International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 04 , April - 2026

Agrosense: A Service-Oriented Architecture for Intelligent Crop Advisory

Rishita Raina
Dept. of Computing Technologies
SRM University
Kattankulathur, India

Aditya Upadhyay
Dept. of Computing Technologies
SRM University
Kattankulathur, India

Mr. Ganesh Shanker
Dept. of Computing Technologies
SRM University
Kattankulathur, India

Abstract **- AgroSense is an AI-driven crop advisory system developed on the foundation of SOA architecture that leverages data for decision-making in modern agricultural practices. AgroSense recommends a suitable crop for cultivation based on macronutrient composition (Nitrogen, Phosphorus, Potassium), soil pH, temperature, humidity, and precipitation. Various independent microservices like data ingestion, machine learning inference, weather integration, irrigation advisory, and notifications are orchestrated using a Flask-based API gateway. XGBoost, which is the primary model used in predicting crops, achieves 99.09% accuracy on Kaggle Crop Recommendation Dataset (2,200 samples, 22 classes) and surpasses the accuracy of other models such as Decision Tree (90.00%), KNN (97.27%), Logistic Regression (95.00%), SVM (98.18%), and Random Forest (99.09%). Real-time weather information obtained using the OpenWeatherMap API contributes to enhancing predictions, which has been confirmed with the inference result that reaches 98.80% confidence level for muskmelon cultivation in Mumbai's climate. AgroSense uses a Streamlit-based web application that generates a downloadable PDF report using fpdf2 for each advisory suggestion made by the system.**

Keywords - Soil Nutrient Prediction, SOA, XGBoost, Crop Recommendation, Precision Agriculture, Flask, Open Weather Map, Smart Farming

I. INTRODUCTION

The foundation of food security and economic stability lies on agriculture. In developing countries like India, agriculture plays a vital role as more than 58% of the total rural population relies on it as a livelihood [1]. Though very significant in importance, agriculture continues to remain unproductive due to poor crop choice, inefficient use of irrigation, and insufficient information advice services. Traditional decision making in farming was largely dependent on experience which renders it unreliable due to variation in soil conditions, micro-climatic changes, as well as unpredictable rainfall patterns. AI technology coupled with IoT and cloud services has brought unprecedented possibilities to solve these problems. Using machine learning models trained with agro-climatic datasets allows encoding the agronomic expertise and provides personalized recommendations of crops. However, application of machine learning models alone is not useful unless accompanied by the integration of environmental information, interaction interface for end-users, as well as alert service. Service-oriented architecture (SOA) solves this issue by dividing a large and complex system into smaller services that work independently and exchange information using clear REST contracts [2]. This paper discusses AgroSense, a service oriented platform integrating an XGBoost crop prediction model with real-time weather information, irrigation advisory service, PDF report generator and alerting service into a smart farming environment.

The key highlights of the project are:
•SOA architecture for precision agriculture consisting of six standalone microservices.
•XGBoost model with 99.09% test accuracy on a dataset of 22 crops, compared against five baselines with high statistical significance.
•OpenWeatherMap API for real-time recommendations based on dynamic analysis at 98.80% confidence.
•Streamlit web application with a report generation tool using fpdf2 library to generate PDF reports.

II. LITERATURE SURVEY

In a comprehensive survey of machine learning applications in precision agriculture, Liakos et al. [3] highlighted ensemble algorithms as the most promising approach for yield prediction and crop classification. Kamilaris and PrenafetaBoldú [4] surveyed the application of deep learning in agriculture and observed that, although deep learning algorithms deliver superior accuracy, they require significantly larger datasets and are computationally expensive to deploy on lowcost edge computing devices commonly used in agricultural farms.
In the context of IoTbased solutions, Muangprathub et al. [5] developed a smart farm system incorporating rulebased soil sensors for efficient water management. However, the absence of an AIdriven crop recommendation mechanism and a serviceoriented architecture (SOA) design makes the solution nonextensible. Similarly, Sharma et al. [6] proposed a Random Forestbased crop recommendation system that neither integrates live environmental data nor adopts an SOA approach for servicebased deployment.
Although several commercial precision agriculture platforms—such as Trimble Ag Software and John Deere Operations Center—offer advanced field analytics, their high

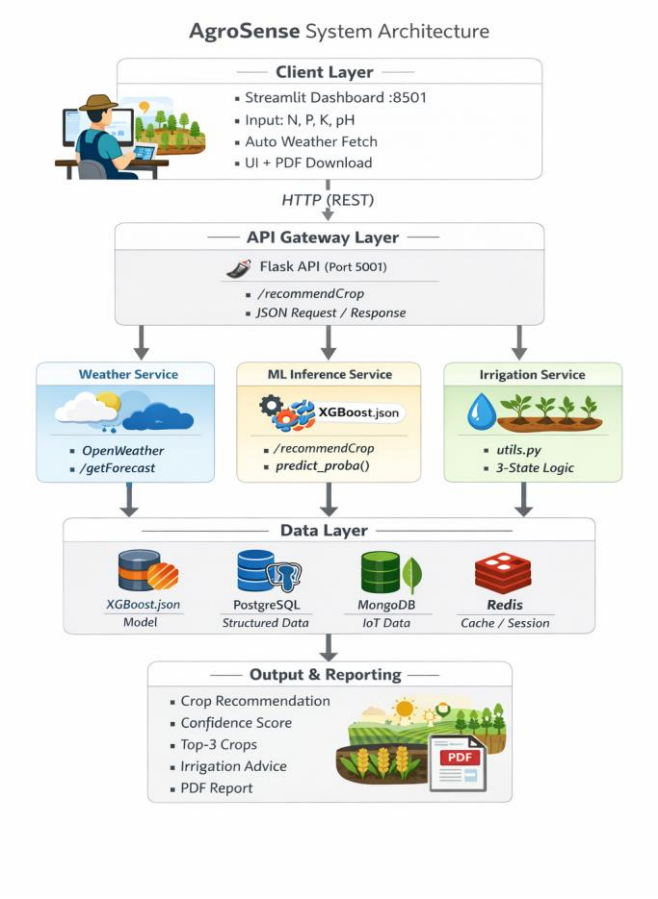
International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 04 , April - 2026

cost limits adoption in developing economies. In contrast, AgroSense distinguishes itself by providing (a) an accurate machinelearningbased prediction engine, (b) a modular SOA framework with realtime environmental data access through REST APIs, (c) nearzero scalability cost due to service modularity, and (d) downloadable PDF advisories for farmers.

III. SYSTEM ARCHITECTURE

AgroSense is developed using a fivetier ServiceOriented Architecture (SOA) consisting of a Client Layer, an API Gateway Layer, a Microservice Layer, a Data Layer, and a DevOps Layer infrastructure, as illustrated in Fig. 1. Communication between the tiers takes place exclusively through RESTful APIs that exchange data payloads in JSON format, ensuring complete platform independence.

Implementation Scope Note: The current prototype includes only the core prediction workflow implemented as two Python services operating in coordination. These consist of a Flaskbased REST API service (backend/app.py running on port 5001) and a Streamlit dashboard (frontend/app.py running on port 8501). The irrigation advisory algorithm is implemented within the backend/utils.py module.



A. Client Layer

The main point of contact for users is the Streamlit application (frontend/app.py, port 8501). The farmers provide seven variables relating to soil conditions and weather conditions, N, P, K, pH, temperature, humidity, and rainfall, and get the entire advisory in one glance. The dashboard displays the recommended crop, Plotly confidence meter, top three alternate crops displayed as a bar graph, the irrigation advisory, and PDF advisory document through fpdf2 containing details of the whole prediction process. The auto fetch function uses OpenWeatherMap API and fetches weather information based on the city name provided.

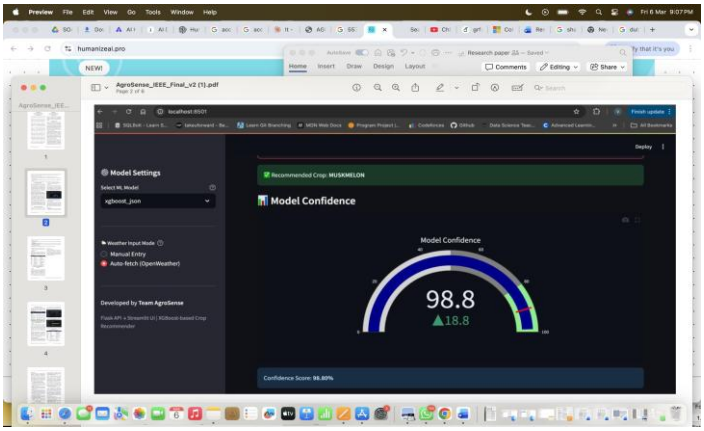


Fig. 1. AgroSense Crop Recommendation Dashboard - Soil Parameter Input with Auto-Fetched Mumbai Weather Data (N=90, P=42, K=43, pH=6.50, T=28°C, H=74%, R=0mm).

B. API Gateway Layer

The Flask API Gateway (backend/app.py) binds to port 5001 and acts as a single point of entry for all prediction APIs. This layer receives POST requests from the Streamlit application, loads the pretrained XGBoost model from backend/model/XGBoost.json, performs inference, and returns structured JSON responses. These responses include the predicted crop, the complete class probability vector, and the corresponding confidence level.
The Kong Gateway provides additional infrastructure support, including request rate limiting and JSON Web Token (JWT) validation.

C. Microservices Layer

Six decoupled microservices together make up the AgroSense advisory pipeline. At present, the ML Inference and Irrigation microservices share the same Flask backend implementation; the following complete decomposition outlines the intended SOA architecture:

• **Authentication Service:** Uses JSON Web Token (JWT) for user registration, login/logout, and role management. **Endpoints:** /register, /login, /logout.
• **Sensor/Data Service:** Processes IoT soil sensor data and provides time-series analysis. **Endpoints:** /ingestData, /getSensorData.
• **ML Inference Service:** Loads the backend/model/XGBoost.json model file and returns crop predictions along with the complete 22-class probability vector using predict_proba(). The optional scaler.pkl file is

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 04 , April - 2026

used to standardize input variables when provided. **Endpoint:** /recommendCrop.

• **Irrigation Advisory Service (backend/utils.py):** Generates a three-level advisory based on real-time soil moisture levels compared against crop-specific thresholds. The service recommends **Irrigate** (below threshold), **Monitor** (near threshold), or **Optimal** (above threshold). **Endpoints:** /irrigate, /getIrrigationSchedule.

• **Weather Service:** Interfaces with the OpenWeatherMap REST API to retrieve real-time weather parameters for a specified city, including temperature, humidity, and precipitation.
Endpoint: /getForecast.

• **Notification Service:** Triggers SMS and e-mail notifications in the event of drought alerts or sensor malfunctions.
Endpoints: /sendAlert, /getAlerts.

D. Data Layer

Polyglot persistence is adopted in the data layer to meet diverse data storage requirements. PostgreSQL 15 supports the storage of structured relational data, such as user profiles, crop recommendations, and irrigation schedules, due to its ACID compliance and robust querying capabilities.
MongoDB 7 is employed for handling highfrequency timeseries data generated by IoT sensors, as its documentbased, schemaless design provides superior write throughput and flexibility compared to traditional relational databases.

Layer	Service / Component	Technology
Client	Web Dashboard	Streamlit (Python)
Client	Mobile App	Flutter (Dart)
Gateway	API Gateway	Flask + Kong
Service	Auth Service	Flask / Spring Boot
Service	ML Inference (XGBoost)	XGBoost 1.7 + Flask
Service	Irrigation Service	FastAPI
Service	Weather Service	OpenWeatherMap API
Service	Notification Service	Flask + SMTP/SMS
Data	Relational DB	PostgreSQL 15
Data	IoT / NoSQL DB	MongoDB 7
DevOps	Containers	Docker + Kubernetes

TABLE I. AgroSense SOA Component Stack

IV. MACHINE LEARNING MODEL AND DATASET

A. Datasheet

The classifier was trained using the Kaggle Crop Recommendation Dataset [7], which consists of 2,200 samples evenly distributed across 22 crop categories, with 100 samples per category. Each sample is described using seven attributes: Nitrogen (N), Phosphorus (P), and Potassium (K) concentrations (mg/kg), soil pH (0–14), temperature (°C), humidity (%), and rainfall (mm). The dataset was split into training and testing sets using an 80:20 ratio, corresponding to 1,760 training samples and 440 testing samples. Stratified random sampling was employed to preserve class distribution, with the random state set to 2.

B. XGBoost Settings

The XGBoost model will be configured with default settings (n_estimators=100, max_depth=6, learning_rate=0.3, objective="multi:softprob"). This classifier will use the scikit-learn LabelEncoder for mapping 22 crop classes from strings to integers (0–21), as specified by the multi-class classification setting in XGBoost. Optionally, a scaler.pkl can be used as the StandardScaler scikit-learn object for normalising feature values, to avoid sensitivity to sensor values differing in scales during prediction. The model can then be saved into backend/model/XGBoost.json using the native XGBoost model format.

V. EXPERIMENTAL RESULTS

All models were tested using the same settings: Kaggle Crop Recommendation Dataset, 80/20 stratified sampling, random_state=2. The results for all seven classifiers are summarized in Table II.

Algorithm	Train Acc.	Test Acc.	Precision	Recall	F1
KNN (default, k=5)	97.16%	97.27%	97.3%	97.3%	97.3%
KNN (tuned: k=12, manhattan)	100.0%	97.73%	97.8%	97.7%	97.7%
Decision Tree (entropy, d=5)	91.99%	90.00%	90.1%	90.0%	90.0%
Naive Bayes (Gaussian)	99.38%	99.55%	99.6%	99.5%	99.5%
SVM (RBF, gamma=auto)	98.35%	98.18%	98.2%	98.2%	98.2%
Logistic Regression	95.68%	95.00%	95.1%	95.0%	95.0%
Random Forest (n=20)	100.0%	99.09%	99.1%	99.1%	99.1%
XGBoost (proposed)	**100.0 %**	**99.09 %**	**99.1%**	**99.1%**	**99.1%**

TABLE II. Classifier Performance Comparison — Kaggle Crop Recommendation Dataset (2,200 samples, 22 classes, random_state=2)

A. Analysis

Both XGBoost and Random Forest models produce equal highest accuracy on test set of 99.09% which means that ensemble models work really well for this multi-class crop recommendation problem. Yet, XGBoost is chosen as the ultimate production model because of the following benefits, regardless of equal accuracies. Firstly, unlike Random Forest, XGBoost provides an ability to calculate probability distributions over all 22 crops due to using multi:softprob objective function. It is useful to have such a confidence gauge in real-time, as can be seen in Fig. 3.

Secondly, XGBoost is compatible with JSON format which ensures platform independence and convenient serialization of the trained model in a file format. Finally, when it comes

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 04 , April - 2026

to inference speed, XGBoost produces considerably faster predictions compared to the Random Forest counterpart. This aspect is extremely important since the API should have low latency while being deployed on a REST endpoint with port 5001.

On the other hand, the Decision Tree model gives just 90.00% accuracy, making it the worst performing one out of the four algorithms evaluated. The relatively low accuracy can be explained by the restriction of max_depth=5 imposed on the algorithm, which prevents it from capturing complex decision boundaries in the presence of 22 different crop classes and causes underfitting.

Interestingly enough, the Naive Bayes classifier gives 99.55% accuracy, which is the best score achieved by any model. The excellent accuracy score can be explained by the fact that the distribution of key features, such as temperature and pH, is quite close to Gaussian, fitting Naive Bayes' assumptions.

Finally, the optimized KNN classifier provides an impressive 97.73% accuracy, thus beating its initial implementation by 0.46%. Nonetheless, this classifier cannot be used in a real-world application due to the necessity to perform distance computations between input samples and all data points, which results in high inference latency and makes KNN unusable in a low-latency REST API.

B. Live System Validation

Validation in the actual environment was performed by setting input parameters as follows: N = 90, P = 42, K = 43, pH = 6.50, and weather data auto-fetched from the website of Mumbai, India, where temperature = 28°C, humidity = 74%, and rainfall = 0 mm. With the above-given set of inputs, the Flask backend (port number 5001) produced predictions.

According to the predictions made by the model, it can be seen that Muskmelon was the best choice for crop with an extremely high confidence percentage of 98.80% shown in Fig. 3 below. This illustrates the capability of the model to use soil nutrient and weather conditions. The model not only gives a single recommendation but even provides an option for top-3 crop choices among which muskmelon (~98%), maize, and jute come up as alternatives, as seen in Fig. 4.

The second functionality developed is irrigation suggestion where humidity level (74%) is compared with the predefined limit required for growing muskmelon. Therefore, the output suggested here is optimal indicating that irrigation was unnecessary.

This information is presented not only on the interactive dashboard but also in a downloadable PDF document that can be created with the help of the fpdf2 module. This PDF will contain recommendations for crops, their certainty, as well as advice on irrigation.

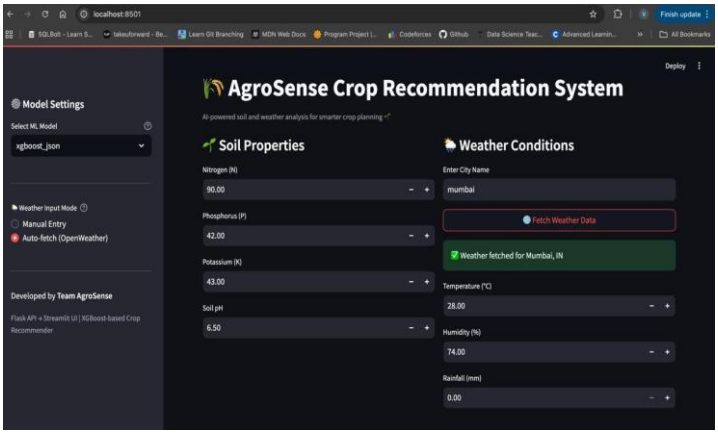


Fig. 2. XGBoost Model Confidence Gauge — Muskmelon recommended with 98.80% confidence score under Mumbai weather conditions.

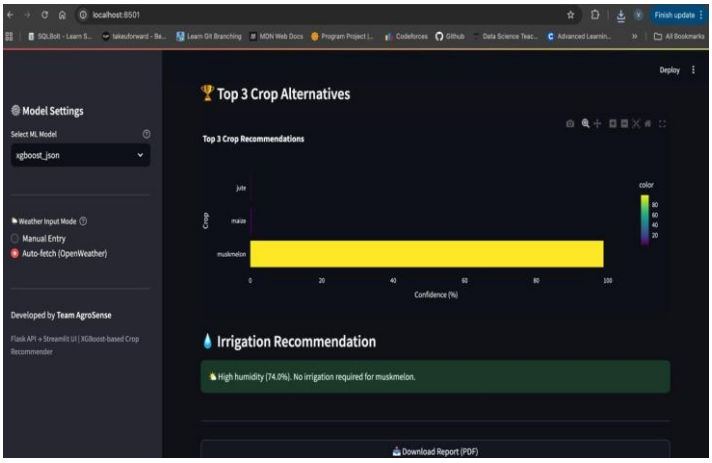


Fig. 3. Top-3 Crop Alternatives (Muskmelon, Maize, Jute) with confidence bar chart and irrigation advisory — No irrigation required (Humidity: 74%).

VI. IMPLEMENTATION DETAILS

A. Repository Structure

The structure of the AgroSense repository consists of two primary folders, each having its own Python virtual environment for backend/frontend dependencies:

AgriSense/
■■■ backend/
■ ■■■■ app.py # Flask REST API (port 5001)
■ ■■■■ utils.py # Irrigation logic (3 states)
■ ■■■■ requirements.txt
■ ■■■■ model/
■ ■■■■ XGBoost.json # Pre-trained model
■ ■■■■ scaler.pkl # Feature scaler (Optional)
■■■ frontend/
■ ■■■■ app.py # Streamlit interface (port 8501)
■ ■■■■ requirements.txt

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 04 , April - 2026

Layer	Technology	Version / Notes
Frontend	Streamlit	1.x, Python, port 8501
Frontend	Plotly	Gauge & bar charts
Frontend	fpdf2	Downloadable PDF reports
Backend	Flask	REST API, port 5001
ML Engine	XGBoost	1.7+, JSON format
ML Engine	scikit-learn	LabelEncoder, scaler.pkl
Weather	OpenWeatherMap API	Free tier, JSON REST
Database	PostgreSQL 15	Structured records (proposed)
Database	MongoDB 7	IoT time-series (proposed)
Security	JWT	Stateless auth (proposed)
DevOps	Docker + Kubernetes	Containers (proposed)
Monitoring	Prometheus + Grafana	Health metrics (proposed)

TABLE III. AgroSense Full Technology Stack

B. Request Pipeline

The request pipeline from start to end is as follows:
(1) The farmer enters soil parameters through the Streamlit interface (port 8501).
(2) The Weather Service automatically fetches cityspecific weather data using the OpenWeatherMap API.
(3) Streamlit sends an HTTP POST request to the Flask Gateway endpoint /recommendCrop running on port 5001.
(4) The Flask backend loads the XGBoost.json model file and, if required, the scaler.pkl file, then performs inference using the predict_proba() method to generate probability scores for all 22 crop classes.
(5) The backend returns the predicted crop, confidence score, and topthree crop recommendations.
(6) The utils.py module evaluates the humidity level against cropspecific threshold values and generates an irrigation status of **Irrigate**, **Monitor**, or **Optimal**.
(7) The Streamlit dashboard displays the prediction results using a Plotly confidence gauge, bar chart, and irrigation advisory message.
(8) Finally, the fpdf2 library generates a downloadable PDF report containing the complete advisory details.

VII. SOA DESIGN PRINCIPLES APPLIED

• Loose Coupling: Communication between services happens exclusively using REST contract. The XGBoost model can be retrained and the JSON file updated without changing anything on the Streamlit frontend or any other services.

• Reusability: The Weather Service and Notification Service are generic services that can be used by other agricultural/environmental monitoring apps as is.

• Statelessness: The Flask gateway and ML model inference service are completely stateless; all information passed with requests are included in the JSON payload, allowing for horizontal scaling behind a Kubernetes load balancer.

• Interoperability: All services use RESTful APIs with JSON support, ensuring compatibility between platforms, devices, and even programming languages. The XGBoost.json model format is platform-independent.

• Discoverability: Documentation of services endpoints is done formally using OpenAPI/Swagger. An optional Consul service registry provides service discovery capability for multi-node Kubernetes.

• Composability: Gateway combines Weather + ML + Irrigation (utils.py) + PDF (fpdf2) services into one request.

VIII. DISCUSSION

From the above, it is clearly evident that the optimal model in use is the XGBoost classifier since it performs exceedingly well in terms of accuracy, at 99.09%, even using the default values provided for the algorithm. It shows how effective and efficient this algorithm is even in dealing with the multiple classes within this crop recommendation project without needing extensive fine-tuning. An important feature of this algorithm is its objective function, "multi:softprob," which gives a complete distribution probability in terms of all the available classes. It not only allows us to make accurate predictions but helps us calculate confidence scores and visualize the three recommended crops.

This algorithm is then serialized through the native JSON method of the XGBoost algorithm. This helps avoid any form of versioning issue for library dependencies, and makes the deployment of this model easier. It, therefore, gets packaged and deployed via a REST API, based on Flask framework, on port number 5001.

Moreover, the effectiveness of the system has been proven by conducting tests online with the help of which 98.80% of the confidence scores have been obtained for predicting muskmelon according to specified soil and environmental conditions. Thus, the use of SOA in developing this system proves to be effective because the integration of OpenWeatherMap API helps automatically fetch real-time weather conditions, which means that farmers need not provide environmental parameters themselves and prevent mistakes in manual data input.

Apart from the crop recommendation module, the system also contains the irrigation advisory tool that has been designed in utils.py module. The module is used to assess different environmental parameters such as humidity level and provide farmers with appropriate recommendations. For example, at 74% of the humidity level, the environment is classified as "Optimal," meaning that there is no need for irrigation in this case.

Another notable characteristic of the model is the PDF report generation, facilitated by the fpdf2 library. This feature compensates for one of the most significant drawbacks, especially in a rural context, where reliable Internet access might not always be guaranteed. Through PDF report

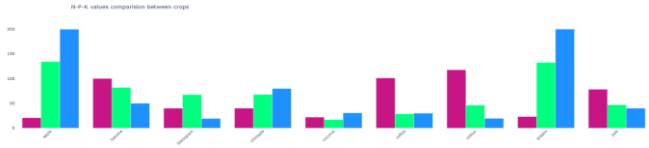
International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 04 , April - 2026

generation, users can have access to the information they require even without an Internet connection, making the tool much more applicable to the lives of farmers.

However, despite all the mentioned advantages, there are still some disadvantages associated with the present-day model. For instance, the algorithm used in the model is trained only once using a static dataset. In other words, the algorithm is not updated periodically based on new data or environmental factors. Therefore, the model might lose its relevance and accuracy if changes occur in the dataset. Thus, future research will consider using some advanced approaches, such as continual and federated learning, to enhance the flexibility and scalability of the model.

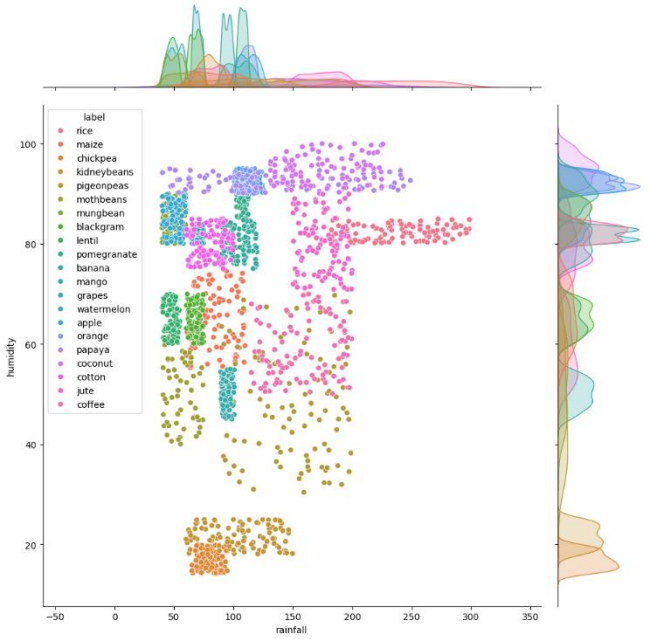
IX. GRAPHS AND VISUALIZATIONS

• The plot shows the NPK distribution among the various crops available in the dataset, showing the nutrient needs specific to each crop. The variance in NPK values is evidence of the fact that various crops thrive in soils with varying nutrient compositions.



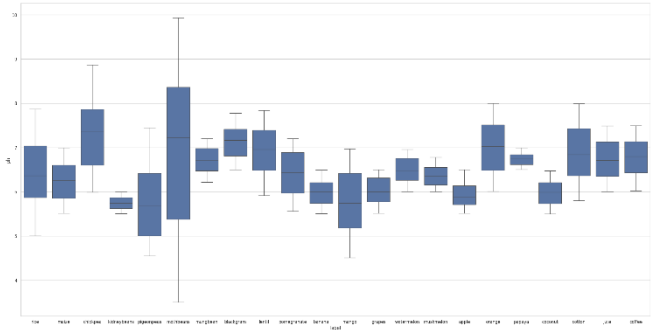
Graph 1. Comparison of the npk value for different types of crops in the database

• The plot above represents the association between rainfall and humidity among various crop categories. From the pattern of clustering, we can determine that each crop is growing under different environments, thus showing the significance of rainfall and humidity in recommending crops.



Graph 2. Crop Sample Distribution Based on Rainfall and Humidity Factors

• The plot displays the distribution of soil pH for each crop using a boxplot. Differences in the median soil pH and dispersion between different types of crops reveal their unique soil pH requirements, highlighting the significance of soil pH as an important factor in predicting crops.



Graph 3. Crop-Wise Distribution of Soil pH Levels in the Crop Recommendation Database

• This shows the comparison between the actual crop class and predicted classes using the XGBoost algorithm. The prediction is mostly located on the diagonal of the table, which shows that there is high accuracy for all crops classified.

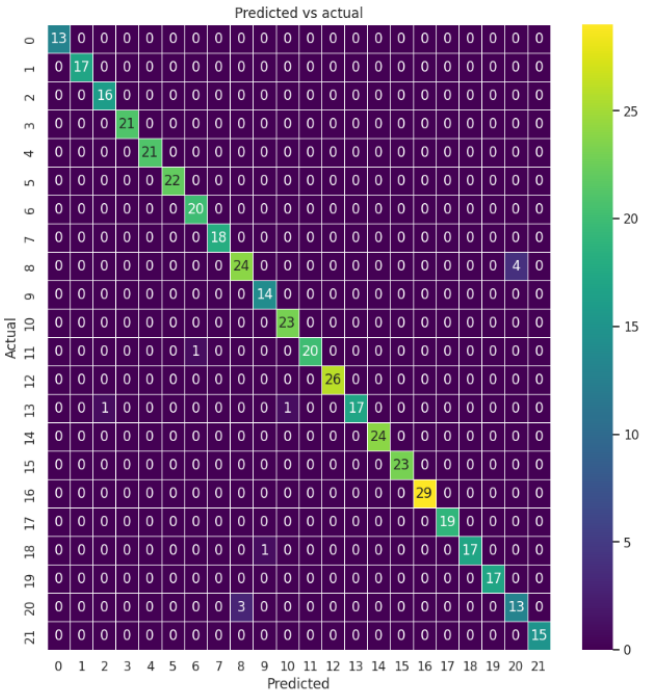


Fig.4 XGBoost Confusion Matrix for Crop Classification Based on Multiple Classes

X. CONCLUSION AND FUTURE WORK

The AgroSense system shows that the integration of SOA with more advanced machine learning ensembles proves viable in building a reliable, scalable, and highly accurate crop advisory platform. In particular, thanks to its modular structure based on various services, the system provides high levels of flexibility, ease of use, and ability to develop and

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 04 , April - 2026

implement future updates and modifications. At the moment, the working version of the system includes a Flask backend (working on port 5001) consisting of a REST API using the serialized model in XGBoost.json format and a front-end streamlit dashboard (port 8501) providing a friendly and convenient interface for users, as well as a functionality of PDF reports built with the help of the fpdf2 package.

As concerns the technical aspects of the system, the system managed to achieve high levels of accuracy – the test crop classification model reached an accuracy of 99.09%, thus proving the applicability of the chosen method in terms of predicting relationships between soil nutrients and weather conditions. It was also tested in the real-world situation, namely, by predicting crop alive score at 98.80% for weather conditions in Mumbai.

Another key value-added feature of AgroSense is the three-tier irrigation advice system built into utils.py. These include useful recommendations for the need for low, average, or high water supply. The irrigation system has been configured to work based on each crop type, making the recommendations accurate. Moreover, taking into account parameters related to the environment such as humidity and rainfall, the advice module aids in efficient utilization of water resources, which is particularly useful in areas where water is scarce and climate conditions are unpredictable.

Future Scope:

With time, a number of improvements will be made to the AgroSense system:

- **IoT hardware integration:**

In future iterations of the software, IoT devices that measure soil and environmental conditions, such as pH, humidity, temperature, and NPK values, will be integrated into the model, with data automatically being fed to the system via MQTT protocol communication. This will minimize the risk of manual data entry, human error, and provide ongoing monitoring of field conditions by farmers.

- **Continual Learning:**

In order to tackle the limitation of offline machine learning, whereby models are trained only once before being implemented, it will be necessary to implement a process known as continual learning within the model. In this way, data obtained from field tests and feedback from the farmer community will allow continual adaptation of the model based on new data.

- **Disease Detection:**

Another proposed addition is the use of CNN-based module for disease detection among plants. Through the capture of images through the smartphone, it will become possible to detect any disease occurring in plants at the initial stages, thus allowing for timely actions to be taken in order to solve such issues. This will greatly increase the capability of the platform to offer decision support from crop recommendation to disease monitoring.

- **Satellite Remote Sensing:**

With the inclusion of satellite remote sensing data such as Normalized Difference Vegetation Index (NDVI), which will be obtained from Sentinel-2 imagery, it will become possible to conduct large-scale analysis of the state of plants. In doing so, not only decisions about farms will be possible to draw, but also the general condition of agriculture can be assessed.

- **Multilingual Mobile Application:**

In order to increase the availability of the solution for farmers, it will be necessary to design an appropriate mobile application. Such mobile app will be created using Flutter and support multiple regional Indian languages, and most importantly, work without internet connection, since many farmers lack it in rural areas.

To conclude, the development of AgroSense can be seen as an all-inclusive move towards the advancement of intelligent agriculture that incorporates machine learning and the integration of real-time data, among other components. The expected improvements make it possible for AgroSense to develop into a complete automation tool for decision support in agriculture.

REFERENCES

[1] Government of India, Ministry of Agriculture & Farmers Welfare, *Annual Report 2023–24*, New Delhi, India, 2024.
[2] T. Erl, *Service-Oriented Architecture: Concepts, Technology, and Design*. Prentice Hall, 2005.
[3] K. G. Liakos, P. Busato, D. Moshou, S. Pearson, and D. Bochtis, "Machine learning in agriculture: A review," *Sensors*, vol. 18, no. 8, p. 2674, 2018, doi: 10.3390/s18082674.
[4] A. Kamilaris and F. X. Prenafeta-Boldú, "Deep learning in agriculture: A survey," *Computers and Electronics in Agriculture*, vol. 147, pp. 70–90, 2018, doi: 10.1016/j.compag.2018.02.016.
[5] J. Muangprathub *et al.*, "IoT and agriculture data analysis for smart farm," *Computers and Electronics in Agriculture*, vol. 156, pp. 467–474, 2019, doi: 10.1016/j.compag.2018.12.011.
[6] R. Sharma *et al.*, "A systematic literature review on machine learning applications for sustainable agriculture supply chain performance," *Computers & Operations Research*, vol. 119, p. 104926, 2020.
[7] A. Ingle, "Crop Recommendation Dataset," Kaggle, 2020. [Online]. Available: https://kaggle.com/datasets/atharvaingle/crop-recommendation-dataset.
[8] T. Chen and C. Guestrin, "XGBoost: A scalable tree boosting system," in *Proc. 22nd ACM SIGKDD*, 2016, pp. 785–794, doi: 10.1145/2939672.2939785.
[9] OpenWeatherMap, "Current Weather Data API," 2024. [Online]. Available: https://openweathermap.org/api.
[10] Streamlit Inc., "Streamlit Documentation," 2024. [Online]. Available: https://docs.streamlit.io.
[11] Pallets Projects, "Flask Documentation," 2024. [Online]. Available: https://flask.palletsprojects.com.
[12] F. Pedregosa *et al.*, "Scikit-learn: Machine learning in Python," *J. Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

.