**AI-Driven Smart Farming Solution**

In modern agriculture, precision farming is essential for optimizing water usage and improving crop yield. This project implements an Edge AI-based Soil Moisture and Environmental Monitoring System using the ESP32S3, AHT11 (temperature & humidity sensor), capacitive soil moisture sensor, and OLED display. The system predicts watering decisions and next-day environmental conditions using machine learning models implemented in an embedded environment.

**Objectives**

* Measure soil moisture, temperature, and humidity in real-time.
* Use Random Forest classification to decide whether to water the soil.
* Use Regression models to predict next-day temperature and humidity.
* Display data on an OLED screen for easy monitoring.
* Utilize NTP-based real-time clock to track and log readings accurately.
* Train models in Python, extract coefficients, and implement them in ESP32.

**Hardware Components**

* XIAO ESP32S3 (Microcontroller)
* AHT11 (Temperature & Humidity Sensor)
* Capacitive Soil Moisture Sensor
* 0.96" OLED Display (I2C interface)
* Push Buttons (Menu navigation)
* WiFi Connectivity (For NTP synchronization)

**Software and Libraries Used**

* Arduino IDE (For programming ESP32S3)
* Adafruit SSD1306 (OLED display driver)
* Adafruit AHTX0 (Sensor driver)
* WiFi and NTPClient libraries (For real-time clock synchronization)
* Scikit-learn (For model training in Python)

**Machine Learning Implementation**

**Random Forest for Watering Decision**

**Why Random Forest?**

* Handles non-linearity: Soil moisture, temperature, and humidity are interdependent.
* Better accuracy: Compared to Naive Bayes, which assumes feature independence.
* Robust to noise: It averages multiple decision trees to reduce errors.
* Low computational cost: Can be converted into simple decision rules for ESP32.

**Comparison of CNN, Decision Tree, and Random Forest:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Algorithm** | **Accuracy** | **Complexity** | **Computation Time** | **Suitability** |
| **CNN** | High | High | Slow | Not suitable for ESP32 |
| **Decision Tree** | Moderate | Low | Fast | May overfit |
| **Random Forest** | High | Moderate | Moderate | Best balance for ESP32 |

**Training Process:**

1. Dataset: Historical data containing Soil Moisture, Temperature, Humidity, and Watering Decision (0 = Do Not Water, 1 = Water).
2. Model Selection: Trained RandomForestClassifier(n\_estimators=10, max\_depth=5).
3. Rule Extraction: Converted the trained model's rules into if-else statements for ESP32 compatibility.

**5.2 Regression for Next-Day Temperature and Humidity Prediction**

**Why Regression?**

* Predicts continuous values (temperature & humidity) instead of binary classification.
* Uses historical trends: Based on past temperature & humidity patterns.
* Low computational overhead: Uses a simple equation for embedded implementation.

**Training Process in Python:**

1. Dataset: Contained Temperature, Humidity, Next-Day Temperature, and Next-Day Humidity.
2. Model Selection: Trained Linear Regression model.
3. Extracted Coefficients: Retrieved weights and intercepts from the model.
4. Model Deployment: Implemented the extracted equation in ESP32 for real-time predictions.

**Regression Equation (Implemented in ESP32):**

Next\_Temperature = (coef1 \* Temperature + coef2 \* Humidity) + intercept;

Next\_Humidity = (coef1 \* Temperature + coef2 \* Humidity) + intercept;

**Real-Time Clock and Date Handling**

* **NTP Client** fetches real-time data via WiFi.
* Converts **Epoch time** into **human-readable format**.
* Handles **date increment** properly using struct tm instead of manual calculations.

**Results and Observations**

* The watering decision was correctly classified based on real-time sensor readings.
* The next-day temperature and humidity predictions aligned well with expected values.
* The OLED display provided an easy-to-read real-time monitoring system.
* Using NTP time synchronization, the system maintained an accurate timestamp.