi untuk setiap parameter
function klasifikasi_water_level($level) {
    if ($level > 90) return "Full";     // Level lebih dari 90
    if ($level > 20) return "Sedang";   // Level antara 21-60
    return "Rendah";                    // Level 20 atau kurang
}

// Mengklasifikasikan nilai Total Dissolved Solids (TDS) menjadi kategori
function klasifikasi_tds($tds) {
    if ($tds <= 300) return "Rendah"; // TDS 300 atau kurang
    if ($tds <= 960) return "Sedang";   // TDS 301-959
    return "Tinggi";                    // TDS > 960
}

function klasifikasi_kekeruhan($kekeruhan) {
    if ($kekeruhan <= 3) return "Jernih";    // Kekeruhan <= 3
    if ($kekeruhan <= 5) return "Sedang";    // Kekeruhan 4-5
    return "Keruh";                          // Kekeruhan > 6
}
```

Figure 16 Classification Function *Parameter*

This function classifies the *parameters* to be classified in the *calculation of the C4.5 algorithm.*

11) Overall *Classification Function of the Sensor*

```
function klasifikasi_keseluruhan($water_level, $tds, $kekeruhan) {
    // Logika klasifikasi keseluruhan tetap sama
    if ($water_level == "Full" && $tds == "Rendah" && $kekeruhan == "Jernih") return "Layak";
    if ($water_level == "Full" && $tds == "Rendah" && $kekeruhan == "Sedang") return "Layak";
    if ($water_level == "Full" && $tds == "Sedang" && $kekeruhan == "Jernih") return "Layak";
    if ($water_level == "Full" && $tds == "Sedang" && $kekeruhan == "Sedang") return "Layak";
    if ($water_level == "Sedang" && $tds == "Sedang" && $kekeruhan == "Sedang") return "Layak";
    if ($water_level == "Sedang" && $tds == "Rendah" && $kekeruhan == "Jernih") return "Layak";
    if ($water_level == "Rendah" && $tds == "Rendah" && $kekeruhan == "Jernih") return "Layak";
    if ($water_level == "Rendah" && $tds == "Sedang" && $kekeruhan == "Sedang") return "Layak";
    return "Tidak Layak";
}
```

Figure 17 All Classification Function *Sensor*

This function is a classification of all *parameter data* with "Eligible" or "Not Eligible".

12) Entropy *Calculation Function*

```php
// Fungsi calculate_entropy()
// Menghitung nilai entropy dari data (untuk perhitungan gain)
// Rumus: -Σ (p * log2(p))
Windsurf: Refactor | Explain | ×
function calculate_entropy($layak, $tidak) {
    $total = $layak + $tidak;
    if($total == 0) return 0; // Jika data kosong, entropy = 0

    // Probabilitas masing-masing kelas
    $p_layak = $layak / $total;
    $p_tidak = $tidak / $total;

    $entropy = 0;
    if($p_layak > 0) $entropy -= $p_layak * log($p_layak, 2);
    if($p_tidak > 0) $entropy -= $p_tidak * log($p_tidak, 2);

    return round($entropy, 3); // Dibulatkan 3 angka desimal
}
```

Figure 18 Counting Function *Entropy*

This function is to calculate *the entropy* of feasible and unfeasible.

13) Function Displaying Attributes of *Parameters*

```php
// Fungsi get_attribute_values()
// Mengambil nilai unik (distinct) dari sebuah atribut di tabel
// Contoh: semua nilai unik dari kolom "water_level"
Windsurf: Refactor | Explain | ×
function get_attribute_values($pdo, $attribute) {
    $sql = "SELECT DISTINCT $attribute FROM data_klasifikasi";
    $stmt = $pdo->prepare($sql);
    $stmt->execute();
    return $stmt->fetchAll(PDO::FETCH_COLUMN); // Hasil berupa array
}
```

Figure 19 Function Displays Attributes of *Parameter*

This function is to display attributes of multiple *parameters.*

14) Data Quantity Function

```php
// Fungsi jumlah_data()
// Menghitung jumlah data sesuai kondisi tertentu
// Contoh: jumlah data dengan kelas_asli = 'Layak'
Windsurf: Refactor | Explain | ×
function jumlah_data($pdo, $condition, $params = []) {
    $sql = "SELECT COUNT(*) FROM data_klasifikasi";

    if(!empty($condition)) {
        $sql .= " WHERE $condition";
    }

    $stmt = $pdo->prepare($sql);
    $stmt->execute($params);
    return (int)$stmt->fetchColumn();
}
```

Figure 20 Data Quantity Function

This function is to calculate the amount of case data stored on the data_klasifikasi.

15) Gain Calculation Function

```php
// Fungsi hitung_gain()
// Menghitung nilai Gain dari suatu atribut
// Gain = Entropy total - Σ(prob * entropy tiap cabang)
Windsurf: Refactor | Explain | ×
function hitung_gain($pdo, $attribute, $entropy_total, $total_data) {
    $values = get_attribute_values($pdo, $attribute); // Ambil semua nilai unik atribut
    $sum_entropy = 0;

    foreach($values as $value) {
        // Hitung jumlah data Layak & Tidak Layak untuk nilai atribut ini
        $layak = jumlah_data($pdo, "$attribute = :value AND kelas_asli='Layak'", [':value' => $value]);
        $tidak = jumlah_data($pdo, "$attribute = :value AND kelas_asli='Tidak Layak'", [':value' => $value]);
        $total = $layak + $tidak;

        if($total == 0) continue;

        // Hitung entropy cabang ini
        $entropy = calculate_entropy($layak, $tidak);
        // Hitung probabilitas cabang
        $prob = $total / $total_data;
        // Jumlahkan entropy cabang * probabilitasnya
        $sum_entropy += ($prob * $entropy);
    }

    // Rumus gain
    return $entropy_total - $sum_entropy;
}
```

Figure 21 Counting Function *Gain*

In this part of the function code, to calculate *the gain* with multiple functions such as jumlah_data, *calculate_entropy, get_attribute_value.*

16) Highest Gain Yield Function

```php
// Fungsi save_best_attribute()
// Menyimpan atribut terbaik (dengan Gain tertinggi) ke tabel gain
Windsurf: Refactor | Explain | ×
function save_best_attribute($pdo, $gains) {
    $max_gain = max($gains); // Cari gain terbesar
    $best_attribute = array_search($max_gain, $gains); // Ambil nama atribut

    // Simpan ke tabel gain
    $sql = "INSERT INTO gain (atribut, gain) VALUES (:atribut, :gain)";
    $stmt = $pdo->prepare($sql);
    $stmt->execute([
        ':atribut' => $best_attribute,
        ':gain' => round($max_gain, 3)
    ]);

    return [
        'atribut' => $best_attribute,
        'gain' => round($max_gain, 3)
    ];
}
```

Figure 22 Function Results *Gain* Highest

This function is to store the result data of the *best gain* and attributes when *the gain calculation* is complete.

17) Function of Creating New Conditions

```php
// Fungsi build_new_condition()
// Membuat kondisi baru untuk filtering data saat split
// Contoh: sebelumnya "1=1", setelah split jadi "tds='Rendah'"
Windsurf: Refactor | Explain | ×
function build_new_condition($pdo, $current_condition, $attribute, $value) {
    $value = $pdo->quote($value);
    $new_condition = "$attribute = $value"; // Amankan nilai agar SQL Injection aman
    return ($current_condition === '1=1') ? $new_condition : "$current_condition AND $new_condition";
}
```

Figure 23 Function of Creating New Conditions

This function is to create a new condition of the *calculation of the C4.5 algorithm.*

18) Best Attribute Function

```php
// Fungsi get_best_attribute_from_remaining()
// Memilih atribut terbaik dari atribut yang tersisa
Windsurf: Refactor | Explain | ×
function get_best_attribute_from_remaining($remaining_attributes, $gains) {
    $filtered_gains = array_intersect_key($gains, array_flip($remaining_attributes));

    if(empty($filtered_gains)) {
        return false;
    }

    $max_gain = max($filtered_gains);

    if($max_gain <= 0) {
        return false;
    }

    return array_search($max_gain, $filtered_gains);
}
```

Figure 24 Best Attribute Function

This function is to get the best *attribute* from the calculation of *the C4.5 algorithm*.

19) Decision *Tree* Node Functions

```php
// Fungsi save_node()
// Menyimpan node pohon ke tabel pohon_keputusan
// - parent: induk node
// - node: nama atribut/nilai
// - decision: keputusan (Layak/Tidak Layak) atau null
Windsurf: Refactor | Explain | ×
function save_node($pdo, $parent, $node, $decision) {
    $sql = "INSERT INTO pohon_keputusan (parent, akar, keputusan) VALUES (:parent, :node, :decision)";
    $stmt = $pdo->prepare($sql);
    $stmt->execute([
        ':parent' => $parent,
        ':node' => $node,
        ':decision' => $decision
    ]);
}
```

Figure 25 Function *Node* Decision Tree

This function is to store *the node data* when the calculation is complete.

20) Function of Creating *a Decision Tree*

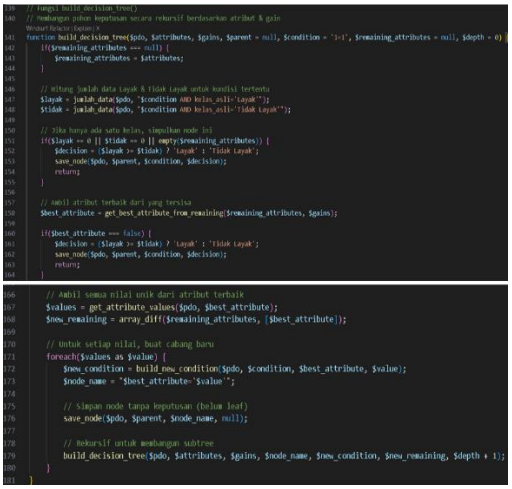


Figure 26 Function of Creating a Decision Tree

This function is to create *a Decision Tree* .

21) C4.5 Algorithm *Calculation Function*

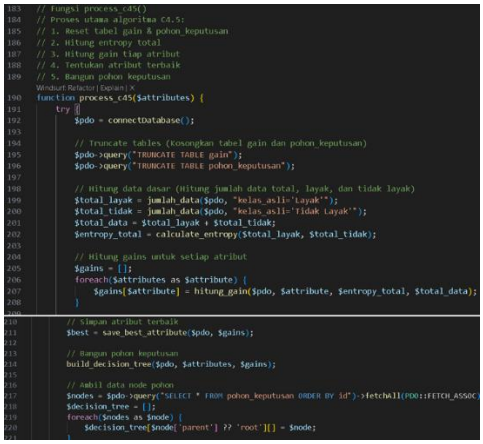


Figure 27 Calculation Function *Algorithm C4.5*

This function is to perform the calculation process *of the C4.5 algorithm* with some of the functions above. At this stage to process the calculation.

22) Function of Creating *a Decision Tree*

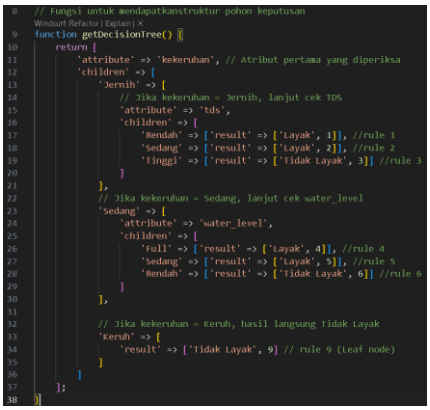


Figure 28 Create Function *Tree Decision*

This function is to search or create *a decesion tree* .

23) Prediction Function of Decision Tree

```php
// Prediksi kelas berdasarkan struktur pohon keputusan
Windsurf: Refactor | Explain | ×
function predictClass($water_level, $tds, $kekeruhan) {
    $tree = getDecisionTree(); // Ambil pohon keputusan
    $currentNode = $tree; // Mulai dari akar pohon

    while (true) {
        // Jika node sudah merupakan hasil akhir (leaf node), kembalikan hasilnya
        if (isset($currentNode['result'])) return $currentNode['result'];
        // Ambil atribut yang sedang diperiksa di node ini
        $attribute = $currentNode['attribute'];

        // Tentukan nilai input yang sesuai dengan atribut
        switch ($attribute) {
            case 'water_level': $value = $water_level; break;
            case 'tds': $value = $tds; break;
            case 'kekeruhan': $value = $kekeruhan; break;
            default: $value = null;
        }

        // Jika nilai tidak ditemukan dalam cabang pohon, kembalikan default "Tidak Layak"
        if (!isset($currentNode['children'][$value])) return ['Tidak Layak', 0];
        // Pindah ke node anak sesuai nilai input
        $currentNode = $currentNode['children'][$value];
    }
}
```

Figure 29 Prediction Function of Decision Tree

This function is to perform predictions against the getDecesionTree.

24) Fungsi *Confunsion Matrix*

```php
// Hitung Confusion Matrix
// Membuat confusion matrix untuk evaluasi hasil klasifikasi
// - TP (True Positive): Layak diprediksi Layak
// - TN (True Negative): Tidak Layak diprediksi Tidak Layak
// - FP (False Positive): Tidak Layak diprediksi Layak
// - FN (False Negative): Layak diprediksi Tidak Layak
Windsurf: Refactor | Explain | ×
function getConfusionMatrix($pdo) {
    // Inisialisasi nilai confusion matrix
    $conf = ['TP' => 0, 'TN' => 0, 'FP' => 0, 'FN' => 0];

    // Ambil data asli dan hasil prediksi
    $stmt = $pdo->query("SELECT kelas_asli, kelas_hasil FROM data_uji");

    // Hitung jumlah TP, TN, FP, FN
    while ($row = $stmt->fetch(PDO::FETCH_ASSOC)) {
        if ($row['kelas_asli'] == 'Layak' && $row['kelas_hasil'] == 'Layak') $conf['TP']++;
        elseif ($row['kelas_asli'] == 'Tidak Layak' && $row['kelas_hasil'] == 'Tidak Layak') $conf['TN']++;
        elseif ($row['kelas_asli'] == 'Tidak Layak' && $row['kelas_hasil'] == 'Layak') $conf['FP']++;
        elseif ($row['kelas_asli'] == 'Layak' && $row['kelas_hasil'] == 'Tidak Layak') $conf['FN']++;
    }
    return $conf;
}
```

Figure 30 Function *Confunsion Matrix*

This function is to find *the confunsion matrix.*

25) Test Data Accuracy and Error Function

```php
// Menghitung metrik evaluasi dari confusion matrix:
// Hitung Akurasi & Kesalahan
Windsurf: Refactor | Explain | ×
function getMetrics($pdo) {
    $conf = getConfusionMatrix($pdo); // Ambil confusion matrix
    $total = array_sum($conf); // Total data uji

    // Hitung akurasi dan error dalam persentase (%)
    $accuracy = $total ? round(($conf['TP'] + $conf['TN']) / $total * 100, 2) : 0;
    $error = $total ? round(($conf['FP'] + $conf['FN']) / $total * 100, 2) : 0;

    // Mengembalikan hasil evaluasi dalam array
    return compact('conf', 'accuracy', 'error');
}
```

Figure 31 Test Data Accuracy and Error Function

This function is to look for accuracy and errors in the prediction calculation from the test data.

8. *IMPLEMENTATION* OF *C4.5* CALCULATION IN *RAPID MINER*

1) Rapid Miner *Decision Tree*

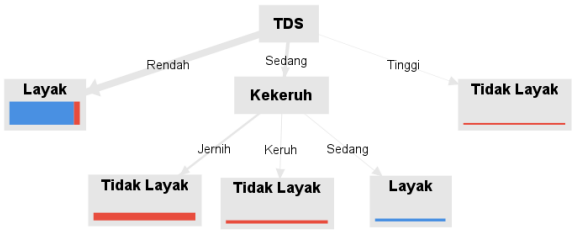


Figure 32 Decision Tree on *Rapid Miner*

In figure 47 The *TDS* attribute is used as the main attribute because it has the most to the final decision. The resulting decision tree structure shows the relationship between *TDS* and turbidity and water classification results. The blue color on the diagram depicts the Eligible condition, while the red color indicates Not Eligible. The structure of this tree shows that water quality is not only influenced by one factor, but is a combination of *TDS* content and the level of turbidity of water, but most of it shows that blue with decent conditions is greater or more influential than red. Thus, the low *variable TDS* attribute results in viable water classification conditions. *The C4.5 algorithm* model is capable of generating accurate and logical classification decisions based on *real-time* IoT sensor *data*.

2) *Rule* Tree Rapid Miner *Decision*

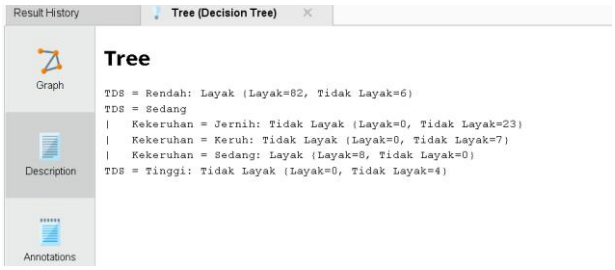


Figure 33 *Rule* Decision Tree on *Rapid Miner*

Figure 48 shows the rule results of the *C4.5 algorithm of* the decision tree on *Rapid Miner.*

3) Confusion Matrix Rapid Miner

accuracy: 97.44%

	true Tidak Layak	true Layak	class precision
pred. Tidak Layak	8	0	100.00%
pred. Layak	1	30	96.77%
class recall	88.89%	100.00%	

Figure 34 *Confusion Matrix* at *Rapid Miner*

Figure 4.49 shows the *Confusion Matrix results* from *the C4.5 algorithm* of the decision tree in *Rapid Miner.*

9. *TOOL TESTING* IMPLEMENTATION

The implementation of tool testers and system testing is a table of tool and system tests that have been tested by the author of the feasibility level and accuracy level. In tables 1 and 2.

10. TOOL TEST RESULTS TABLE

The tool test results table is a system *record* to test whether the tool runs properly without *errors* or not.

Table 1 Tool Testing

No.	Process Name	Test Results
1.	Wemos D1 R32 *connection* to *Wifi* internet	Ya
2.	Wemos D1 R32 *Connection* to *PC (Personal Computer)*	Ya
3.	Water level *data capture*	Ya
4.	TDS data retrieval	Ya
5.	Turbidity data capture	Ya
6.	*Monitoring* data from *sensors*	Ya

7.	Connection to *the database*	Ya

11. **SYSTEM TEST RESULTS TABLE**

The system test results table as a system *record* to test whether the system created can run properly without any *errors.*

Table 2 System Testing

No.	Name of Halam and Process	Test Results
1.	Home	Ya
2.	Halam Data *sensor*	Ya
3.	Export page "Retrieving Sensor Data "	Ya
4.	Classification Page	Ya
5.	Calculation Page	Yes
6.	Calculation Page *C4.5*	Yes
7.	Import data *testing page*	Yes
8.	Prediction results page	Yes
9.	Decision Tree Page	Yes

CONCLUSION

An information system for determining drinking water quality can be implemented using an AI- and IoT-based C4.5 method, as described below. The tool or system developed in this study can be utilized by refill depot owners in an Internet of Things (IoT)-based framework to monitor water quality in real time. With this tool, depot owners can observe the quality of refillable water, as the system classifies drinking water quality based on the sensor data received. The classification results provide information on whether water is suitable or unsuitable for consumption. Implementing the C4.5 algorithm method on the drinking water quality classification system at the refill depot yielded an accuracy rate of 97.44% based on confusion matrix calculations. This high level of accuracy demonstrates the effectiveness of AI-driven water quality monitoring and decision-making in practical applications.

REFERENCES

Akhai, Shalom, & Taneja, Tanu. (2025). The Critical Role Of Water Quality: Health Impacts, Contaminants, And Sustainable Solutions For Environmental And Human Well-Being. In *Smart Water Technology For Sustainable Management In Modern Cities* (Pp. 101–116). Igi Global Scientific Publishing.

Alwansyah, & Fahrurozi, A. (2024). Implementation Of The Internet Of Thing (Iot) Shrimp Farming Vannamei Water Quality Monitoring System On An Android-Based Application. Scientific Journal Of Technology And Engineering, 29(1), 71–85. Https://Doi.Org/10.35760/Tr.2024.V29i1.11227

Ana, Nurmalasari, & Rohayani, Y. (2020). Application Of The Waterfall Method In The Design Of Freight Forwarding Accounting Information System. In Journal Of Accounting Information Systems (Vol. 01, Issue 01). Https://Ejournal.Bsi.Ac.Id/Ejurnal/Index.Php/Justian

Archana, C. M., Kanakalakshmi, A., Nithya, K., Kaarunya, E., & Renugadevi, K. (2025). Health Effects Of Emerging Contaminants: Effects On Humans Via Ingestion Of

Contaminated Water. In *Emerging Contaminants In Water* (Pp. 269–306). Springer.

Asa, F., Kolastriwan Romanda, E., Titing, J. N., Claris, M., Nurak, S., Zuhri, N., Geno, P., Kaesmetan, Y. R., Stikom, T. I., & Kupang, U. (2024). Decision Support System To Determine The Quality Of The Refillable Mineral Water Depot Using The Topsis Method (Technique For Order Preference By Similarity To Ideal Solution). Methodika Journal, 10.

Baco, S., Musrawati, Anugrah, A., & Iskandar. (2019). Design And Construction Of Microcontroller-Based Drinking Water Monitoring System. 14, 2.

Baeldung On Computer Science. (2024, March 18). Uml State Diagrams Explained . Https://Www.Baeldung.Com/Cs/Uml-State-Diagrams

Chaurasia, Shardesh Kumar, & Sharma, Navdeep. (2025). Current Trends And Future Directions Of Hydroponics In Urban Agriculture: A Competent Technology For Food Production And Wastewater Management. *Journal Of Scientific Agriculture*, *9*, 177–188.

Cullmann, Astrid, Rechlitz, Julia, Sundermann, Greta, & Wägner, Nicole. (2025). External Costs Of Water Pollution In The Drinking Water Supply Sector. *American Journal Of Agricultural Economics*, *107*(2), 504–531.

Fazrie Ramadhan, Z., Sugiarto, B., Haviani Laluma, R., & Gunawansyah. (2023). Designing An Internet Of Things (Iot) Drinking Water Consumption Feasibility Monitoring System.

Irene, Julius, Irene, Bridget Nneka, & Daniels, Chux. (2025). Not A Drop To Drink: Addressing Nigeria's Deepening Freshwater Crisis. *Water*, *17*(12), 1731.

Ishaque, Waseem, & Zia Ur Rehman, Muhammad. (2025). Impact Of Climate Change On Water Quality And Sustainability In Baluchistan: Pakistan's Challenges In Meeting United Nations Sustainable Development Goal (Unsdg) Number 6. *Sustainability*, *17*(6), 2553.

Kurniawan, Dea, Waskito, Budhi, & Verawati, Noning. (2025). Business Communication Strategy In Improving Customer Confidence Regarding The Quality Of Drinking Water Companies For Regional Drinking Water. *Interaction Communication Studies Journal*, *2*(1), 10.

Machona, Sophie, Morara Ogendi, George, & Ahana, Bayongwa Samuel. (2025). Assessment Of Groundwater Quality For Drinking Using The Water Quality Index. *Water Quality Research Journal*, *60*(1), 151–163.

Mapuka, Fortunate N., Nel, Werner, & Kalumba, Ahmed M. (2024). Exploring Household Water Conservation Methods In Rural South Africa: A Case Of The Mbhashe And Mnquma Local Municipalities. *Sustainable Water Resources Management*, *10*(4), 145.

Masum, Ahmad, Ahmad, Nehaluddin, Arowosaiye, Yusuf Ibrahim, & Aziz, Hajah Hanan Hj Awang Abdul. (2025). The Right To A Clean, Healthy And Sustainable Environment In Brunei Darussalam. *European Journal Of Arts, Humanities And Social Sciences*, *2*(3), 86–97.

Mukherjee, Abhijit, & Dash, Adya A. (2024). Water Matters: The Thirst, The Demand, And The Society. In *Water Matters* (Pp. 3–11). Elsevier.

Rusprayunita, Nurendah Ratri Azhar, Puspitasari, Sepsiana, Mahiroh, Hodimatum, Setyani, Enrika Rahayu, Citraningtyas, Veriana Indah, Wahyuningsih, Windu Syawalina, & Rauf, Annisa Utami. (2025). Water Pollution Reduction For Sustainable Urban Development. In *Sustainable Urban Environment And Waste Management: Theory And Practice* (Pp.

1–21). Springer.

Tella, Taleat Adewale, Festus, Ben, Olaoluwa, Temitope Daud, & Oladapo, Abiodun Sinmiat. (2025). Water And Wastewater Treatment In Developed And Developing Countries: Present Experience And Future Plans. In *Smart Nanomaterials For Environmental Applications* (Pp. 351–385). Elsevier.