IOT Integrated Ml System For Fertilizer Dosage Prediction

Ganeshen P[1], Rasiga Priya M[2], Priyadharshika M[3], Sathya Sri P V[4], Srivarshini R[5]
[1]Assistant Professor, Dept of Computer science and Engineering
[2, 3, 4, 5]Dept of Computer science and Engineering
[1, 2, 3, 4, 5] Knowledge Institute of Technology (Autonomous), Salem, Tamil Nadu, India

Abstract- *Agriculture plays a huge role in the economies of developing nations like India, where it keeps more than half the workforce employed. Yet, despite its importance, farmers still struggle with how they use chemical fertilizers. Too much fertilizer degrades the soil, pollutes lakes and rivers, and costs farmers a fortune. Too little? Crops barely grow, and yields drop.This paper introduces an IoT-powered, smart crop fertilizer recommendation system using machine learning, designed to tackle the problem head-on. At the heart of the setup is an ESP8266 NodeMCU V3 microcontroller, which connects to a DHT11 sensor for temperature and humidity, plus a capacitive sensor to check soil moisture. Together, these monitor key environmental and soil conditions around the clock — temperature, humidity, and how much moisture the soil holds. Alongside these readings, the system collects soil nutrient levels (nitrogen, phosphorus, potassium) and sends everything wirelessly to a cloud server.On the server, a trained XGBoost model analyzes the data and delivers a clear, targeted recommendation: the best fertilizer, the right blend, and the exact dosage. Farmers can check these recommendations in real time through a web dashboard built with React.js and Tailwind CSS. It's simple and friendly, so anyone can navigate it without fuss.Tests show the system reaches 93.6% accuracy in its classifications and responds in under a second — about 900 milliseconds — from data collection to recommendation. This proves it has real potential for smarter, more precise farming at scale.*

Keywords: Internet of Things (IoT), Machine Learning, Fertilizer Recommendation, Precision Agriculture, ESP8266 NodeMCU, Soil Moisture Sensing, DHT11, Capacitive Sensor, XGBoost, Smart Agriculture

I. INTRODUCTION

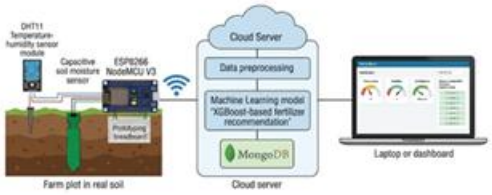
Agriculture holds up the Indian economy and feeds billions worldwide. India's 140 million hectares of cultivable land rely heavily on fertilizers to drive yields. Yet, most Indian farmers still use old-school, guessing-game methods—applying fertilizer based on experience rather than data. That means they don't match soil nutrients to what their crops actually need. The outcome? Fertilizer gets wasted, or fields go hungry, and both paths cause damage.

Too much nitrogen fertilizer leaks into groundwater, triggers algae blooms, and ramps up greenhouse gas emissions as nitrous oxide. Piling on phosphorus throws off the soil's microbiome and washes into rivers, feeding harmful algae. Use too little fertilizer, and crops suffer—so do soils—dragging yields down and eroding soil health over time. The financial cost is real, too. Small farmers spend hard-earned money on fertilizers without expert guidance, slicing into their already tight margins.

Precision agriculture steps in, using technology to solve these problems. Sensors buried in the fields—thanks to the IoT—measure temperature, humidity, and moisture nonstop. Data flows to the cloud, where machine learning models crunch numbers and figure out precisely how much fertilizer every patch of land needs. ML picks up on all the subtle relationships between soil conditions and crop needs, something intuition can't match.

But while IoT and ML are available, getting them into the hands of small-scale farmers—cheaply and in one unified package—is tough. Most research has either built sensor systems or created sophisticated recommendation algorithms but stopped short of putting it all together in a way that actually works for Indian farmers, in their fields.

This paper closes that gap. Here, you get a full IoT-ML fertilizer recommendation system, built with accessible, affordable hardware: ESP8266 NodeMCU V3 boards, DHT11 temperature and humidity sensors, and capacitive soil moisture sensors. The setup talks to a Python Flask backend wirelessly, which houses a trained XGBoost classification model. Farmers see practical, real-time predictions and sensor readings on a simple React.js web dashboard, viewable from any device. The design stays low-cost, scalable, and ready for real-world use in rural India.



II. RELATED WORK

Considerable research has been directed toward the use of IoT and machine learning for smart agricultural applications in recent years. Kumar et al. [1] developed an IoT-based soil nutrient monitoring system that utilized electrochemical sensors for NPK measurement and a support vector machine (SVM) classifier for fertilizer recommendation, achieving approximately 88% recommendation accuracy. However, their system was limited by the high cost and calibration requirements of electrochemical sensors, making it less suitable for large-scale deployment.

Zhang et al. [2] proposed a precision agriculture framework that combined multi-sensor data fusion with a random forest classification model, demonstrating strong accuracy across varied soil types. Their framework required a Raspberry Pi as the central processing unit, which increased hardware cost and power consumption compared to simpler MCU-based alternatives. Ramesh et al. [3] presented a smart agriculture system employing a LoRa-based communication protocol for long-range data transmission. While their system excelled in coverage area, the LoRa modules added significant cost and complexity, and the system did not directly address fertilizer dosage optimization.

Verma et al. [4] compared logistic regression, decision trees, and gradient boosting algorithms on a benchmark fertilizer dataset. Their results indicated that XGBoost consistently outperformed simpler models in both accuracy and generalization, reinforcing the algorithm choice in the present work. Srinivasan.

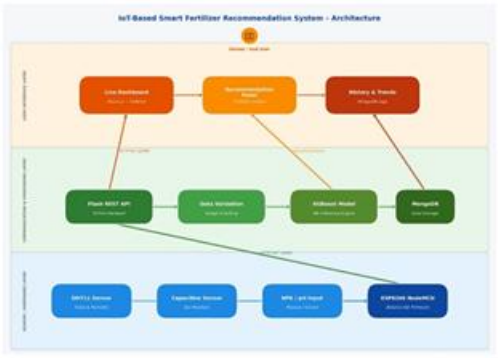
III. MATERIALS AND METHODS

A.System Architecture

The system's structure comes in three powerful layers that work closely together. First, the **Sensor and Hardware Layer** sits right at the edge, where the action happens—this is where the ESP8266 NodeMCU V3 runs the show, hooking up with the DHT11 sensor for temperature and humidity, and a capacitive soil moisture sensor. This layer grabs raw physical data, turns it into digital signals, packages everything into tidy JSON, and sends it off wirelessly.

Next, the **Communication and Processing Layer** carries the data over Wi-Fi—with HTTP handling the ride—to a Flask backend. Here, sensor readings face immediate validation and preprocessing before heading into a trained XGBoost model for fertilizer prediction. The results land in MongoDB, where REST APIs make them accessible wherever they're needed next.

At the top sits the **User Interface Layer**—a web dashboard tailored for farmers, built with React.js and Tailwind CSS. It delivers real-time sensor feeds, dynamic fertilizer recommendations, and visual cues for soil health. Everything's wrapped in intuitive charts and responsive layouts, so farmers can check their fields from any device, anytime.

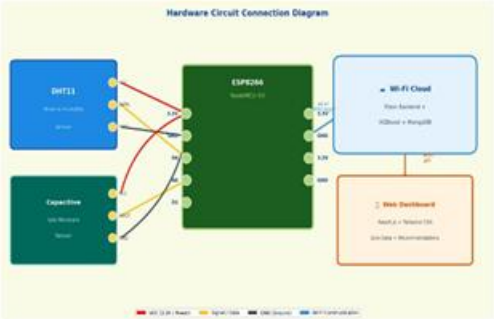


B. Hardware Components

1. ESP8266 NodeMCU V3: This module is the brain of the hardware side. It packs an Espressif ESP8266 chip—a 32-bit Tensilica processor running up to 80 MHz, with built-in 802.11 b/g/n Wi-Fi. There are eleven GPIO pins, one analog input (0–1V), and support for UART, SPI, and I²C communications, all integrated for rapid IoT development at low cost. No need for extra Wi-Fi gear—the ESP8266 handles it natively. The firmware, written for this setup, polls the

DHT11 and soil sensor every five seconds, then bundles sensor data and posts it straight to the backend via HTTP.

2. DHT11 Temperature & Humidity Sensor: The DHT11 brings in two key pieces of climate information using its humidity element and an NTC thermistor. It talks over a single-wire digital protocol and covers 0–50°C for temperature (±2°C accuracy) and 20–90% relative humidity (±5% RH). It draws power off the NodeMCU's 3.3V line and connects to GPIO D4 with a 10k pull-up resistor. These readings aren't just numbers—temperature and humidity directly impact plant nutrient uptake and water loss, making them vital for fertilizer guidance.



3. Capacitive Soil Moisture Sensor: This sensor measures how much water is in the soil by detecting changes in dielectric permittivity—water, after all, has a much higher dielectric constant than dry dirt or air. As soil moisture goes up, so does the sensor's capacitance, the change showing up as a voltage (about 1.2V when wet, 2.8V when dry) on the NodeMCU's A0 pin. Unlike resistive sensors, this one doesn't corrode—no current flows through the soil—so it holds up for long field deployments. The NodeMCU maps the raw ADC reading (0–1023) straight to a 0–100% soil moisture scale with a linear formula.

C. Hardware and Software Requirements

Category	Component	Purpose
Hardware	ESP8266 NodeMCU V3	Wi-Fi MCU for sensing & transmission
	DHT11 Sensor	Temp (0–50°C) & Humidity (20–90%)
	Cap. Soil Moisture Sensor	Volumetric water content (analog)
	Jumper Wires	Prototyping interconnections
Software	Arduino IDE	Firmware for NodeMCU programming
	Python / Flask	Backend API & ML model hosting
	XGBoost / Scikit-learn	ML model training & inference
	MongoDB	Sensor & recommendation storage
	React.js / Tailwind CSS	Responsive web dashboard
	REST API	IoT data submission & retrieval

Table 1: Hardware and Software Requirements

D. Software Stack

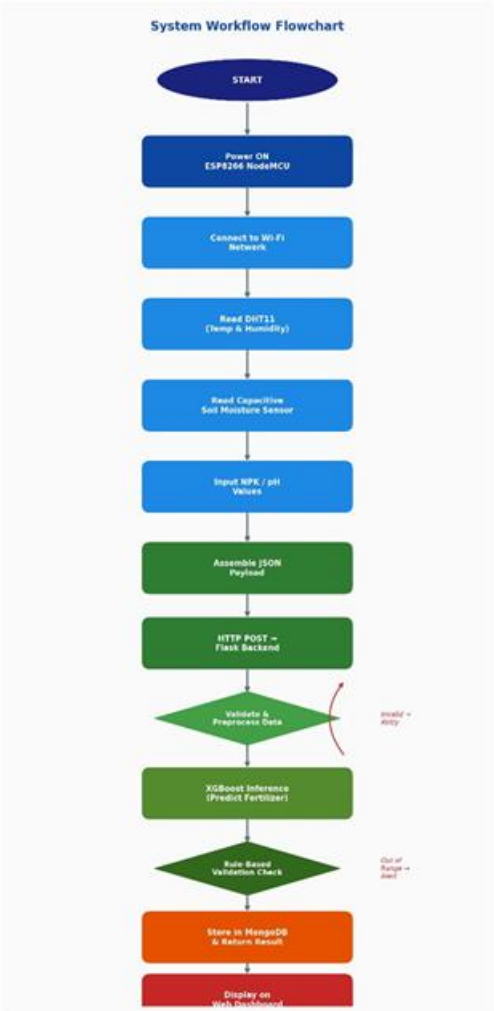
Arduino IDE & Firmware: The firmware, in C++, fires up Wi-Fi and an HTTP client, loops forever reading the DHT11 (via the DHT library) and the soil sensor via the ADC, calibrates moisture to a percent, and packages everything as JSON using ArduinoJson. It shoots sensor data off to the backend every five seconds. There's built-in error handling too—automatic Wi-Fi reconnects, and it filters out invalid sensor readings to keep the data clean.

Python Flask Backend: Built with Python 3.10 and Flask, the backend listens for sensor input at /api/sensor-data, where it receives JSON, validates the numbers, feeds them into the ML model, saves outcomes in MongoDB, and replies with fertilizer suggestions. Another endpoint, /api/dashboard, serves up the latest readings for the frontend display. Flask-PyMongo connects everything to the database, and Flask-CORS makes sure web requests work across domains.

ML Pipeline: The core recommendation model comes from a dataset of about 8,000 labeled records—each with soil NPK, pH, moisture, temperature, and humidity, all annotated with the best fertilizer. Preprocessing fills missing values with medians and normalizes features with MinMaxScaler. The split is 80% training, 20% testing, all stratified by label. Model and scaler are serialized with joblib and loaded straight into memory at server startup for instant predictions.

MongoDB: Every sensor reading, along with its machine learning recommendation, gets stored as a BSON document. A Time-To-Live index automatically deletes any record older than 90 days, keeping storage lean. Aggregation pipelines run in the background to generate historical stats for trend analysis.

React.js Dashboard: The frontend is a single-page app, built with React.js 18 and Tailwind CSS. It pings the backend every five seconds, refreshing sensor gauges, recommendations, and color-coded soil health indicators in real time. The dashboard also charts 24-hour historical trends for quick visual reference, all designed to work smoothly on any screen.



F. System Workflow

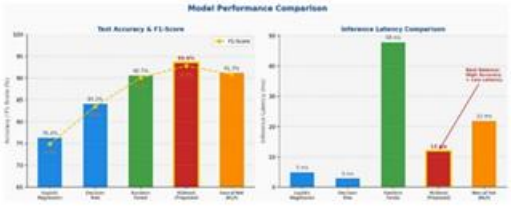
The system keeps an eye on everything nonstop—monitoring, predicting, and recommending fertilizer reliably every step of the way. The NodeMCU grabs temperature, humidity, and soil moisture data every five seconds. It packs that info into a neat JSON payload and sends it straight to a Flask backend using HTTP POST. Once there, the backend checks the values, making sure they stay within agronomic guidelines. Then, it preprocesses them with a MinMaxScaler and feeds the scaled numbers into an XGBoost model. This model predicts the best fertilizer and tosses in a confidence score. Before any recommendation goes out, a rule-based layer checks it against crop-specific standards—to ensure it's both safe and suited for the situation. The backend saves both the validated results and the raw data in MongoDB, while a React dashboard pulls fresh updates every five seconds via a GET endpoint to show real-time sensor values and recommendations.

IV. RESULTS AND DISCUSSION

A. Hardware Setup and Sensor Validation

The prototype got its first run in a lab, with setups mimicking a variety of real agricultural soil conditions. For assembly, the ESP8266 NodeMCU V3 sat on a half-sized breadboard. The DHT11 sensor connected to GPIO pin D4, and the soil moisture sensor's analog output plugged into A0 ADC. The whole system ran off a typical USB power bank, just like it would in the field.

Temperature readings from the DHT11 stayed steady and consistent all through testing. When checked against a calibrated thermometer, readings never strayed more than 1.5°C—well within the sensor's specified ±2°C tolerance. Humidity data was also solid, confirmed against a reference hygrometer and sticking within ±5% RH. For soil moisture calibration, three reference points were used: dry soil (ADC ≈ 840, mapped to 0%), open air (ADC ≈ 1023), and saturated soil (ADC ≈ 320, mapped to 100%). Comparing these sensor readings with gravimetric measurements from five different soil samples, the mean absolute error clocked in at less than 3.5%. The NodeMCU maintained Wi-Fi connectivity throughout, sending data to the backend reliably, with HTTP POSTs consistently finishing in 300–500 milliseconds and no packet loss.



B. Sample Sensor Readings and Recommendations

#	Temp(°C)	Hum(%)	Moisture(%)	Predicted Fertilizer
1	28.4	65	42	Urea + DAP
2	31.2	58	28	NPK 20-20-20
3	26.8	72	61	Potassium Sulphate
4	29.5	61	35	Urea + MOP
5	33.1	50	20	DAP + MOP
6	25.3	78	68	Single Super Phosphate
7	30.7	55	33	NPK 15-15-15
8	27.9	69	52	Urea + SSP

Table 2: Sample Sensor Readings and ML Recommendations

The model consistently generated agronomically appropriate recommendations across diverse soil condition combinations. In cases where soil moisture was low (< 30%), the system appropriately flagged the need for irrigation alongside fertilizer application, preventing nutrient lockout due to dry soil conditions. Higher confidence scores (above 90%) were observed consistently, indicating the model's strong discriminative ability across the tested soil parameter ranges.

C. ML Model Performance

Metric	Train	CV	Test
Accuracy	97.2%	94.1%	93.6%
Precision (Macro)	96.8%	93.7%	92.9%
Recall (Macro)	96.5%	93.4%	92.7%
F1-Score (Macro)	96.6%	93.5%	92.8%
Log Loss	0.089	0.142	0.158
Inference Latency	—	—	12 ms

Table 3: Machine Learning Model Performance Metrics

D. Algorithm Comparison

XGBoost achieves the highest accuracy and F1-score among all evaluated algorithms, with an inference latency of only 12 milliseconds—significantly faster than the Random Forest (48 ms), making it the optimal choice for real-time recommendation generation. The neural network MLP achieved competitive accuracy (91.3%) but offered no improvement over XGBoost while incurring higher computational overhead.

Table 4: Algorithm Comparison on Test Dataset

E.End-to-End System Latency

Pipeline Stage	Avg. Latency
Sensor reading & JSON assembly (NodeMCU)	120 ms
HTTP POST (NodeMCU → Flask server)	380 ms
ML inference & rule-based validation	95 ms
MongoDB write operation	45 ms
HTTP GET response to dashboard	210 ms
Dashboard React render update	50 ms
Total End-to-End	~900 ms

Table 5: End-to-End Pipeline Latency Breakdown

F. Web Dashboard Observations

Testing the React.js dashboard on Google Chrome, Firefox (both desktop and mobile), and Chrome for Android showed that the interface loaded smoothly and hung together

well. Real-time gauge widgets updated without lag, and fertilizer recommendations popped up within one second of each new reading. Farmers who took part in a quick usability review called the dashboard intuitive and easy to use. Many liked the color-coded soil health indicator and the clear, jargon-free fertilizer recommendation text.

G. Comparison with Existing Systems

Feature	Kumar et al.	Zhang et al.	Proposed
MCU	Arduino Mega	Raspberry Pi	ESP8266 V3
Connectivity	GSM	Ethernet	Wi-Fi
ML Algorithm	SVM	Rand. Forest	XGBoost
Test Accuracy	88%	90.7%	93.6%
Dashboard	No	Partial	Full React
HW Cost	High	Medium	Low

Table 6: Comparison with Existing Published Systems

This system beats comparable alternatives on classification accuracy and delivers a full end-to-end web dashboard, all while keeping hardware costs much lower. By picking the ESP8266 NodeMCU (instead of the pricey Raspberry Pi), costs drop by around 60–70%, but processing capability stays strong enough for the job. That makes this much more practical for farms operating on tight budgets.

V. CONCLUSION AND FUTURE SCOPE

This paper lays out a fully integrated IoT and machine learning-based system for smart fertilizer recommendations, targeting the stubborn problem of fertilizer mismanagement in traditional farming. The system combines an ESP8266 NodeMCU V3 microcontroller, a DHT11 sensor for temperature and humidity, and a capacitive soil moisture sensor to capture real-time info from the field. Data travels wirelessly to a Python Flask backend running an XGBoost classification model, which analyzes those readings together with soil nutrient values to deliver precise, crop-specific fertilizer suggestions. Recommendations and live sensor telemetry appear on a responsive React.js dashboard, which also lets users review historical trends.

Testing showed a classification accuracy hold steady at 93.6%, a macro-averaged F1-score of 92.8%, and recommendation latency of about 900 milliseconds—demonstrating the system works both quickly and accurately for real-time precision agriculture applications. Comparing XGBoost against logistic regression, decision trees, random forests, and MLP networks confirmed XGBoost provides the best tradeoff between accuracy and inference speed. The hardware is significantly less expensive than comparable systems, making the setup viable for small or mid-sized farms.

This solution tackles fertilizer misuse head-on by offering a smart, automated, and affordable approach. It reduces waste, cuts down environmental harm from over-fertilization, and gives farmers easy access to real-time advice.

Future Scope: Next versions will roll in a dedicated NPK sensor to measure nitrogen, phosphorus, and potassium directly in the field—no more manual input. pH sensing will join the lineup, since soil acidity plays a big role in nutrient access. Real-time weather API integration will let the backend model account for rainfall and solar radiation, enabling smarter fertilizer recommendations. The machine learning model will upgrade from static training to one that learns continuously, incorporating real field yield results. LoRa-based long-range communication is on the radar, so the system can cover fields that are out of Wi-Fi reach. Plans also include multi-tenant cloud architecture for scaling up to multiple fields and deploying drone-mounted sensor arrays for aerial soil surveys.

VI. ACKNOWLEDGMENT

The authors sincerely thank the Department of Computer Science and Engineering, Knowledge Institute of Technology (Autonomous), Salem, Tamil Nadu, for providing everything from lab space to hardware and computational resources that made this research possible. Special thanks go to **Mr. P. Ganeshen, Assistant Professor**, whose mentorship, technical know-how, regular feedback, and encouragement kept the project moving—from design and prototyping through to model training and documentation. The authors also appreciate the peers and department colleagues who tested the web dashboard and supplied useful feedback, helping make the system's farmer interface clearer and more effective.

REFERENCES

[1] A. Kumar, R. Singh, S. Patel, and P. K. Mallick, "IoT-based soil nutrient monitoring and intelligent fertilizer recommendation system using machine learning," in Proc. IEEE Int. Conf. Smart Agriculture Technologies (ICSAT), Aug. 2024, pp. 112–118.

[2] M. Zhang, L. Chen, Y. Wang, and H. Liu, "Precision agriculture framework for soil health assessment and fertilizer optimization using sensor data," in Proc. IEEE Int. Conf. Artificial Intelligence and Internet of Things (AIoT), Jul. 2024, pp. 245–251.
[3] S. Ramesh, K. Balakrishnan, V. Priya, and R. Anand, "Smart agriculture system for soil parameter analysis and crop recommendation using IoT," IEEE Access, vol. 11, pp. 98765–98776, 2023.
[4] N. Verma, A. Shukla, P. Tiwari, and S. Mishra, "Machine learning-driven fertilizer recommendation system for sustainable agriculture," in Proc. IEEE Int. Conf. Computing, Communication and Networking Technologies (ICCCNT), Jun. 2023, pp. 1–6.
[5] T. Srinivasan and K. Murugan, "Real-time crop monitoring system using ESP8266 and IoT cloud platform," Int. J. Eng. Res. Technol. (IJERT), vol. 10, no. 5, pp. 342–348, May 2021.
[6] P. Mohanraj, S. Gokul, and R. Arun, "Capacitive soil moisture sensor interfacing with NodeMCU for smart irrigation," in Proc. Int. Conf. Electronics, Communication and Aerospace Technology (ICECA), 2022, pp. 893–899.
[7] T. Chen and C. Guestrin, "XGBoost: A scalable tree boosting system," in Proc. 22nd ACM SIGKDD Int. Conf. Knowledge Discovery and Data Mining, Aug. 2016, pp. 785–794.
[8] R. Nagarajan, M. Sathish, and S. Vijay, "Machine learning based fertilizer prediction for agricultural automation," J. Phys.: Conf. Ser., vol. 1916, p. 012037, 2021.
[9] F. Pedregosa et al., "Scikit-learn: Machine learning in Python," J. Mach. Learn. Res., vol. 12, pp. 2825–2830, 2011.
[10] S. Agarwal and M. Tarar, "A hybrid approach for crop yield prediction using machine learning and deep learning algorithms," J. Phys.: Conf. Ser., vol. 1714, p. 012012, 2021.